Exact Geometric Collision Detection

Elmar Schömer Jürgen Sellen Markus Welsch *

Abstract

Exact computation is an important paradigm for the implementation of geometric algorithms. In this paper, we consider for the first time the practically important problem of collision detection under this aspect. The task is to decide whether a polyhedral object can perform a prescribed *sequence* of translations and rotations in the presence of stationary polyhedral obstacles. We present an exact decision method for this problem which is purely based on integer arithmetic. Our approach guarantees that the required binary length of intermediate numbers is bounded by $14L+22$, where L denotes the maximal bit-size of any input value.

1 Introduction

Exact computation is widely recognized as one of the key issues in the design of geometric algorithms in the near future. Recent work on exact algorithms focuses on traditional problems of computational geometry, such as the construction of Voronoi diagrams [9, 1, 6]. However, there are reasons for believing that exact computation will also be an important paradigm for the implementation of future CAD tools, from solid modeling to manufacturing applications.

Our interest in exact algorithms for *collision detection* is motivated by an ongoing research project [7]: In order to generate and verify assembly plans in CAD systems, there is a growing interest in the development of interactive simulation tools. The main task of such a tool is the verification of user-specified motions with respect to collision-freeness.

Assembly operations usually consist of simple motions but require high accuracy. The latter fact implies that the simulation software has to be carefully designed in order not to produce incorrect results because of rounding errors. If using fixed precision arithmetic, geometric algorithms such as collision detection have to deal with robustness issues. This requires a thorough and consistent handling of tolerances. Though these are strong arguments for an exact approach, the computational burden of adequate algorithms – which may be measured by the maximal bit-length of integer or floating-point numbers in intermediate computations – is generally considered as too high for practice. (In the following, we denote the bit-length above as precision.)

In contrast to the *algebraic model*, we assume a *bit model* in which not only the number of input values n but also their bit-length L determines the complexity of algorithms. We note that collision detection in the algebraic model has recently been solved in subquadratic time by using parametric search [8]. Our focus in this paper is however the bit model, and our main contribution is a decision scheme with surprisingly low constants for the needed precision.

There are two major problems in the design of exact collision detection algorithms.

The first problem arises because of the desire to deal with *sequences* of motions. If rotations are specified by rotation axis and angle, given by rational numbers, then successive rotations lead to algebraic numbers of increasing degree. This suggests that the considered problem is not of *bounded algebraic depth* as defined in [9], and that the number of bits that are needed to perform collision tests exactly grows exponentially. This growth of complexity has two sources: the specification of motions *relative* to each other, and the parameterization of rotations. In our scheme, we describe motions as transitions between absolutely given intermediate configurations, and use the quaternion calculus to describe rotations. This allows to decide collision-freeness of motion sequences with uniformly bounded precision $O(L)$.

The idea to use quaternions to decide collision-freeness is not new, and is e.g. used by Canny [2, 3]. But this is only one step towards a practicable exact algorithm. As the practicability crucially hinges on the constants hidden in the O-notation above, the main part is to derive clever predicates that can be decided with low precision. We finally arrive at a bound of $14L+22$. To compare this to a standard result in the field, note that Fortune's sweep algorithm to compute the Voronoi diagram for point sites in the plane (with integer coordinates) already requires a precision of roughly $15L$.

*Universität des Saarlandes, Fachbereich 14, Informatik, Lehrstuhl Prof. G. Hotz, Postfach 151150, D-66041 Saarbrücken, Germany. E-mail: {schoemer,sellen,welsch}@cs.uni-sb.de Fax: (049)681-3024421

1.1 Problem Formulation and Results

In our model, assembly parts are rigid bodies, represented by compact polyhedra. Each polyhedron is described by its boundary representation, and the coordinates of its vertices are given as L-bit integers. We consider the motion of one part, specified by a *trajectory*

$$\tau = (C_1, \ldots, C_k),$$

while the other parts are stationary. The trajectory consists of a sequence of intermediate configurations

$$C_i = (\mathfrak{p}_i, \mathfrak{o}_i),$$

where $\mathfrak{o}_i \in \mathbb{R}^3$ is a vector specifying the position of the polyhedron, and $\mathfrak{p}_i \in \mathbb{R}^4$ a quaternion specifying its orientation. We require that either $\mathfrak{o}_i = \mathfrak{o}_{i+1}$ or $\mathfrak{p}_i = \mathfrak{p}_{i+1}$, and assume that the coordinates of $C_i \in \mathbb{R}^7$ are L-bit integers. The motion from C_i to C_{i+1} is defined by the linear equation

$$C(t) = C_i + t(C_{i+1} - C_i), \quad t \in [0,1],$$

and corresponds to either a translation or a rotation.

Under these assumptions, we shall prove

Theorem 1 *The problem to decide whether the trajectory τ specifies a collision-free motion can be decided exactly by using integer arithmetic. The maximal bit-size of intermediate values can be bounded by $M = 14L + 22$.*

For this abstract, we did choose to reduce operations to integer arithmetic. It is important to note that in practical implementations one can also choose to use *big-floats* with precision M, and that one can often avoid to compute with full precision. (Here, big-float is assumed to be a floating-point data type with definable precision.)

Extensions to which our scheme can easily be adapted include

(a) that input numbers are rational with homogeneous coordinates (cf. [9]), and

(b) that configurations of stationary polyhedra are given by quaternions, too. (This would allow to deal with sequences of motions of different parts.)

Omitting additive factors, we computed a bound of $18L$ for case (a), and a bound of $22L$ for case (b).

1.2 Overview

In section 2, we start with preliminaries on collision detection and the quaternion calculus. In the first part, we briefly sketch the use of quaternions to treat rotational motions. In the second part, we compare our collision detection approach to Canny's classical method [2, 3], for which we computed a required precision of roughly $40L$.

In sections 3 and 4, we derive predicates to decide different types of collision. All these predicates can be decided with a precision of roughly $14L$. For simplicity in this abstract, we only deal with the 'generic' cases in these sections and omit the treatment of degeneracies.

Section 5 contains some experimental results and concluding remarks.

2 Preliminaries

2.1 Orientations Described by Quaternions

The orientation of a rigid body in 3-space can be described by a quaternion

$$\mathfrak{u} = \begin{bmatrix} u_0 \\ \mathbf{u} \end{bmatrix} \in \mathbb{R}^4.$$

This orientation results from a rotation of the world frame about the axis with direction $\mathbf{u} \in \mathbb{R}^3$. The rotation angle is determined by $|\mathbf{u}|$ and the scalar u_0.

The 0-th component of a quaternion is called the scalar part, the other components comprise the vector part. Vectors in 3-space are interpreted as quaternions with scalar part 0. Quaternions form a vector space with an associative multiplication defined by

$$\mathfrak{q} \cdot \mathfrak{p} = \begin{bmatrix} q_0 \\ \mathbf{q} \end{bmatrix} \cdot \begin{bmatrix} p_0 \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} q_0 p_0 - \mathbf{q}^T \mathbf{p} \\ q_0 \mathbf{p} + p_0 \mathbf{q} + \mathbf{q} \times \mathbf{p} \end{bmatrix}.$$

The quaternion product is linear in $\mathfrak{p}$ and $\mathfrak{q}$. The conjugate quaternion $\mathfrak{q}^*$ of $\mathfrak{q}$ is formed by negating the vector part of $\mathfrak{q}$. The product $\mathfrak{q} \cdot \mathfrak{q}^*$ yields the scalar value $q_0^2 + \mathbf{q}^2$, which corresponds to the length of $\mathfrak{q}$ under the Euclidean metric in $\mathbb{R}^4$. Quaternions which satisfy $\mathfrak{q} \cdot \mathfrak{q}^* = 1$ are called unit quaternions (in sections 3 and 4 we assume w.l.o.g. that the scalar parts of quaternions are positive). Let $\mathfrak{u}$ be a given (integer) quaternion. Then the mapping

$$\mathfrak{a} = \begin{bmatrix} 0 \\ \mathbf{a} \end{bmatrix} \mapsto \mathfrak{a}' = \begin{bmatrix} 0 \\ \mathbf{a}' \end{bmatrix} = \frac{\mathfrak{u} \cdot \mathfrak{a} \cdot \mathfrak{u}^*}{\mathfrak{u} \cdot \mathfrak{u}^*}$$

describes a rotation of the vector $\mathbf{a}$ about the axis $\mathbf{u}$ about the angle $\varphi = 2 \arctan(|\mathbf{u}|/u_0)$. In matrix notation, this amounts to $\mathbf{a}' = \mathbf{U} \cdot \mathbf{a}$, with

$$\mathbf{U} = \frac{(u_0^2 - \mathbf{u}^2)\mathbf{I} + 2\mathbf{u}\mathbf{u}^T + 2u_0\mathbf{u}^\times}{u_0^2 + \mathbf{u}^2}. \qquad (1)$$

Here, $\mathbf{u}^\times$ denotes the canonical skew-symmetric matrix corresponding to $\mathbf{u}$.

Now let us have a closer look at the motions defined by the *linear transition* between two configurations C_i and C_{i+1}, as defined in the introduction.

First consider the translation determined by $C_i = (\mathfrak{p}, \mathfrak{o}_i)$ and $C_{i+1} = (\mathfrak{p}, \mathfrak{o}_{i+1})$. Let $\mathbf{P}$ be the rotation matrix corresponding to $\mathfrak{p}$, and $\mathbf{s} = \mathfrak{o}_{i+1} - \mathfrak{o}_i$. Then the position of a vertex $\mathbf{v}$ of the moving polyhedron at time $t \in [0,1]$ is given by $\mathbf{v}(t) = \mathbf{P}\mathbf{v} + \mathfrak{o}_i + t\mathbf{s}$.

Now consider the linear transition between $C_i = (\mathfrak{p}, \mathfrak{o})$ and $C_{i+1} = (\mathfrak{q}, \mathfrak{o})$, corresponding to some rotation. Let

$$\mathfrak{u}(t) = \mathfrak{p} + t(\mathfrak{q} - \mathfrak{p}),\ t \in [0,1],$$

and $\mathbf{U}(t)$ the adequate rotation matrix. Then the locus of a vertex $\mathbf{v}$ of the moving polyhedron at time t is $\mathbf{v}(t) = \mathbf{U}(t) \cdot \mathbf{v} + \mathfrak{o}$. The described motion is a rotation around the line with center $\mathfrak{o}$ and direction vector $\mathbf{r} = p_0\mathbf{q} - q_0\mathbf{p} + \mathbf{p} \times \mathbf{q}$, which is identical to the vector part of the quaternion product $\mathfrak{q} \cdot \mathfrak{p}^*$.

Instead of applying the translation $\mathfrak{o}_i$ to the moving polyhedron, we can also apply the inverse translation $-\mathfrak{o}_i$ to the obstacles. Proceeding in this way leads to simpler formulas in sections 3 and 4. There, we shall assume that $\mathfrak{o}_i = 0$, and – to capture the inverse translation of obstacles – that obstacle coordinates are $(L+1)$-bit numbers.

2.2 Collision Between Polyhedra

There are several ways to characterize collisions. Canny reduces the task of detecting a collision between two polyhedra to the test whether an edge of one polyhedron pierces a face of the other. Roughly speaking, faces are considered as intersections of half-spaces, and the test corresponds to intersecting time intervals during which edges/halfplanes overlap. This leads to the need to compare different collision times.

The disadvantage of this method becomes apparent in the case of rotational motions. In this case the collision times are roots of quadratic equations with integer coefficients of size $5L$. Thus, we need to compare two expressions of the form

$$\frac{\alpha_1 + \sqrt{\beta_1}}{\gamma_1} : \frac{\alpha_2 + \sqrt{\beta_2}}{\gamma_2}.$$

This is a standard primitive which arises in many algorithms, e.g., Fortune's sweep algorithm for Voronoi diagrams. Using the known techniques to separate two such values, we obtained a needed precision of roughly $40L$. Though it is not clear whether this gap is sharp, it is an artifact of the method (for comparison, note that Voronoi diagrams can be computed incrementally without using the disastrous comparisons that are inherent in Fortune's algorithm).

Our improvement is based on *avoiding* the comparison between two collision times. To achieve this, we need to follow a different strategy to characterize collisions: Two polyhedra are in contact if an edge (vertex) of one polyhedron touches an edge (face) of the other. We proceed in two steps. In the first step we expand edges to lines and faces to planes and examine the existence of the resulting collision times. In the second step we decide whether the corresponding edges (respectively, a vertex and a face) really collide at these potential collision times. The main point is that the latter test can be decided by predicates which are *independent* of the actual collision times. (We derive these predicates by exploiting the invariants of the motion in a clever way.)

3 Collision Between two Edges

In this section, we shall decide whether a moving edge $l_{ab}(t)$ collides with a stationary edge l_{cd}. The basic step in 3.1 and 3.2 is the calculation of the moment of collision between the corresponding infinite lines $L_{ab}(t)$ and L_{cd}. $L_{ab}(t) \cap L_{cd} \neq \emptyset$ implies that the endpoints $\mathbf{a}(t)$, $\mathbf{b}(t)$ and $\mathbf{c}$, $\mathbf{d}$ lie in a common plane. This can be expressed by

$$\det \begin{bmatrix} 1 & 1 & 1 & 1 \\ \mathbf{a}(t) & \mathbf{b}(t) & \mathbf{c} & \mathbf{d} \end{bmatrix} = 0. \qquad (2)$$

3.1 Translation

Let the moving polyhedron be in orientation $\mathfrak{p}$ and perform a translation in direction $\mathbf{s}$. Then the vertices move according to $\mathbf{a}(t) = \mathbf{P}\mathbf{a} + t\,\mathbf{s}$ and $\mathbf{b}(t) = \mathbf{P}\mathbf{b} + t\,\mathbf{s}$.

We first compute the time t_0 when the line $L_{ab}(t)$ collides with the line L_{cd}. Expansion of the determinant in equation (2) yields a linear equation of the form

$$\alpha t + \beta = 0, \text{ with} \qquad (3)$$
$$\alpha = \mathbf{s}^T(\mathbf{P}(\mathbf{b}-\mathbf{a}) \times (\mathbf{d}-\mathbf{c})),$$
$$\beta = (\mathbf{c} \times \mathbf{d})^T\mathbf{P}(\mathbf{b}-\mathbf{a}) + (\mathbf{d}-\mathbf{c})^T\mathbf{P}(\mathbf{a} \times \mathbf{b}).$$

In order to determine α and β exactly one needs integers of length $5L + 9$. If a collision takes place during the translation, then the solution t_0 of equation (3) must lie in the interval $[0,1]$. Omitting the cases $t_0 = 0$ and $t_0 = 1$, this can be decided as follows:

$$[0 < t_0 < 1] \iff [\alpha > -\beta > 0] \vee [\alpha < -\beta < 0].$$

If the lines $L_{ab}(t)$ and L_{cd} collide at time $t_0 \in [0,1]$, we additionally need to check whether the corresponding edges intersect at time t_0. For this test, we consider the plane H_{ab} (H_{cd}) which is spanned by L_{ab} (L_{cd}) and the translation direction $\mathbf{s}$. Then the collision of lines is a collision of edges iff a and b lie on different sides of H_{cd} and c and d lie on different sides of H_{ab}. This can be decided by the following four predicates:

$$(\mathbf{s} \times \mathbf{P}(\mathbf{b}-\mathbf{a}))^T(\mathbf{x}-\mathbf{P}\mathbf{a}) \lessgtr 0, \text{ with } \mathbf{x} \in \{\mathbf{c},\mathbf{d}\},$$
$$(\mathbf{s} \times (\mathbf{d}-\mathbf{c}))^T(\mathbf{x}-\mathbf{c}) \lessgtr 0, \text{ with } \mathbf{x} \in \{\mathbf{P}\mathbf{a},\mathbf{P}\mathbf{b}\}.$$

In both cases, integers of length $5L+8$ suffice to calculate the predicates exactly. (Note that these predicates are independent from t_0. We shall proceed in a similar way in the case of rotations.)

3.2 Rotation

Let the rotation of each vertex $\mathbf{v}$ of the moving polyhedron be described by $\mathbf{v}(t) = \mathbf{U}(t) \cdot \mathbf{v}$. Again, we first compute the collision times for the rotating line $L_{ab}(t)$ and the stationary line L_{cd}. Equation (2) yields

$$(\mathbf{c} \times \mathbf{d})^T\mathbf{U}(t)(\mathbf{b}-\mathbf{a}) + (\mathbf{d}-\mathbf{c})^T\mathbf{U}(t)(\mathbf{a} \times \mathbf{b}) = 0.$$

Substituting $\mathbf{U}(t)$ according to equation (1), this is equivalent to the quadratic equation

$$\alpha_1 t^2 + \beta_1 t + \gamma_1 = 0, \text{ with} \qquad (4)$$

$$\begin{aligned}
\alpha_1 &= ((q_0-p_0)^2 - (\mathbf{q}-\mathbf{p})^2)\omega + 2(q_0-p_0)(\mathbf{q}-\mathbf{p})^T\mathbf{w} \\
&+ 2(\mathbf{q}-\mathbf{p})^T(\mathbf{c} \times \mathbf{d})(\mathbf{q}-\mathbf{p})^T(\mathbf{b}-\mathbf{a}) \\
&+ 2(\mathbf{q}-\mathbf{p})^T(\mathbf{d}-\mathbf{c})(\mathbf{q}-\mathbf{p})^T(\mathbf{a} \times \mathbf{b}), \\
\beta_1 &= 2(p_0(q_0-p_0) - \mathbf{p}^T(\mathbf{q}-\mathbf{p}))\omega \\
&+ 2(p_0(\mathbf{q}-\mathbf{p})^T + (q_0-p_0)\mathbf{p}^T)\mathbf{w} \\
&+ 2\mathbf{p}^T(\mathbf{b}-\mathbf{a})(\mathbf{q}-\mathbf{p})^T(\mathbf{c} \times \mathbf{d}) \\
&+ 2\mathbf{p}^T(\mathbf{d}-\mathbf{c})(\mathbf{q}-\mathbf{p})^T(\mathbf{a} \times \mathbf{b}) \\
&+ 2\mathbf{p}^T(\mathbf{a} \times \mathbf{b})(\mathbf{q}-\mathbf{p})^T(\mathbf{d}-\mathbf{c}) \\
&+ 2\mathbf{p}^T(\mathbf{c} \times \mathbf{d})(\mathbf{q}-\mathbf{p})^T(\mathbf{b}-\mathbf{a}), \\
\gamma_1 &= (p_0^2 - \mathbf{p}^2)\omega + 2p_0\mathbf{p}^T\mathbf{w} \\
&+ 2\mathbf{p}^T(\mathbf{c} \times \mathbf{d})\mathbf{p}^T(\mathbf{b}-\mathbf{a}) + 2\mathbf{p}^T(\mathbf{d}-\mathbf{c})\mathbf{p}^T(\mathbf{a} \times \mathbf{b}).
\end{aligned}$$

Here, $\omega = (\mathbf{c} \times \mathbf{d})^T(\mathbf{b}-\mathbf{a}) + (\mathbf{a} \times \mathbf{b})^T(\mathbf{d}-\mathbf{c})$, and $\mathbf{w} = (\mathbf{b}-\mathbf{a}) \times (\mathbf{c} \times \mathbf{d}) - (\mathbf{d}-\mathbf{c}) \times (\mathbf{a} \times \mathbf{b})$. The coefficients of this quadratic equation have length at most $5L+12$.

If a collision between the lines $L_{ab}(t)$ and L_{cd} occurs ($\beta_1^2 - 4\alpha_1\gamma_1 \geq 0$), we subsequently investigate whether the rotating edge $l_{ab}(t)$ collides with the edge l_{cd} during a full rotation. (We have to consider that the start orientation of $l_{ab}(t)$ is given by the quaternion $\mathfrak{p}$. To indicate that both endpoints are in this orientation, we use the notation $\mathbf{a}' = \mathbf{P}\mathbf{a}$ and $\mathbf{b}' = \mathbf{P}\mathbf{b}$.)

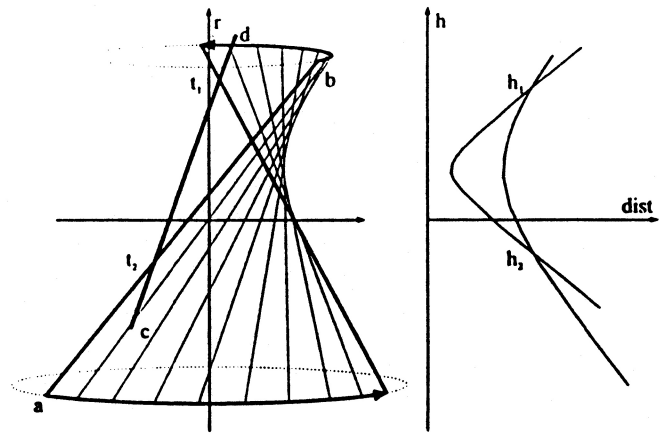

Figure 1: Collision of two edges during a rotation

For that purpose, we calculate the *height* of the collision point with respect to the rotation axis $\mathbf{r}$. Let

$$\mathbf{s}_{a'b'}(h) = H(h) \cap l_{a'b'}, \quad \mathbf{s}_{cd}(h) = H(h) \cap l_{cd}.$$

be the intersection points of the two edges with the plane $H(h) : \mathbf{r}^T\mathbf{x} = h$. W.l.o.g. we assume

$$\mathbf{r}^T\mathbf{a}' < \mathbf{r}^T\mathbf{b}' \quad \text{and} \quad \mathbf{r}^T\mathbf{c} < \mathbf{r}^T\mathbf{d}. \qquad (5)$$

If $l_{a'b'}$ collides with l_{cd} during the rotation, then there exists some

$$h \in [\max\{\mathbf{r}^T\mathbf{a}', \mathbf{r}^T\mathbf{c}\}, \min\{\mathbf{r}^T\mathbf{b}', \mathbf{r}^T\mathbf{d}\}], \text{ such that}$$
$$\text{dist}(\mathbf{s}_{a'b'}(h), L_{0r}) = \text{dist}(\mathbf{s}_{cd}(h), L_{0r}). \qquad (6)$$

This is equivalent to the question whether two hyperbolic segments intersect. As illustrated in figure 1 these segments result from a representation of $l_{a'b'}$ and l_{cd} in cylindrical coordinates. With

$$\mathbf{s}_{a'b'}(h) = \frac{\mathbf{r} \times (\mathbf{a}' \times \mathbf{b}') + h(\mathbf{b}'-\mathbf{a}')}{\mathbf{r}^T(\mathbf{b}'-\mathbf{a}')},$$

condition (6) can be rewritten as a quadratic equation

$$\alpha_2 h^2 + \beta_2 h + \gamma_2 = 0, \text{ with}$$

$$\begin{aligned}
\alpha_2 &= (\mathbf{d}-\mathbf{c})^2(\mathbf{r}'^T(\mathbf{b}-\mathbf{a}))^2 - (\mathbf{b}-\mathbf{a})^2(\mathbf{r}^T(\mathbf{d}-\mathbf{c}))^2, \\
\beta_2 &= 2\mathbf{r}^T((\mathbf{c} \times \mathbf{d}) \times (\mathbf{d}-\mathbf{c}))(\mathbf{r}'^T(\mathbf{b}-\mathbf{a}))^2 \\
&- 2\mathbf{r}'^T((\mathbf{a} \times \mathbf{b}) \times (\mathbf{b}-\mathbf{a}))(\mathbf{r}^T(\mathbf{d}-\mathbf{c}))^2, \\
\gamma_2 &= (\mathbf{r} \times (\mathbf{c} \times \mathbf{d}))^2(\mathbf{r}'^T(\mathbf{b}-\mathbf{a}))^2 \\
&- (\mathbf{r}' \times (\mathbf{a} \times \mathbf{b}))^2(\mathbf{r}^T(\mathbf{d}-\mathbf{c}))^2.
\end{aligned}$$

Here, $\mathbf{r}' = \mathbf{P}^T\mathbf{r} = p_0\mathbf{q} - q_0\mathbf{p} - \mathbf{p} \times \mathbf{q}$. The coefficients α_2, β_2 and γ_2 of this equation have length $\leq 8L+13$, $\leq 11L+19$ and $\leq 14L+22$, respectively.

Till now, we have derived two quadratic equations that define collision times and collision heights. Let

$$t^{\pm} = \frac{-\beta_1 \pm \sqrt{\beta_1^2 - 4\alpha_1\gamma_1}}{2\alpha_1} \quad \text{and}$$
$$h^{\pm} = \frac{-\beta_2 \pm \sqrt{\beta_2^2 - 4\alpha_2\gamma_2}}{2\alpha_2}.$$

These values are related to each other: assuming (5), the moment of collision t^+ (t^-) corresponds to the value h^+ (h^-). Hence a collision occurs iff

$$\begin{aligned}
&[0 < t^+ < 1] \\
\wedge\ &[\max\{\mathbf{r}'^T\mathbf{a}, \mathbf{r}^T\mathbf{c}\} < h^+ < \min\{\mathbf{r}'^T\mathbf{b}, \mathbf{r}^T\mathbf{d}\}] \\
\vee\ &[0 < t^- < 1] \\
\wedge\ &[\max\{\mathbf{r}'^T\mathbf{a}, \mathbf{r}^T\mathbf{c}\} < h^- < \min\{\mathbf{r}'^T\mathbf{b}, \mathbf{r}^T\mathbf{d}\}].
\end{aligned}$$

It remains to test whether the real roots $x^{\pm}$ of a quadratic equation lie in a given interval. This amounts to deciding predicates of the form $[x^{\pm} \lesseqgtr X]$. Under the precondition that real roots exist, it holds for example:

$$\begin{aligned}
&[x^+ < X] \iff \\
&[\alpha > 0] \wedge [2X\alpha + \beta > 0] \wedge [X^2\alpha + X\beta + \gamma > 0] \\
\vee\ &[\alpha < 0] \wedge ([2X\alpha + \beta < 0] \vee [X^2\alpha + X\beta + \gamma > 0]) \\
\vee\ &[\alpha = 0] \wedge \quad [\beta > 0] \wedge \quad [X\beta + \gamma > 0]\ .
\end{aligned}$$

All atomic predicates can be evaluated with integers of length $\leq 14L + 22$.

4 Collision Between a Vertex and a Face

We proceed similar to section 3 to check whether a moving point $\mathbf{a}(t)$ collides with an obstacle face. W.l.o.g. we assume that this face is a triangle Δ_{bcd}. Again, the basic step is to calculate the moment of collision between $\mathbf{a}(t)$ and the infinite plane

$$H_n : \mathbf{n}^T\mathbf{x} = n_0,$$

with $\mathbf{n} = \mathbf{b} \times \mathbf{c} + \mathbf{c} \times \mathbf{d} + \mathbf{d} \times \mathbf{b}$ and $n_0 = \mathbf{b}^T(\mathbf{c} \times \mathbf{d})$.

4.1 Translation

Let $\mathbf{a}(t) = \mathbf{Pa} + t\mathbf{s}$. Substituting this into the plane equation H_n yields a linear equation

$$\alpha t + \beta = 0.$$

Analogous to 3.1, we can decide if this linear equation has a solution $t_0 \in [0, 1]$ by evaluating simple predicates involving α and β. The bit-length of α and β can be bounded by $5L + 8$.

In a second step, we check whether the point a collides with the face Δ_{bcd} during an 'infinite' translation. Let H_{bc} (H_{cd}, H_{db}) be the plane spanned by s and the direction of edge l_{bc} (l_{cd}, l_{db}). We reduce the test above to deciding the relative position of $\mathbf{a}' = \mathbf{Pa}$ with respect to these planes. A collision takes place iff

$$\begin{aligned}
&(\mathbf{s} \times (\mathbf{c} - \mathbf{b}))^T\ (\mathbf{Pa} - \mathbf{c}) \leq 0 \\
\wedge\ &(\mathbf{s} \times (\mathbf{d} - \mathbf{c}))^T\ (\mathbf{Pa} - \mathbf{d}) \leq 0 \\
\wedge\ &(\mathbf{s} \times (\mathbf{b} - \mathbf{d}))^T\ (\mathbf{Pa} - \mathbf{b}) \leq 0.
\end{aligned}$$

This can be decided by computing integers of length $\leq 5L + 8$.

4.2 Rotation

Let the vertex a move according to $\mathbf{a}(t) = \mathbf{U}(t) \cdot \mathbf{a}$, where $\mathbf{U}(t)$ describes the transition from orientation $\mathfrak{p}$ to orientation $\mathfrak{q}$. By substituting this into the plane equation H_n, we obtain a quadratic equation in t that defines the collision times:

$$\begin{aligned}
&\alpha_3 t^2 + \beta_3 t + \gamma_3 = 0, \text{ with} \\
\alpha_3 &= \mathbf{n}^T\mathbf{a}\left((q_0 - p_0)^2 - (\mathbf{q} - \mathbf{p})^2\right) \\
&+ 2(\mathbf{q} - \mathbf{p})^T\mathbf{n}(\mathbf{q} - \mathbf{p})^T\mathbf{a} \\
&+ 2(q_0 - p_0)(\mathbf{q} - \mathbf{p})^T(\mathbf{a} \times \mathbf{n}) \\
&- n_0\left((q_0 - p_0)^2 + (\mathbf{q} - \mathbf{p})^2\right), \\
\beta_3 &= 2\mathbf{n}^T\mathbf{a}\left(p_0(q_0 - p_0) - \mathbf{p}^T(\mathbf{q} - \mathbf{p})\right) \\
&+ 2\mathbf{n}^T\mathbf{p}\mathbf{a}^T(\mathbf{q} - \mathbf{p}) + 2\mathbf{a}^T\mathbf{p}\mathbf{n}^T(\mathbf{q} - \mathbf{p}) \\
&+ 2\left(p_0(\mathbf{q} - \mathbf{p})^T + (q_0 - p_0)\mathbf{p}^T\right)(\mathbf{a} \times \mathbf{n}) \\
&- 2n_0\left(p_0(q_0 - p_0) + \mathbf{p}^T(\mathbf{q} - \mathbf{p})\right), \\
\gamma_3 &= \mathbf{n}^T\mathbf{a}(p_0^2 - \mathbf{p}^2) + 2\mathbf{n}^T\mathbf{p}\mathbf{a}^T\mathbf{p} \\
&+ 2p_0\mathbf{p}^T(\mathbf{a} \times \mathbf{n}) - n_0(p_0^2 + \mathbf{p}^2).
\end{aligned}$$

This equation has coefficients of length $\leq 5L + 11$. The roots are

$$t^{\pm} = \frac{-\beta_3 \pm \sqrt{\beta_3^2 - 4\alpha_3\gamma_3}}{2\alpha_3}.$$

One can verify that at time t^+ $\mathbf{a}(t)$ crosses from the back side of the plane H_n to the front side and at time t^- back again. This implies that

$$\begin{aligned}
&\mathbf{a}(t^+) \in H^+_{n \times r}\ ,\ \mathbf{a}(t^-) \in H^-_{n \times r}\ , \\
&\text{with}\quad H^{\pm}_{n \times r} : (\mathbf{n} \times \mathbf{r})^T\mathbf{x} \gtrless 0.
\end{aligned}$$

A collision of $\mathbf{a}(t)$ with the front side of H_n is only possible if $t^- \in [0, 1]$ (collisions with the back side may be ignored). This requires that the equation $\alpha_3 t^2 + \beta_3 t + \gamma_3 = 0$ has real roots at all, i.e., $\beta_3^2 - 4\alpha_3\gamma_3 \geq 0$. The check of this condition can be performed with integers of length

$\leq 10L + 22$, and for the decision whether $t^- \in [0,1]$ it suffices to compute with integers of length $5L + 12$.

In the sequel we study the question under which condition the point $\mathbf{a}$ not only hits the plane H_n but also the triangle Δ_{bcd} during its rotation about the axis $\mathbf{r} = p_0\mathbf{q} - q_0\mathbf{p} + \mathbf{p} \times \mathbf{q}$. For that purpose we intersect Δ_{bcd} with the plane $H_r : \mathbf{r}^T(\mathbf{x} - \mathbf{a}') = 0$. In general this yields a line segment l. Let $\mathbf{s}_1$ and $\mathbf{s}_2$ be its endpoints. We distinguish three cases in which a collision of $\mathbf{a}' = \mathbf{Pa}$ with the front side of Δ_{bcd} occurs during a full rotation:

(1) $\mathbf{s}_1, \mathbf{s}_2 \in H^-_{n \times r}$:
$\mathbf{a}'$ collides with l if
$\min\{\mathbf{s}_1^2, \mathbf{s}_2^2\} \leq \mathbf{a}'^2 \leq \max\{\mathbf{s}_1^2, \mathbf{s}_2^2\}$.

(2) $\mathbf{s}_1 \in H^-_{n \times r}$, $\mathbf{s}_2 \in H^+_{n \times r}$:
$\mathbf{a}'$ collides with l if $\mathbf{a}'^2 \leq \mathbf{s}_1^2$.

(3) $\mathbf{s}_1 \in H^+_{n \times r}$, $\mathbf{s}_2 \in H^-_{n \times r}$:
$\mathbf{a}'$ collides with l if $\mathbf{a}'^2 \leq \mathbf{s}_2^2$.

For the calculation of $\mathbf{s}_1$ and $\mathbf{s}_2$ we determine the intersection of the line segments l_{bc}, l_{cd} and l_{db} with H_r. The intersection point $\mathbf{s}$ of the line L_{bc} with H_r is

$$\mathbf{s} = \mathbf{b} + \lambda_s(\mathbf{c} - \mathbf{b}), \text{ with } \lambda_s = \frac{\mathbf{r}^T(\mathbf{a}' - \mathbf{b})}{\mathbf{r}^T(\mathbf{c} - \mathbf{b})}.$$

The point $\mathbf{s}$ only lies on l_{bc} if $0 \leq \lambda_s \leq 1$, i.e., if

$$\min\{\mathbf{r}^T\mathbf{b}, \mathbf{r}^T\mathbf{c}\} \leq \mathbf{r}^T\mathbf{a}' \leq \max\{\mathbf{r}^T\mathbf{b}, \mathbf{r}^T\mathbf{c}\}. \qquad (7)$$

In order to solve this inequality it is sufficient to use integers of length $\leq 3L + 5$. If (7) is fulfilled, then $\mathbf{s}$ is an endpoint of $l = H_r \cap \partial\Delta_{bcd}$. To decide the three cases above, we need to compare the (squared) distance of $\mathbf{s}$ to the axis of rotation with that of $\mathbf{a}'$. In addition, the condition $\mathbf{s} \in H^{\pm}_{n \times r}$ must be decided. All involved inequalities can be expressed as comparisons between integers of length $\leq 8L + 15$.

5 Conclusion

Engineering applications such as assembly planning constitute a challenging target for the exact computation paradigm. The price we have to pay for an exact algorithm crucially depends on the *average* precision that is needed to carry out decisions exactly. In this paper, we have presented an exact collision detection scheme with a required *worst case* precision that beats comparable approaches, and that is well in the reach of existing big-number packages.

True implementation issues go beyond the scope of this abstract. Here, conceptual decisions like the representation of numbers influence the final performance. Note that there are recent efforts [5, 4] to provide big-number platforms with a wide range of functionality, e.g., a type *real* in LEDA [5] with implicit error parameters and automatic re-evaluation if decisions require higher than the present accuracy.

To obtain some preliminary estimates on what we can expect from the exact approach, we implemented a sign test of the discriminant of equation (4) by using the LEDA data types *double*, *integer* and *real*. For 10000 randomly chosen input values of bit-length 30, the fixed-precision *double* implementation needed 0.8 sec. for all tests, but failed several times to give the correct answer. The exact *(big-)integer* implementation needed 14.2 sec. and the *real*-implementation 13.4 sec. for the same values. Taking into account that real input yields more critical sign tests than random input, the integer data type seems favorable. After all, the price to pay for an exact algorithm seems acceptable for specific applications. The full algorithm is currently under implementation, and we expect more thorough empirical results in the future.

References

[1] C. Burnikel, K. Mehlhorn, S. Schirra: *How to compute the Voronoi diagram of line segments: theoretical and experimental results*, Proc. ESA 94, LNCS Vol. 855, 1994, pp. 227-239.

[2] J. Canny: *On detecting collision between polyhedra*, Proc. ECAI, 1984, pp. 533-542.

[3] J. Canny: *The Complexity of Robot Motion Planning*, ACM Doctoral Dissertation, MIT Press.

[4] T. Dubé, C.-K. Yap: *A Basis for Implementing Exact Geometric Algorithms*, Manuscript.

[5] S. Näher: *The LEDA User Manual, Version 3.1*, Max-Planck-Institut für Informatik, Saarbrücken, 1995.

[6] K. Mehlhorn, S. Näher: *The Implementation of Geometric Algorithms*, 13th World Computer Congress IFIP94, Vol. 1, 1994, pp. 223-231.

[7] E. Schömer: *Interaktive Montageplanung mit Kollisionserkennung*, PhD Thesis, Saarbrücken, 1994.

[8] E. Schömer, C. Thiel: *Efficient collision detection for moving polyhedra*, Tech. Report 94-147, Max-Planck-Institut für Informatik, Saarbrücken, 1994 (to appear: 11th ACM Symp. on Comp. Geom.).

[9] C.-K. Yap: *Towards exact geometric computation*, Proc. 5th Canadian Conf. on Comp. Geom., 1993, pp. 405-419 (to appear: Computational Geometry Theory and Applications).