An Algorithm for Computing the Exact Configuration Space of a Rotating Object in 3-space

Przemysław Dobrowolski

***Abstract*—In this paper we present and test an algorithm for constructing SO(3) configuration spaces. It is capable of handling polyhedral scenes (triangulated) as well as ball-only scenes (sphere-trees). Without loss of generality we consider objects rotating around the zero point. In a scene, consisting of n triangular faces (or balls) of obstacles and m triangular faces (or balls) of a rotating object, the complexity of the presented algorithm is $O(n^3m^3log(nm))$. The algorithm is output-sensitive, which means that it discards all unnecessary geometry and takes only a minimum number of geometry needed to obtain a correct final configuration space. Configuration spaces are represented as graphs of intersections on the border of free configuration space. This algorithm is a generalization of a few previous related works: the case of a rotating and translating polygon on a plane and the case of a rotating and translating segment (or a cigar-like object) in 3-space.**

***Index Terms*—motion planning, exact algorithm, rotations in 3-space, rotating polyhedron, rotating sphere-tree**

I. Introduction

MOTION planning is currently a common task in robotics. It has been reconsidered since the 80s. There are two categories of algorithms: approximate and exact. In this paper we consider the exact category. Exact (or combinatorial) algorithms are usually slower and more involved, but always give a correct answer. For a survey on motion planning algorithms, including combinatorial ones, the reader can refer to [1] or [2].

In this paper, the author presents a research on those cases of determining a free space of motion planning problem, where 3-dimensional rotations are involved. The simplest such case is a motion planning in a purely rotational space $\mathrm{SO}(3)$. An introduction to the research of this case is also presented in the paper [3].

A near-optimal algorithm was developed. Given an arbitrary rotating polyhedron in a polyhedral scene or a ball approximated object in a ball-scene, it determines the exact configuration space of the rotating object. An important feature of the proposed algorithm is its output-sensitivity. It means that it is not possible to reduce the number of considered geometry predicates because it is already minimal. In practical scenes, even if the asymptotic complexity of the algorithm is $O(n^3m^3log(nm))$, the algorithm works in acceptable times. Having computed a configuration space, one can perform motion planning for arbitrary begin and end rotations. There exist a few exact motion planners with rotations. In the case of a planar movements, these include: needle movement [4], convex polygon movement [5] and arbitrary polygon movement [6]. In 3-space an algorithm for planning movement of a needle and a cigar-like object was proposed by Koltun [7].

The paper is organised as follows. In the first section we introduce basic definitions and properties. On top of that, we give a concept of a collision predicate, which is a basic tool for combinatorial motion planning. In the next section, collision predicates are used to define collision surfaces in a configuration space. The crucial part of the proposed algorithm is the intersection algorithm. It creates an arrangement of all surfaces. With the precomputed arrangement, one can easily execute motion planning queries. This is discussed in the last section, along with some performance results.

A. Problem description and motivation

The new algorithm is expected to be useful in some areas. Firstly, it increases the number of different topologies in which we can do exact motion planning. Secondly, it can be used to create a hybrid motion planning algorithms of new class. Hybrid algorithms, mix different motion planning algorithms in order to achieve better practical performance and output quality. There exist hybrid algorithms like [8], [9], [10], [11] or [12], but none of them uses exact description of rotational part. All referred hybrid algorithms use some kind of rotational space approximation, like *slicing* in [12] or ACD tree in [9], which is only *resolution-complete*. In [10] different PRM methods are mixed, while in [11] authors combine a PRM planner and a ACD tree together. The algorithm, proposed in [8] differers from all above. It is based on a Voronoi roadmap and utilizes a concept of a *bridge*. Author's own research on complete motion planning showed that it would be possible to create a rigid body planner in $\mathbb{R}^3$, assuming that there exist an method of creating an exact description of $\mathrm{SO}(3)$ configuration space.

B. Three dimensional space of rotations

The space of rotations $\mathrm{SO}(3)$ of a three dimensional Cartesian space is three dimensional, but not homeomorphic to $\mathbb{R}^3$. In fact, it is homeomorphic to a 3-sphere where each pair of antipodal points are identified. Due to non-Cartesian nature of the space of rotations, algorithms are usually more involved that those operating in a Cartesian space.

C. $SO(3)$ configuration space

During the research, a new result was obtained - a complete description of the configuration space of rotations of

Manuscript received October 1, 2012.

P. Dobrowolski is with the Faculty of Mathematics and Information Science, Warsaw University of Technology, Warsaw, Poland, e-mail: dobrowolskip@mini.pw.edu.pl

a polyhedron in a polyhedral scene. We will now introduce definitions that are used in the new algorithm.

In case of the SO(3), a *configuration* (placement, orientation) is a rotation. *Configuration space* is the set of all configurations. By convention, we mark it with $\mathcal{C}$. A subset $\mathcal{C}_{forbidden}$ of $\mathcal{C}$ that cause the rotating polyhedron to collide with any of the obstacles, is called a *forbidden* subset of rotations. A complementary subset $\mathcal{C}_{allowed}$ of rotations is called an *allowed* (free) subset of rotations.

A configuration in a configuration space is represented by a *spinor* $s \in \text{Spin}(3)$. A spinor representation of rotations is quite new in computational geometry, but in mechanics it has already been used for a few decades. A spinor is an element of the geometric algebra. It has already been proven practically that the geometric algebra can be quite useful [13]. This is because of its generality and great insight in all operations. For an introduction into the geometric algebra one can refer to [14] or [13]. A spinor is a number of the form: $s = s_0 + s_{12}\mathbf{e}_{12} + s_{23}\mathbf{e}_{23} + s_{31}\mathbf{e}_{31}$. The numbers $s_0, s_{12}, s_{23}, s_{31} \in \mathbb{R}$ are called *coefficients* and satisfy the identity:

$$s_0^2 + s_{12}^2 + s_{23}^2 + s_{31}^2 = 1$$

The three base elements: $\mathbf{e}_{12}, \mathbf{e}_{23}, \mathbf{e}_{31}$ satisfy a number of identities, similar to quaternion algebra:

$$\mathbf{e}_{12}\mathbf{e}_{12} = \mathbf{e}_{23}\mathbf{e}_{23} = \mathbf{e}_{31}\mathbf{e}_{31} = -1$$

$$\mathbf{e}_{12}\mathbf{e}_{23} = -\mathbf{e}_{31},\ \mathbf{e}_{23}\mathbf{e}_{31} = -\mathbf{e}_{12},\ \mathbf{e}_{31}\mathbf{e}_{12} = -\mathbf{e}_{23}$$

$$\mathbf{e}_{12} = -\mathbf{e}_{21},\ \mathbf{e}_{23} = -\mathbf{e}_{32},\ \mathbf{e}_{31} = -\mathbf{e}_{13}$$

$$\mathbf{e}_{12}\mathbf{e}_{23}\mathbf{e}_{31} = 1$$

In the paper we use the following notation:

- scalars: a, b, c, ...
- vectors: U, V, A, B, K, L, ...
- spinor: s

A vector is a number of the form: $V = V_x\mathbf{e}_1 + V_y\mathbf{e}_2 + V_z\mathbf{e}_3$. The scalar product is denoted by $\cdot$ symbol. In geometric algebra we have: $U \cdot V = \frac{1}{2}(UV + VU)$ for vectors U and V. The exterior product is related to the cross product of two vectors by the identity: $U \wedge V = \mathbf{e}_{123} U \times V$. Finally, the exterior product of two vectors is defined as: $U \wedge V = \frac{1}{2}(UV - VU)$.

Basically, it can be assumed that spinors are closely related to unit quaternions via a simple isomorphism:

$$\mathbf{i} = -\mathbf{e}_{23},\ \mathbf{j} = -\mathbf{e}_{31},\ \mathbf{k} = -\mathbf{e}_{12}$$

In geometric algebra, it is a Clifford multiplication that rotates a vector. Let s be a spinor, and $V = V_x\mathbf{e}_1 + V_y\mathbf{e}_2 + V_z\mathbf{e}_3$ vector being rotated. The explicit rotation formula is:

$$\begin{aligned}
&\mathscr{R}_s(V) = s^{-1}Vs \\
=&(s_0 - s_{12}\mathbf{e}_{12} - s_{23}\mathbf{e}_{23} - s_{31}\mathbf{e}_{31}) \\
&(V_x\mathbf{e}_1 + V_y\mathbf{e}_2 + V_z\mathbf{e}_3) \\
&(s_0 + s_{12}\mathbf{e}_{12} + s_{23}\mathbf{e}_{23} + s_{31}\mathbf{e}_{31}) \\
=&((s_0s_0 - s_{12}s_{12} + s_{23}s_{23} - s_{31}s_{31})V_x + \\
&2(V_y(s_0s_{12} + s_{23}s_{31}) + V_z(s_{12}s_{23} - s_0s_{31})))\mathbf{e}_1 + \\
&((s_0s_0 - s_{12}s_{12} - s_{23}s_{23} + s_{31}s_{31})V_y + \\
&2(V_x(s_{23}s_{31} - s_0s_{12}) + V_z(s_{12}s_{31} + s_0s_{23})))\mathbf{e}_2 + \\
&((s_0s_0 + s_{12}s_{12} - s_{23}s_{23} - s_{31}s_{31})V_z + \\
&2(V_x(s_{12}s_{23} + s_0s_{31}) + V_y(s_{12}s_{31} - s_0s_{23})))\mathbf{e}_3
\end{aligned}$$

With a spinor representation one gets a common toolbox for handling rotations on a plane and in the space. This is because spinors are defined for an arbitrary dimension and obey the same transformation rules.

There are few identities for the rotation formula, which are used frequently. For any pair of vectors A and B and scalars α and β, we have:

$$\begin{aligned}
\mathscr{R}_s(\alpha A + \beta B) &= \alpha\mathscr{R}_s(A) + \beta\mathscr{R}_s(B) \\
\mathscr{R}_s(A \times B) &= \mathscr{R}_s(A) \times \mathscr{R}_s(B) \\
\mathscr{R}_s(A) \cdot \mathscr{R}_s(B) &= A \cdot B \\
\mathscr{R}_s(A) &= \mathscr{R}_{-s}(A)
\end{aligned} \tag{1}$$

Each of the above identities is easy to prove. The last identity states that a rotation about a spinor s is equal to a rotation about a spinor $-s$.

Our new algorithm is an exact algorithm. In particular, it means that all results are mathematically correct. To achieve this, we use arbitrary precision rational numbers from GMP [15] library. The required interface is provided by CGAL's [16] arithmetic module. Clifford algebra is generated by a vector space. We use vectors to describe corresponding points.

II. The proposed algorithm

Our new algorithm constructs a graph of intersections on the border of free configuration space. This is enough to effectively trace a motion path in configuration space. Nevertheless, we also give an information about the possibility of expanding the algorithm to support the complete arrangement of cells.

The overall algorithm is presented in pseudo-code 1.

Each step is discussed separately in the following sections.

A. Collision predicates

A *collision predicate* or shortly a *predicate* is a formula which takes a rotation argument and yields a positive value for collision, a negative when there is no collision and zero when two objects are touching. In general, a predicate is:

$$\mathcal{P}_s : \text{Spin}(3) \longrightarrow \mathbb{R}$$

We distinguish three subsets of Spin(3) depending on the sign of the predicate:

$$\begin{aligned}
F(\mathcal{P}) &:= \{s \in \text{Spin}(3) \colon \mathcal{P}_s > 0\} \\
A(\mathcal{P}) &:= \{s \in \text{Spin}(3) \colon \mathcal{P}_s < 0\} \\
B(\mathcal{P}) &:= \{s \in \text{Spin}(3) \colon \mathcal{P}_s = 0\}
\end{aligned}$$

Algorithm 1 Compute graph of SO(3) configuration space

Require: $\mathcal{R}$ - rotating objects, $\mathcal{O}$ - stationary objects
$P \leftarrow PredicateListFromScene(\mathcal{R}, \mathcal{O})$
$Q \leftarrow SpinQuadricListFromPredicateList(P)$
$Q \leftarrow RemoveDuplicateSpinQuadrics(Q)$
$QSIC \leftarrow \emptyset$
for $pair(q_1, q_2) \in Q$ **do**
  $QSIC \leftarrow IntersectSpinQuadrics(q_1, q_2)$
end for
$QSIP \leftarrow \emptyset$
for $pair(q, qsic) \in (Q, QSIC)$ **do**
  $QSIP \leftarrow IntersectSpinQuadricSpinQSIC(q, qsic)$
end for
$G \leftarrow ComposeGraph(QSIC, QSIP)$

All rotations that cause a collision are usually called *forbidden* and denoted by F. The other subset A is a subset of *allowed* rotations, which do not cause collision. The subset of rotations which cause touching between the rotating object and the obstacle is called a *border* subset of rotations B. In practical usages the subset B is usually considered as a colliding one. Despite that, it is important to distinguish this scenario for the consistency of the theory presented in this paper.

A predicate introduce an *oriented surface* in a configuration space. Configurations that cause a collision, are on one side of the surface. On the other side, there are configurations that do not cause a collision. The surface is a set of configurations that "touch" an obstacle, but not penetrate it. In combinatorial method we consider a set of predicates, created from a given scene. Historically, the first to use a term collision predicate was Canny [17]. A concept of a predicate is also known under different names, such as a "contact" in [6] or a "basic contact", as in [18]. Although the general idea is similar, the detailed theory presented in this paper is new.

We say that two predicates are *equivalent* if and only if they have the same subsets of allowed and forbidden rotations.

Definition 1 (Equivalent predicate)**.** *Two predicates $\mathcal{P}$ and $\mathcal{P}'$ are equivalent if and only if the following holds:*

$$(A(\mathcal{P}) = A(\mathcal{P}')) \wedge (F(\mathcal{P}) = F(\mathcal{P}'))$$

Equivalent predicates are denoted by: $\mathcal{P} \equiv \mathcal{P}'$

For example, we can multiply a predicate by a positive scalar value and get an equivalent predicate. From the definition, it immediately follows that:

$$\mathcal{P} \equiv \mathcal{P}' \longrightarrow B(\mathcal{P}) = B(\mathcal{P}')$$

The definition 1 contains an equal sign - it can be shown that the predicate equivalence is indeed an equivalence relation, as the name suggests. For all applications, we can take any equivalent predicate from the same equivalence class.

A predicate is *opposite* to a given if and only if their allowed and forbidden subsets of rotations are interchanged. Note, that the border sets are the same.

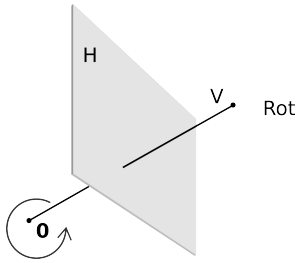

Fig. 1. A predicate of type $\mathcal{H}$

Definition 2 (Opposite predicate)**.** *A predicate $\mathcal{P}$ is opposite to a predicate $\mathcal{P}'$ if and only if the following holds:*

$$(A(\mathcal{P}) = F(\mathcal{P}')) \wedge (F(\mathcal{P}) = A(\mathcal{P}'))$$

A predicate which is opposite to $\mathcal{P}$ is denoted by $-\mathcal{P}$.

The operation is symmetrical:

$$\mathcal{P} \equiv -\mathcal{P}' \iff \mathcal{P}' \equiv -\mathcal{P}$$

Minus operator is suggestive. In fact, a predicate can be multiplied by -1 to get an opposite predicate.

Now, we introduce two basic predicates:

Definition 3 (A half-space predicate $\mathcal{H}$)**.** *Assume that $\mathbf{H} = (U, d)$ is a half-space with a normal vector U and the distance d from the zero. Let V be a non-zero rotating vector. The formula:*

$$\mathcal{H}_s(U, d, V) = U \cdot \mathscr{R}_s(V) + d \tag{2}$$

is called a half-space predicate.

The above formula yields a positive or a negative value depending on whether the rotated vector v has it end on positive or negative side of the half-plane. We assume that v's start point is at 0. A schematic view of $\mathcal{H}$ predicate is shown in figure 1.

The general half-space equation is $(U, d) := U \cdot X + d$. For an arbitrary point X the formula (U, d) yields a positive value when X is *in* the half-space. A given V point is rotating, so $X = \mathscr{R}_s(V)$ and thus $\mathcal{H}_s(U, d, V) = U \cdot \mathscr{R}_s(V) + d$.

Definition 4 (A screw predicate $\mathcal{S}$)**.** *Assume that K and L are ends of a stationary segment and A and B are ends of a rotating segment. The formula:*

$$\mathcal{S}_s(K, L, A, B) = (K \times L) \cdot \mathscr{R}_s(A - B) + (K - L) \cdot \mathscr{R}_s(A \times B) \tag{3}$$

is called a screw predicate.

The above formula yields a positive or a negative value depending on whether the rotating vector is oriented clockwise or counter-clockwise in respect to the stationary vector. A schematic view of $\mathcal{S}$ is shown in figure 2.

The $\mathcal{S}$ predicate is not an obvious construction like $\mathcal{H}$ predicate. One can compare $\mathcal{S}$ predicate to a similar phenomenon is an orientation of magnetic field due to moving charges. The construction of $\mathcal{S}$ predicate is originally based on an observation made in [19] by Devillers and Guigue. In the paper, the authors introduce matrix, whose determinant is an orientation test. This test reveals, whether a screw directed along a given ray turns in the direction of a second ray. From this statement, we took the name of the $\mathcal{S}$ predicate (screw).

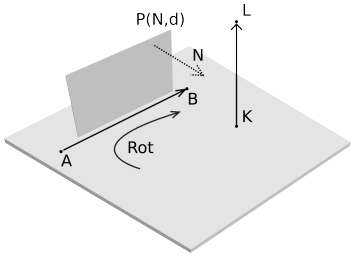

Fig. 2. A predicate of type $\mathcal{S}$

Construction of $\mathcal{S}$ predicate

In the following equations we omit rotation operator $\mathscr{R}$ for increased readability: $A := \mathscr{R}_s(A)$ and $B := \mathscr{R}_s(B)$. First, we define a plane, which normal vector is N and which contains the AB segment:

$$P = N \cdot X + d = N \cdot X - N \cdot A = N \cdot (X - A)$$

The normal vector N depends on the rotating points A and B and is equal to:

$$N = (B - A) \times (L - K)$$

It cannot be guaranteed that N is non-zero. We leave this case for later discussion. Assuming, that N is a non-zero vector, the orientation test is a "point on plane side" test for point $X = K$:

$$\begin{aligned}
P &= ((B - A) \times (L - K)) \cdot (K - A) = \\
&((A - B) \times (K - L)) \cdot (K - A) = \\
&(A - B) \cdot ((K - L)) \times (K - A)) = \\
&(A - B) \cdot (K \times K + L \times A - K \times A - L \times K) = \\
&(A - B) \cdot (-L \times K) + (A - B) \cdot ((L - K) \times A) = \\
&(A - B) \cdot (K \times L) + (-B) \cdot ((L - K) \times A) = \\
&(A - B) \cdot (K \times L) - ((L - K) \times A) \cdot B = \\
&(A - B) \cdot (K \times L) - (L - K) \cdot (A \times B) = \\
&(K \times L) \cdot (A - B) + (K - L) \cdot (A \times B)
\end{aligned}$$

which is a definition of $\mathcal{S}$ predicate. In the above calculations we used the identities (1).

A $\mathcal{S}$ predicate features some symmetries of the arguments. They are useful, when $\mathcal{TT}$ predicate is introduced:

Lemma 1 (Equivalent and opposite $\mathcal{S}$ predicates)**.** *Let $\mathcal{S}(K, L, A, B)$ be a predicate. The following identities hold:*

$$\begin{aligned}
\mathcal{S}(K, L, A, B) &\equiv \mathcal{S}(L, K, B, A) \\
&\equiv -\mathcal{S}(L, K, A, B) \\
&\equiv -\mathcal{S}(K, L, B, A)
\end{aligned} \tag{4}$$

Proof: All three identities are similar. We prove the first and the second one:

$$\begin{aligned}
&\mathcal{S}(K, L, A, B) = \\
&(K \times L) \cdot \mathscr{R}_s(A - B) + (K - L) \cdot \mathscr{R}_s(A \times B) = \\
&-(L \times K) \cdot -\mathscr{R}_s(B - A) + -(L - K) \cdot -\mathscr{R}_s(B \times A) = \\
&(L \times K) \cdot \mathscr{R}_s(B - A) + (L - K) \cdot \mathscr{R}_s(B \times A) = \\
&\mathcal{S}(L, K, B, A)
\end{aligned}$$

The second identity:

$$\begin{aligned}
&\mathcal{S}(K, L, A, B) = \\
&(K \times L) \cdot \mathscr{R}_s(A - B) + (K - L) \cdot \mathscr{R}_s(A \times B) = \\
&-(L \times K) \cdot \mathscr{R}_s(A - B) + -(L - K) \cdot \mathscr{R}_s(A \times B) = \\
&-((L \times K) \cdot \mathscr{R}_s(A - B) + (L - K) \cdot \mathscr{R}_s(A \times B)) = \\
&-\mathcal{S}(L, K, A, B)
\end{aligned}$$

■

In the definition of an $\mathcal{S}$ predicate, we assumed that the normal vector is non-zero. When the normal vector is zero, the predicate value is undefined. This corresponds to a situation when the rotating segment is parallel to the stationary one. The subset of rotations which have the described property is called an associated singular surface for a $\mathcal{S}$ predicate.

Definition 5 (Associated singular surface for a $\mathcal{S}$ predicate)**.** *Let $\mathcal{S}(K, L, A, B)$ be a predicate. The following surface $\tilde{\mathcal{S}}$ is a singular surface of $\mathcal{S}$.*

$$\tilde{\mathcal{S}} = \{s \in \mathit{Spin}(3) \colon \|(K - L) \times \mathscr{R}_s(A - B)\|^2 = 0\} \tag{5}$$

The value of the predicate is undefined for all points on $\tilde{\mathcal{S}}$.

Singular surface $\tilde{\mathcal{S}}$ is a 4-th degree surface in Spin(3). In the proposed algorithm constructing SO(3) configuration space, we do not need to evaluate rotations on a singular surface directly. Nevertheless, there exist a scenario when we may need to evaluate $\mathcal{S}$ predicate value on a singular surface. This is a situation when a rotation is given by a user of the system, and it is on the singular surface. If such situation occurs, we can execute one of the classical procedures for triangle intersection (see [19], [20] or [21]) with the given rotation and check the result. In case the proposed algorithm is implemented in parallel, we can also use a highly parallel triangle intersection routines, as presented in [22]. To check whether a given rotation lies on a singular surface, we can plug it directly into (5).

It can be shown that $\mathcal{H}$ and $\mathcal{S}$ are both special cases of a new predicate $\mathcal{G}$. As a result, we need only a $\mathcal{G}$ predicate to be considered.

Definition 6 (A general predicate $\mathcal{G}$)**.** *Assume that K and L are ends of a stationary segment and A and B are ends of a rotating segment and c is a scalar. The formula:*

$$\begin{aligned}
&\mathcal{G}_s(K, L, A, B, c) = \\
&(K \times L) \cdot \mathscr{R}_s(A - B) + (K - L) \cdot \mathscr{R}_s(A \times B) + c
\end{aligned}$$

is called a general predicate.

$\mathcal{G}$ predicate is artificial by the construction. This is because, both $\mathcal{H}$ and $\mathcal{S}$ can be converted to $\mathcal{G}$ easily. The equivalence of $\mathcal{H}$ to $\mathcal{G}$ and $\mathcal{S}$ to $\mathcal{G}$ predicates is due to the following two propositions:

Proposition 1 ($\mathcal{H}$ to $\mathcal{G}$ predicate correspondence)**.** *A predicate $\mathcal{H}(U, d, V)$ is a special case of a predicate*

$\mathcal{G}(K, L, A, B, c)$ *with the following parameters:*

$$\begin{aligned} K &= V \times R \\ L &= V \times (V \times R) \\ A &= U \\ B &= -U \\ c &= 2d\|V \times R\|^2 \end{aligned}$$

where R is an arbitrary non-zero vector that is not parallel to V.

Proof: We plug in the above parameters into a $\mathcal{G}$ predicate to obtain:

$$\begin{aligned} K \times L &= (V \times R) \times (V \times (V \times R)) \\ &= V(V \times R) \cdot (V \times R) - (V \times R)(V \times R) \cdot V \\ &= V\|V \times R\|^2 \end{aligned}$$

$$\begin{aligned} A - B &= 2U \\ A \times B &= 0 \end{aligned}$$

$$\begin{aligned} &\mathcal{G}(K, L, A, B, c) \\ &= (K \times L) \cdot \mathscr{R}_s(A - B) + (K - L) \cdot \mathscr{R}_s(A \times B) + c \\ &= V\|V \times R\|^2 \cdot \mathscr{R}_s(2U) + 2d\|V \times R\|^2 \\ &= 2\|V \times R\|^2(V \cdot \mathscr{R}_s(U) + d) \\ &= 2\|V \times R\|^2 \mathcal{H}(U, d, V) \\ &\equiv \mathcal{H}(U, d, V) \end{aligned}$$

■

During our research we tried different methods of choosing an R vector. We use a simple and practical one: algorithm 2. The method is as follows. First, we project the given vector onto an axial plane in a way, that the projection is not a single point (it means, that the vector is not orthogonal to the axial plane). This is realized by checking consecutive coordinates for being non-zero. Secondly, we take a vector, which is orthogonal to the given one, and which lies on the selected axial plane. Finally, we recover the vector's coordinate which was lost during the projection and set is to the original value. In other words, we can leave the coordinate unchanged all the time, but do the operation of taking an orthogonal vector just on the selected axial plane (two vector coordinates).

Algorithm 2 Select an R vector

Require: v - a non-zero vector
  if $v.x \neq 0$ **then**
    return $vector[-v.y, v.x, v.z]$ - *Z is unchanged*
  else $\{v.y \neq 0\}$
    return $vector[v.x, -v.z, v.y]$ - *X is unchanged*
  else
    return $vector[v.z, v.y, -v.x]$ - *Y is unchanged*
  end if

Proposition 2 ($\mathcal{S}$ to $\mathcal{G}$ predicate correspondence)**.** *Predicate $\mathcal{S}(K, L, A, B)$ is a special case of a predicate $\mathcal{G}(K, L, A, B, c)$ with the following setting:*

$$c = 0$$

Proof: Setting $c = 0$ in $\mathcal{G}$ predicate immediately gives a definition of $\mathcal{S}$ predicate. ■

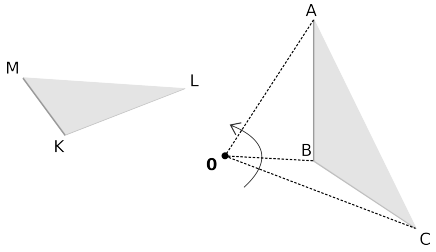

Fig. 3. A predicate of type $\mathcal{TT}$

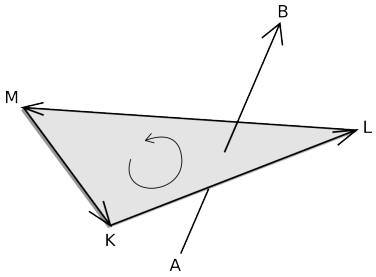

Fig. 4. Construction of $\mathcal{TT}$ predicate

We assume, that the proposed algorithm should trace collision of polyhedra borders. Without any loss, it can be assumed that all faces are triangles. Two (empty) polyhedra collide if and only if their borders intersect. This is realized by checking for collisions between all pairs of triangles: one from the rotating polyhedron and the other one from scene obstacles. By using ideas from [19] about collision of triangles, we define a new predicate that detects a triangle-triangle collision. A schematic view of $\mathcal{TT}$ predicate is shown in figure 3:

Proposition 3 (A triangle-triangle predicate $\mathcal{TT}$)**.** *A generic collision test between a stationary triangle KLM and a rotating triangle ABC can be expressed with 9 predicates of type $\mathcal{S}$.*

$$\begin{aligned} &\mathcal{TT}(K, L, M, A, B, C) > 0 \iff \\ &intersect\_triangle\_triangle(\triangle KLM, \mathscr{R}_s(\triangle ABC)) \end{aligned}$$

The characteristic matrix $M_{\mathcal{TT}}$ is:

$$M_{\mathcal{TT}} = \begin{bmatrix} \mathcal{S}(K, L, A, B) & \mathcal{S}(K, L, B, C) & \mathcal{S}(K, L, C, A) \\ \mathcal{S}(L, M, A, B) & \mathcal{S}(L, M, B, C) & \mathcal{S}(L, M, C, A) \\ \mathcal{S}(M, K, A, B) & \mathcal{S}(M, K, B, C) & \mathcal{S}(M, K, C, A) \end{bmatrix}$$

The predicate yields a positive value (collision) if and only if there exist a row or a column in $M_{\mathcal{TT}}$ that all it's elements are of the same non-zero sign.

Proof: Two triangles collide if and only if any edge of the first triangle intersects the second triangle or vice versa. A single $\mathcal{S}$ predicate is used to detect the orientation of a rotating directed segment in respect to the other stationary directed segment. We start with forming a single chain of three $\mathcal{S}$ predicates for directed segments KL, LM and MK and directed rotating segment AB. This is depicted in figure 4. Triangle vertices can be oriented clockwise or counter-clockwise. Assuming that the vertices' order is given, we define:

$$\begin{aligned} \omega :=&(\mathcal{S}(K, L, A, B) > 0) \wedge (\mathcal{S}(L, M, A, B) > 0) \wedge \\ &(\mathcal{S}(M, K, A, B) > 0) \end{aligned}$$

which detects whether an oriented rotating segment collides with a stationary triangle with ordered vertices. Now we generalize this test to handle the case with a different orientation of the rotating segment or a different orientation of triangle's vertices. There are three additional tests to handle. We write a generalized test, which ignores both orientations:

$$\begin{aligned}\omega' :=&[(\mathcal{S}(K,L,A,B)>0)\wedge(\mathcal{S}(L,M,A,B)>0)\wedge\\ &(\mathcal{S}(M,K,A,B)>0)]\vee\\ &[(\mathcal{S}(K,M,A,B)>0)\wedge(\mathcal{S}(M,L,A,B)>0)\wedge\\ &(\mathcal{S}(L,K,A,B)>0)]\vee\\ &[(\mathcal{S}(K,L,B,A)>0)\wedge(\mathcal{S}(L,M,B,A)>0)\wedge\\ &(\mathcal{S}(M,K,B,A)>0)]\vee\\ &[(\mathcal{S}(K,M,B,A)>0)\wedge(\mathcal{S}(M,L,B,A)>0)\wedge\\ &(\mathcal{S}(L,K,B,A)>0)]\end{aligned}$$

by using the identities (4) we can do the following reduction:

$$\begin{aligned}\omega' :=&[(\mathcal{S}(K,L,A,B)>0)\wedge(\mathcal{S}(L,M,A,B)>0)\wedge\\ &(\mathcal{S}(M,K,A,B)>0)]\vee\\ &[(\mathcal{S}(K,L,A,B)<0)\wedge(\mathcal{S}(L,M,A,B)<0)\wedge\\ &(\mathcal{S}(M,K,A,B)<0)]\end{aligned}$$

The last step is to combine the remaining cases for segments BC and CA:

$$\begin{aligned}\pi :=&[(\mathcal{S}(K,L,A,B)>0)\wedge(\mathcal{S}(L,M,A,B)>0)\wedge\\ &(\mathcal{S}(M,K,A,B)>0)]\vee\\ &[(\mathcal{S}(K,L,A,B)<0)\wedge(\mathcal{S}(L,M,A,B)<0)\wedge\\ &(\mathcal{S}(M,K,A,B)<0)]\vee\\ &[(\mathcal{S}(K,L,B,C)>0)\wedge(\mathcal{S}(L,M,B,C)>0)\wedge\\ &(\mathcal{S}(M,K,B,C)>0)]\vee\\ &[(\mathcal{S}(K,L,B,C)<0)\wedge(\mathcal{S}(L,M,B,C)<0)\wedge\\ &(\mathcal{S}(M,K,B,C)<0)]\vee\\ &[(\mathcal{S}(K,L,C,A)>0)\wedge(\mathcal{S}(L,M,C,A)>0)\wedge\\ &(\mathcal{S}(M,K,C,A)>0)]\vee\\ &[(\mathcal{S}(K,L,C,A)<0)\wedge(\mathcal{S}(L,M,C,A)<0)\wedge\\ &(\mathcal{S}(M,K,C,A)<0)]\end{aligned}$$

Similarly, if we analyse the mirror case with edges AB, BC and CA intersecting triangle KLM, we get the following test π':

$$\begin{aligned}\pi' :=&[(\mathcal{S}(K,L,A,B)>0)\wedge(\mathcal{S}(K,L,B,C)>0)\wedge\\ &(\mathcal{S}(K,L,C,A)>0)]\vee\\ &[(\mathcal{S}(K,L,A,B)<0)\wedge(\mathcal{S}(K,L,B,C)<0)\wedge\\ &(\mathcal{S}(K,L,C,A)<0)]\vee\\ &[(\mathcal{S}(L,M,A,B)>0)\wedge(\mathcal{S}(L,M,B,C)>0)\wedge\\ &(\mathcal{S}(L,M,C,A)>0)]\vee\\ &[(\mathcal{S}(L,M,A,B)<0)\wedge(\mathcal{S}(L,M,B,C)<0)\wedge\\ &(\mathcal{S}(L,M,C,A)<0)]\vee\\ &[(\mathcal{S}(M,K,A,B)>0)\wedge(\mathcal{S}(M,K,B,C)>0)\wedge\\ &(\mathcal{S}(M,K,C,A)>0)]\vee\\ &[(\mathcal{S}(M,K,A,B)<0)\wedge(\mathcal{S}(M,K,B,C)<0)\wedge\\ &(\mathcal{S}(M,K,C,A)<0)]\end{aligned}$$

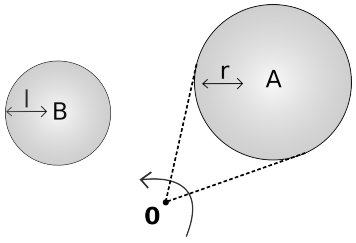

Fig. 5. A predicate of type $\mathcal{BB}$

Finally, we select all unique predicates from π and π' and compose them in a matrix $M_{\mathcal{TT}}$:

$$\begin{bmatrix}\mathcal{S}(K,L,A,B) & \mathcal{S}(K,L,B,C) & \mathcal{S}(K,L,C,A)\\ \mathcal{S}(L,M,A,B) & \mathcal{S}(L,M,B,C) & \mathcal{S}(L,M,C,A)\\ \mathcal{S}(M,K,A,B) & \mathcal{S}(M,K,B,C) & \mathcal{S}(M,K,C,A)\end{bmatrix}$$

One can observe that π and π' tests correspond exactly to columns and rows of $M_{\mathcal{TT}}$ respectively. This completes the proof. ■

Assume that $\mathcal{TT}_s^i, i = 1...nm$ are $\mathcal{TT}$ predicates of all triangle pairs for a given scene. The following formula is a $\mathcal{TT}$ *predicate sentence* for a scene:

$$\mathcal{TT}sentence_s := \bigcup_i \mathcal{TT}_s^i$$

The sentence yields a positive value if there is any collision in the scene.

Remark 1. *$\mathcal{TT}$ include a total of nine predicates of type $\mathcal{S}$. Because of that, there are also nine singular surfaces for a $\mathcal{TT}$ predicate.*

An alternative approach is to build a scene only from balls (spheres). There are several advantages of that discussed in one of the following sections. In a ball-only scene, both the rotating object and obstacles are sets of balls. We only need to consider one type of collision - a rotating ball and a stationary ball. A schematic view of $\mathcal{BB}$ predicate is shown in figure 5:

It can be shown, that a collision predicate for two balls (one is rotating, the other is stationary) is equivalent to an $\mathcal{H}$ predicate:

Proposition 4 (A ball-ball predicate $\mathcal{BB}$)**.** *A generic collision test between a rotating ball of radius r centered at A and a stationary ball of radius l centered at B can be expressed with one predicate of type $\mathcal{H}$.*

Ball-ball predicate is defined as:

$$\mathcal{BB}(B,l,A,r) = (r+l) - \|\mathscr{R}_s(A) - B\|$$

An equivalent $\mathcal{H}$ predicate is:

$$\mathcal{BB}(B,l,A,r) \equiv \mathcal{H}(2B, (r+l)^2 - \|A\|^2 - \|B\|^2, A)$$

The predicate yields positive value (collision) if and only if there is a collision between two balls.

Proof: Let $\mathcal{BB}$ be a ball-ball collision predicate. We consider the set of all rotations causing collision. The chain

of equivalent sets is:

$$\begin{aligned}
\mathcal{BB}(B,l,A,r) > 0 &\iff \\
(r+l) - \|\mathscr{R}_s(A) - B\| > 0 &\iff \\
(r+l) > \|\mathscr{R}_s(A) - B\| &\iff \\
(r+l)^2 > \|\mathscr{R}_s(A) - B\|^2 &\iff \\
(r+l)^2 > (s^{-1}As - B)^2 &\iff \\
(r+l)^2 > (s^{-1}As)(s^{-1}As) - s^{-1}AsB - Bs^{-1}As + BB & \\
\iff (2B)\cdot s^{-1}As + (r+l)^2 - AA - BB > 0 & \\
\iff \mathcal{H}(2B, (r+l)^2 - \|A\|^2 - \|B\|^2, A) > 0 &
\end{aligned}$$

The same equations are for non-collision scenario, when the $>$ relation is changed to $<$. Thus,

$$\mathcal{BB}(B,l,A,r) \equiv \mathcal{H}(2B, (r+l)^2 - \|A\|^2 - \|B\|^2, A)$$

by definition 1. ∎

Assume that $\mathcal{BB}^i_s, i = 1...nm$ are $\mathcal{BB}$ predicates of all ball pairs for a given scene. The following formula is a $\mathcal{BB}$ *predicate sentence* for a scene:

$$\mathcal{BB}sentence_s := \bigcup_i \mathcal{BB}^i_s$$

Remark 2. *$\mathcal{BB}$ is composed of one predicate of type $\mathcal{H}$. It is free of singular surfaces, so a $\mathcal{BB}$ predicate is also free of singular surfaces.*

B. Spin-quadrics

A generic $\mathcal{G}$ predicate sets an oriented quadric (a quadratic surface) in configuration space. The quadric is a quadratic form in spin-space, thus it is henceforth called *spin-quadric*. In case of Spin(3) configuration space we have:

Proposition 5. *A general predicate $\mathcal{G}(K,L,A,B,c)$ can be reduced to a quadratic form in Spin(3). It represents a spin-quadric which can be expressed by:*

$$\mathcal{G}_s = \mathbf{s}^T M_{\mathcal{G}} \mathbf{s}$$

with matrix

$$M_{\mathcal{G}} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{12} & a_{22} & a_{23} & a_{24} \\ a_{13} & a_{23} & a_{33} & a_{34} \\ a_{14} & a_{24} & a_{34} & a_{44} \end{bmatrix}$$

or, equivalently:

$$\begin{aligned}
\mathcal{G}_s = a_{11}s_{12}^2 + a_{22}s_{23}^2 + a_{33}s_{31}^2 + a_{44}s_0^2 + \\
2(a_{12}s_{12}s_{23} + a_{13}s_{12}s_{31} + a_{14}s_{12}s_0 + \\
a_{23}s_{23}s_{31} + a_{24}s_{23}s_0 + a_{34}s_{31}s_0)
\end{aligned}$$

where $\mathbf{s} = [s_{12}, s_{23}, s_{31}, s_0]^T$ and $M_{\mathcal{G}}$'s elements are linear expressions of $\mathcal{G}$'s parameters K, L, A, B and c:

$$\begin{aligned}
a_{11} &= -P_{xyz} - P_{yzx} + P_{zxy} - Q_{xyz} - Q_{yzx} + Q_{zxy} + c \\
a_{22} &= +P_{xyz} - P_{yzx} - P_{zxy} + Q_{xyz} - Q_{yzx} - Q_{zxy} + c \\
a_{33} &= -P_{xyz} + P_{yzx} - P_{zxy} - Q_{xyz} + Q_{yzx} - Q_{zxy} + c \\
a_{44} &= +P_{xyz} + P_{yzx} + P_{zxy} + Q_{zxy} + Q_{yzx} + Q_{xyz} + c \\
a_{12} &= +P_{xxy} + P_{zyz} + Q_{xxy} + Q_{zyz} \\
a_{13} &= +P_{yxy} + P_{zzx} + Q_{yxy} + Q_{zzx} \\
a_{14} &= +P_{xzx} - P_{yyz} - Q_{xzx} + Q_{yyz} \\
a_{23} &= +P_{xzx} + P_{yyz} + Q_{xzx} + Q_{yyz} \\
a_{24} &= +P_{yxy} - P_{zzx} - Q_{yxy} + Q_{zzx} \\
a_{34} &= -P_{xxy} + P_{zyz} + Q_{xxy} - Q_{zyz}
\end{aligned}$$

where P and Q are linear combinations:

$$\begin{aligned}
P_{\alpha\beta\gamma} &= (K_\alpha - L_\alpha)(A_\beta B_\gamma - A_\gamma B_\beta) \\
Q_{\alpha\beta\gamma} &= (A_\alpha - B_\alpha)(K_\beta L_\gamma - K_\gamma L_\beta)
\end{aligned}$$

Proof: Let $K = [K_x, K_y, K_z]^T$, $L = [L_x, L_y, L_z]^T$, $A = [A_x, A_y, A_z]^T$, $B = [B_x, B_y, B_z]^T$ and c are given parameters. We directly expand $\mathcal{G}(K,L,A,B,c)$ with the coefficients by using a symbolic calculator. We apply automatic formula simplification and obtain:

$$\begin{aligned}
&\mathcal{G} = \\
&(-P_{xyz} - P_{yzx} + P_{zxy} - Q_{xyz} - Q_{yzx} + Q_{zxy})s_{12}^2 + \\
&(P_{xyz} - P_{yzx} - P_{zxy} + Q_{xyz} - Q_{yzx} - Q_{zxy})s_{23}^2 + \\
&(-P_{xyz} + P_{yzx} - P_{zxy} - Q_{xyz} + Q_{yzx} - Q_{zxy})s_{31}^2 + \\
&(P_{xyz} + P_{yzx} + P_{zxy} + Q_{zxy} + Q_{yzx} + Q_{xyz})s_0^2 + \\
&2(P_{xxy} + P_{zyz} + Q_{xxy} + Q_{zyz})s_{12}s_{23} + \\
&2(P_{yxy} + P_{zzx} + Q_{yxy} + Q_{zzx})s_{12}s_{31} + \\
&2(P_{xzx} - P_{yyz} - Q_{xzx} + Q_{yyz})s_{12}s_0 + \\
&2(P_{xzx} + P_{yyz} + Q_{xzx} + Q_{yyz})s_{23}s_{31} + \\
&2(P_{yxy} - P_{zzx} - Q_{yxy} + Q_{zzx})s_{23}s_0 + \\
&2(-P_{xxy} + P_{zyz} + Q_{xxy} - Q_{zyz})s_{31}s_0 + c
\end{aligned}$$

It is almost a quadratic form, but the c scalar. We use the basic property $s_{12}^2 + s_{23}^2 + s_{31}^2 + s_0^2 = 1$ of the Spin(3) space, and write:

$$c = c(s_{12}^2 + s_{23}^2 + s_{31}^2 + s_0^2) = cs_{12}^2 + cs_{23}^2 + cs_{31}^2 + cs_0^2$$

From the above, we can write an equal formula:

$$\begin{aligned}
&\mathcal{G} = \\
&(-P_{xyz} - P_{yzx} + P_{zxy} - Q_{xyz} - Q_{yzx} + Q_{zxy} + c)s_{12}^2 + \\
&(P_{xyz} - P_{yzx} - P_{zxy} + Q_{xyz} - Q_{yzx} - Q_{zxy} + c)s_{23}^2 + \\
&(-P_{xyz} + P_{yzx} - P_{zxy} - Q_{xyz} + Q_{yzx} - Q_{zxy} + c)s_{31}^2 + \\
&(P_{xyz} + P_{yzx} + P_{zxy} + Q_{zxy} + Q_{yzx} + Q_{xyz} + c)s_0^2 + \\
&2(P_{xxy} + P_{zyz} + Q_{xxy} + Q_{zyz})s_{12}s_{23} + \\
&2(P_{yxy} + P_{zzx} + Q_{yxy} + Q_{zzx})s_{12}s_{31} + \\
&2(P_{xzx} - P_{yyz} - Q_{xzx} + Q_{yyz})s_{12}s_0 + \\
&2(P_{xzx} + P_{yyz} + Q_{xzx} + Q_{yyz})s_{23}s_{31} + \\
&2(P_{yxy} - P_{zzx} - Q_{yxy} + Q_{zzx})s_{23}s_0 + \\
&2(-P_{xxy} + P_{zyz} + Q_{xxy} - Q_{zyz})s_{31}s_0
\end{aligned}$$

which is precisely a quadratic form, as stated in the theorem. ∎

As a result of the this step, we get a list of spin-quadrics. These spin-quadrics introduce an arrangement in Spin(3). A predicate sentence for the scene is also stored in order to be used later. During the research, we learned that some of the computed spin-surfaces can be doubled. Mostly, these are doubled because of the $\mathcal{S}$ predicates. If we consider a triangulated polyhedron, there are a lot of edges which belong to two different triangles. In fact, this gives two overlapping spin-surfaces with the opposite orientation. There is no point in taking the both spin-surfaces into the arrangement, so we omit one of these spin-surfaces. In general, we take only one spin-surface from all that are the same up to a scaling factor. To recover the information about the missing spin-surfaces, we maintain a list of "virtual spin-surfaces", pointing to the remaining "real" ones. We also store the information, whether the virtual spin-surface has got the same or the opposite orientation. We have tested our algorithm on different scenes, and usually more than a half of all spin-surfaces were doubled, and thus removed. In an algorithm with a complexity of $O(n^3)$ this is a theoretical speed-up of factor 8. The reason that there are so many doubled spin-surfaces is that we have manifold geometry - there are no free edges of triangles. In a scene consisting only of non-intersecting triangles there would be no doubled spin-surfaces at all. Note, that in a ball-only scene, there should be very little redundancy in spin-surfaces. Rarely, there are doubled balls in scene. So, because the spin-surfaces are only generated from a ball-ball collision, there can not be many doubled spin-surfaces.

C. Spin-quadric intersection as graph edges

Computing an intersection of two spin-quadrics is not an easy task. It is not easy, even in the case of an intersection of two quadratic surfaces in $\mathbb{R}^3$. There exist algorithms that partially solve this problem. In particular, [23], [24] and [25] are known methods. Only recently, first complete implementations of quadric intersection in $\mathbb{R}^3$ were developed: [26] and [27]. The case of an intersection, where the resulting curve is not singular, was solved first. This is called a smooth case. A standard procedure is to follow a method proposed by Levin [28]. The problematic cases are those, involving singular intersections. Much work is needed to handle all specific cases, one by one. Currently, two implementations are available: QI library [27] (available online [29]) and Berberich's implementation [26]. The first one, returns parametrization of the resulting intersection. The latter does not. Both implementations present a similar performance. Although the dimensionality of Spin(3) and $\mathbb{R}^3$ is the same, both spaces have a different topology. One of the main results of this paper is to show that, a quadric intersection library for $\mathbb{R}^3$ can also be used to perform intersections in Spin(3).

Theorem 1 (Spin-surface intersecting algorithm)**.** *Let $\mathcal{A}$ be an algorithm for quadric intersection in $\mathbb{P}^3$ (projective space). By using only $\mathcal{A}$, it is possible to construct an algorithm $\mathcal{B}$ which intersects a pair of spin-quadrics in Spin(3). The asymptotic complexity of $\mathcal{B}$, is the same as the asymptotic complexity of $\mathcal{A}$.*

Proof: Assume that in Spin(3) there are given two spin-quadrics with matrices $M_{\mathcal{G}}$ equal to P and Q. The intersection is a set of spinors $\mathbf{s} = [s_{12}, s_{23}, s_{31}, s_0]^T$ satisfying:

$$\begin{cases} \mathbf{s}^T\mathbf{s} = 1 \\ \mathbf{s}^T P\mathbf{s} = 0 \\ \mathbf{s}^T Q\mathbf{s} = 0 \end{cases}$$

It is impossible that all of $s_{12}, s_{23}, s_{31}, s_0$ are equal to zero simultaneously because of the first equation. Assume for now that s_0 is non-zero. We can divide the second and the third equation by s_0^2, obtaining:

$$\begin{cases} \mathbf{t}^T P\mathbf{t} = 0 \\ \mathbf{t}^T Q\mathbf{t} = 0 \end{cases}$$

The newly introduced vector $\mathbf{t}$ is equal to $[\frac{s_{12}}{s_0}, \frac{s_{23}}{s_0}, \frac{s_{31}}{s_0}, 1]^T$. We can observe that the second and the third equations form a quadric intersection problem in $\mathbb{R}^3$. This problem can be effectively solved by $\mathcal{A}$. Both quadrics are given in terms of $t_x = \frac{s_{12}}{s_0}, t_y = \frac{s_{23}}{s_0}, t_z = \frac{s_{31}}{s_0}$. Now, we use the assumption that $\mathcal{A}$ returns a parametrized intersection curve $\Gamma(\xi) = [\Gamma_x(\xi), \Gamma_y(\xi), \Gamma_z(\xi), \Gamma_w(\xi)]^T$ is given in homogeneous coordinates:

$$t_x = \frac{\Gamma_x(\xi)}{\Gamma_w(\xi)}, t_y = \frac{\Gamma_y(\xi)}{\Gamma_w(\xi)}, t_z = \frac{\Gamma_z(\xi)}{\Gamma_w(\xi)}, \xi \in \mathbb{P}^1$$

To recover all of $s_{12}, s_{23}, s_{31}, s_0$, we first compute s_0. Summing up the squares of t_x, t_y and t_z we get:

$$t_x^2 + t_y^2 + t_z^2 = \frac{s_{12}^2}{s_0^2} + \frac{s_{23}^2}{s_0^2} + \frac{s_{31}^2}{s_0^2} = \frac{s_{12}^2 + s_{23}^2 + s_{31}^2}{s_0^2}$$
$$= \frac{1 - s_0^2}{s_0^2} = \frac{1}{s_0^2} - 1$$

so,
$1 + t_x^2 + t_y^2 + t_z^2 = \frac{1}{s_0^2}$ and $s_0^2 = \frac{1}{1+t_x^2+t_y^2+t_z^2}$
by plugging in the intersection curve Γ, one can write:

$$s_0^2 = \frac{1}{\frac{\Gamma_w(\xi)^2}{\Gamma_w(\xi)^2} + \frac{\Gamma_x(\xi)^2}{\Gamma_w(\xi)^2} + \frac{\Gamma_y(\xi)^2}{\Gamma_w(\xi)^2} + \frac{\Gamma_z(\xi)^2}{\Gamma_w(\xi)^2}}$$
$$= \frac{\Gamma_w(\xi)^2}{\Gamma_w(\xi)^2 + \Gamma_x(\xi)^2 + \Gamma_y(\xi)^2 + \Gamma_z(\xi)^2}$$

Using the last identity of (1), a rotation by a spinor s is identified with a rotation by a spinor $-s$. Hence, in the above equation we can take a square root of both sides without a loss of generality. Finally we get:

$$s_0 = \frac{\Gamma_w(\xi)}{\|\Gamma(\xi)\|}$$

where $\|\Gamma(\xi)\| = \sqrt{\Gamma_w(\xi)^2 + \Gamma_x(\xi)^2 + \Gamma_y(\xi)^2 + \Gamma_z(\xi)^2}$. The remaining spinor coordinates are:

$$s_{12} = s_0 t_{12} = \frac{\Gamma_w(\xi)}{\|\Gamma(\xi)\|}\frac{\Gamma_x(\xi)}{\Gamma_w(\xi)} = \frac{\Gamma_x(\xi)}{\|\Gamma(\xi)\|}$$
$$s_{23} = s_0 t_{23} = \frac{\Gamma_w(\xi)}{\|\Gamma(\xi)\|}\frac{\Gamma_y(\xi)}{\Gamma_w(\xi)} = \frac{\Gamma_y(\xi)}{\|\Gamma(\xi)\|}$$
$$s_{31} = s_0 t_{31} = \frac{\Gamma_w(\xi)}{\|\Gamma(\xi)\|}\frac{\Gamma_z(\xi)}{\Gamma_w(\xi)} = \frac{\Gamma_z(\xi)}{\|\Gamma(\xi)\|}$$

The above formulas can be finally rewritten as the *spin-quadric intersection parametrization*:

$$s(\xi) = \pm\frac{\Gamma_w(\xi) + \Gamma_x(\xi)\mathbf{e}_{12} + \Gamma_y(\xi)\mathbf{e}_{23} + \Gamma_z(\xi)\mathbf{e}_{31}}{\|\Gamma(\xi)\|}$$

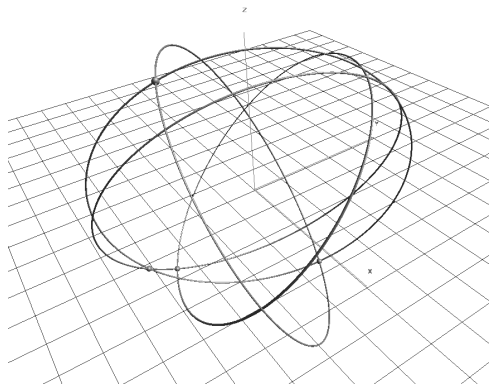

Fig. 6. An example set of spin-QSICs in Spin(3)

or, equivalently in a vector form:

$$\mathbf{s}(\xi) = \pm\frac{[\Gamma_x(\xi), \Gamma_y(\xi), \Gamma_z(\xi), \Gamma_w(\xi)]^T}{\|\Gamma(\xi)\|}$$
$$\mathbf{s}(\xi) = \pm\frac{\Gamma(\xi)}{\|\Gamma(\xi)\|}$$

where $\Gamma := intersect(P, Q)$.

It can be easily seen that the complexity of $\mathcal{B}$ is the same as the complexity of $\mathcal{A}$. A note is also needed about initial choice of s_0 as the coordinate by which we divided the remaining coordinates. Because it is not possible that all of $s_{12}, s_{23}, s_{31}, s_0$ are simultaneously zero, we can non-constructively divide the Spin(3) space into a number of fragments in which the selected spin component is non-zero. In each of these fragments, we repeat the proof, with the respect of different spin component. ■

The last theorem strictly depends on the fact, that the intersecting algorithm represents the intersection curve in homogeneous coordinates. In Spin(3) configuration space, quadric intersection generates a curve, which is called a *spin quadric surface intersection curve* (spin-QSIC) in analogy to *quadric surface intersection curve* (QSIC), as used in literature [30] and [31].

In figure 6 we show an example projection of spin-QSIC curves onto $\mathbb{R}^3$. Three coefficients: s_{12}, s_{23} and s_{31} are projected onto screen and s_0 is the spin-QSIC thickness.

D. Quadric surface intersection curve

In this section we present a short description and major features of the algorithm by Dupont. It was published in three parts [32], [33] and [34]. It is a complete implementation of quadric surface intersection in $\mathbb{R}^3$. The algorithm works in real projective space $\mathbb{P}^3$. Points are represented by quadruplets

$$X = [X_0, X_1, X_2, X_3]^T \neq [0, 0, 0, 0]^T$$

with the equivalence relation $[X_0, X_1, X_2, X_3]^T \sim [\lambda X_0, \lambda X_1, \lambda X_2, \lambda X_3]^T$ for all $\lambda \neq 0$ (homogeneous space). Dupont et. at. define a quadric surface Q_S by an implicit equation of degree 2 in $\mathbb{P}^3$:

$$\sum_{0 \leq i \leq j \leq 3} \alpha_{ij} X_i X_j = 0,\ \alpha_{ij} \in \mathbb{Q}$$

The equation is a quadratic form in $\mathbb{P}^3$, hence it can be written as $X^T S X = 0$, with a real symmetric matrix S associated to Q_S. The upper-left 3x3 matrix of S will be called S'.

The initial step of the algorithm [32] is computation of inertia. By looking at eigenvalues of S and S', we can deduce the type of a given quadric surface. It is known, that for a symmetric matrix, all eigenvalues are real. Assume that, σ_S^+ and σ_S^- are the numbers of positive and negative eigenvalues of S, respectively. Similarly, assume that $\sigma_{S'}^+$ and $\sigma_{S'}^-$ are the numbers of positive and negative eigenvalues of S'. Dupont et. al. introduce a two pairs of numbers, each called the *inertia* of Q_S and $Q_{S'}$:

$$\sigma_S = (\max(\sigma_S^+, \sigma_S^-), \min(\sigma_S^+, \sigma_S^-)),$$
$$\sigma_{S'} = (\max(\sigma_{S'}^+, \sigma_{S'}^-), \min(\sigma_{S'}^+, \sigma_{S'}^-))$$

Depending on the values of both inertias, it is possible to distinguish the following euclidean types of a quadric: empty, ellipsoid, hyperboloid of two sheets, elliptic paraboloid, point, point at infinity, hyperboloid of one sheet, hyperbolic paraboloid, cone, elliptic cylinder, hyperbolic cylinder, parabolic cylinder, line, line at infinity, intersecting planes, parallel planes, plane, double plane, double plane at infinity and the whole space (a total of 20 types). The reader can refer to [32] for a detailed information about correspondence between inertia values and quadric surface types. An extended information about quadric types is also available in [35].

Given two quadrics Q_S and Q_T with matrices S and T, an important concept is a *pencil* of quadrics:

Definition 7. *The set*

$$\mathbf{R}(S, T) = \{\lambda S + \mu T\colon\ (\lambda, \mu) \in \mathbb{P}^1\}$$

is called the pencil of quadrics.

It can be shown that for any two distinct (coprime) quadrics from a pencil, their intersection is always the same. In other words, two distinct quadrics from a pencil define it uniquely. The equation $det(\mathbf{R}(S, T)) = 0$ is called the characteristic polynomial. The key idea comes from [28]. Assume, that $\mathbf{R}(S, T)$ is a pencil of two given quadrics. When the intersection is not degenerated, it is always possible to choose a quadric Q_U from the pencil, which is a ruled surface. It means that the surface can be parametrized by two linear coefficients. The next step is to plug the ruled quadric $Q_U\colon\ U \in \mathbf{R}(S, T)$ into one of the given quadrics. We obtain an equation with two unknown parameters, that we can solve and give the parametrized intersection curve. We omit technical details of this evaluation, providing only the final result.

Assume that we intersect a smooth quadric Q_S with a generic quadric Q_T. Dupont et al. [32] show that it is possible to choose a quadric Q_R from the pencil $\mathbf{R}(S, T)$, in terms of which, the intersection is given by the formula:

$$\Gamma_\epsilon(\xi) = \Gamma_R(\xi, \tau_\epsilon(\xi))$$
$$\Gamma_\epsilon(\xi) \in [\mathbb{Q}(\sqrt{\delta})[\xi, \sqrt{\Delta}]]^4$$

which yields homogeneous quadruplets. The first argument $\xi \in \mathbb{P}^1$ is a free argument, and the second argument is a zero of a second-degree polynomial $\gamma(\xi, \tau) = 0$. Since, there are two such solutions τ_ϵ, the formula consists of two arcs for

$\epsilon \in \{-1, 1\}$. In particular, we have:

$$\gamma(\xi, \tau) = a_2(\xi)\tau^2 + a_1(\xi)\tau + a_0(\xi) \quad (6)$$
$$\gamma(\xi, \tau) \in \mathbb{Q}(\sqrt{\delta})[\xi, \tau]$$

$$\tau_\epsilon(\xi) = (2a_0(\xi), -a_1(\xi) + \epsilon\sqrt{\Delta}) \quad (7)$$
$$\tau_\epsilon(\xi) \in \mathbb{Q}(\sqrt{\delta})[\xi, \sqrt{\Delta}]$$

and

$$\Delta(\xi) = a_1^2(\xi) - 4a_0(\xi)a_2(\xi)$$
$$\Delta(\xi) \in \mathbb{Q}(\sqrt{\delta})[\xi]$$

where $a_0, a_1, a_2 \in \mathbb{Q}(\sqrt{\delta})[\xi]$. These coefficients are obtained by plugging Q_R into Q_S or Q_T.

When the intersection is singular, Dupont et al. provide an extensive implementation for each case. All singular parametrizations are rational - they do not contain the square root. We omit all details, which are precisely discussed in [34]. In the following sections we assume, the intersection is given as $\Gamma(\xi)$ with $\xi \in \mathbb{P}^1$ for the rational case.

E. Spin-quadric and spin-QSIC intersection as a graph vertex

An important part of the new algorithm is the construction of graph vertices. The number of possible vertices is $O(n^3)$ for an arrangement of n surfaces. We use construction ideas from [36]. Assume that $\mathbf{s}_1(\xi)$ is the first intersection of spin-quadrics Q_S and Q_T, given in terms of Q_X from the their pencil. Let $\mathbf{s}_2(\xi)$ be the second intersection of spin-quadrics Q_U and Q_V given in terms of Q_Y. An intersection of two spin-QSIC is called a *spin-quadric surface intersection point(s)* (spin-QSIP). Note that we assume that the intersection can be made up from more than one point. In fact, there can be at most 8 points in one spin-QSIP. We also stress the fact, that the parameter ξ in each of the spin-QSICs is generally associated with two different spin-surfaces and thus is not related. From the previous section we know that the spin-QSIC can be smooth or rational. We have two cases, which we discuss separately:

- at least one of $\mathbf{s}_1(\xi)$ or $\mathbf{s}_2(\xi)$ is rational
- both $\mathbf{s}_1(\xi)$ and $\mathbf{s}_2(\xi)$ are smooth

Rational spin-QSIC

Without the loss of generality, we assume that $\mathbf{s}_1(\xi)$ is rational. We use that fact that the intersection of two curves is equal to the intersection of one of the curves with an arbitrary surface that the second curve lies on:

$$\begin{aligned}\mathbf{s}_1(\xi) \cap \mathbf{s}_2(\xi) &= \mathbf{s}_1(\xi) \cap (Q_U \cap Q_V) \\ &= \mathbf{s}_1(\xi) \cap Q_U = \mathbf{s}_1(\xi) \cap Q_V \\ &= \{\mathbf{s}_1(\xi) \colon \mathbf{s}_1(\xi)^T U \mathbf{s}_1(\xi) = 0\} \\ &= \{\mathbf{s}_1(\xi) \colon \mathbf{s}_1(\xi)^T V \mathbf{s}_1(\xi) = 0\}\end{aligned}$$

We can choose any of spin-quadrics Q_U, Q_V and write:

$$\begin{aligned}&\{\mathbf{s}_1(\xi) \colon \mathbf{s}_1(\xi)^T U \mathbf{s}_1(\xi) = 0\} = \\ &\{\mathbf{s}_1(\xi) \colon (\pm\frac{\Gamma_1(\xi)}{\|\Gamma_1(\xi)\|})^T U (\pm\frac{\Gamma_1(\xi)}{\|\Gamma_1(\xi)\|}) = 0\} = \\ &\{\mathbf{s}_1(\xi) \colon \Gamma_1(\xi)^T U \Gamma_1(\xi) = 0\}\end{aligned}$$

where $\Gamma_1(\xi)$ is the homogeneous intersection curve associated to $\mathbf{s}_1(\xi)$.

The matrix product $\Gamma_1(\xi)^T U \Gamma_1(\xi)$, is a univariate polynomial with the degree at most 8:

$$h_{\mathbf{s}_1 \cap \mathbf{s}_2}(\xi) = \Gamma_1(\xi)^T U \Gamma_1(\xi)$$

The polynomial, which zeroes define spin-QSIP points, will be called *a spin-QSIP determinant*.

Now, we define a spin-QSIP as:

$$\mathbf{s}_1(\xi) \cap \mathbf{s}_2(\xi) = \{\mathbf{s}_1(\xi) \colon h_{\mathbf{s}_1 \cap \mathbf{s}_2}(\xi) = 0\} \quad (8)$$

Smooth spin-QSIC

The case when two spin-QSICs are smooth is more involved. First, we make an observation that the spin-QSIP determinant is the same if we consider intersection of $\Gamma_1(\xi)$ and $\Gamma_2(\xi)$ instead of $\mathbf{s}_1(\xi)$ and $\mathbf{s}_2(\xi)$. Now, we follow a method proposed by Hemmer [36]. The method used for rational spin-QSIC does not work for smooth case because we obtain a spin-QSIP determinant with a square root. Since, we want to get a univariate determinant polynomial we use different method. We use the fact that:

$$\begin{aligned}(Q_S \cap Q_T) \cap Q_U &= (Q_S \cap Q_X) \cap Q_U \\ &= (Q_X \cap Q_S) \cap (Q_X \cap Q_U)\end{aligned}$$

so, it is possible to utilize the same parametrization on Q_X twice and get a common parameter space $(\xi, \tau) \in \mathbb{P}^1 \times \mathbb{P}^1$. In our practical implementation, we first intersect Q_S and Q_T to obtain the first intersection and Q_X quadric as a by-product. Then, we intersect Q_X with Q_U. Since, the Q_X is already a ruled quadric, the algorithm of Dupont et. al. chooses the Q_X as the parametrization and in result, we get the second intersection in the correct parametrization. From (6), we know that $\Gamma_1(\xi)$ is the zero set of the biquadratic polynomial

$$\begin{aligned}\gamma_1(\xi, \tau) &= \Gamma_1(\xi, \tau)^T X \Gamma_1(\xi, \tau) \\ &= a_2(\xi)\tau^2 + a_1(\xi)\tau + a_0(\xi)\end{aligned} \quad (9)$$

where

$$\tau_\epsilon(\xi) = (2a_0(\xi), -a_1(\xi) + \epsilon\sqrt{\Delta}) \quad (10)$$

The second curve $\Gamma_2(\xi)$ is the zero set of

$$\begin{aligned}\gamma_2(\xi, \tau) &= \Gamma_2(\xi, \tau)^T X \Gamma_2(\xi, \tau) \\ &= b_2(\xi)\tau^2 + b_1(\xi)\tau + b_0(\xi)\end{aligned} \quad (11)$$

A standard approach for solving multi-variable system of equations is a method of resultants. We eliminate τ variable and obtain:

$$\begin{aligned}res(\xi) &= resultant(\gamma_1, \gamma_2, \tau) \\ &= u_{02}(\xi)^2 - u_{01}(\xi)u_{12}(\xi)\end{aligned} \quad (12)$$

where $u_{ij} = a_i b_j - a_j b_i$. In case of a smooth spin-QSIC, the determinant polynomial is:

$$h_{\mathbf{s}_1 \cap \mathbf{s}_2}(\xi) = res(\xi)$$

where $res(\xi)$ is defined as in (12). There are at most 8 roots of $res(\xi)$. Because of the fact that spin-QSIC parametrization includes the square root, we have to discard all complex solutions (a negative value under the square root sign). Since, there are always two arcs of a smooth spin-QSIC, we include

the correct arc identifier $\epsilon = \pm 1$ along with each of the spin-QSIP points. We use the following theorem 8 from [37] to distinguish both complex intersections and correct arcs of spin-QSICs.

Theorem 2 (Hemmer)**.** *Let γ_1, τ_ϵ, γ_2 and $res \neq 0$ be defined as in the equations* (9)*,* (10)*,* (11) *and* (12) *respectively. And let ξ_0 denote a real root (if any) of res. Moreover, let $\tau_\epsilon(\xi)$ be a valid parametrization for ξ_0, that is, ξ_0 is not a root of $a_0(\xi)$.*

There are 3 cases:

1) $\Delta(\xi_0) < 0$*: ξ_0 corresponds to two complex intersection points*
2) $\Delta(\xi_0) = 0$*: ξ_0 corresponds to a real endpoint of both arcs*
3) $\Delta(\xi_0) > 0$*:*
   - *If $u_{01}(\xi_0) \neq 0$, then ξ_0 corresponds to one real intersection point on arc $\Gamma_\epsilon(\xi)$, with $\epsilon = -sign(u_{01})sign(2a_0u_{02} - a_1u_{01})_{|\xi_0}$*
   - *If $u_{01}(\xi_0) = 0$, then ξ_0 corresponds to two real points, one on each arc*

The next step is to locate real roots of spin-QSIP determinant.

Real root isolation

Since we are dealing with polynomials of the degree up to 8, we cannot use any formula for root computation. A commonly used approach is real root isolation. The basic principle is to give a list of non-overlapping intervals in which, there is a single root. The polynomial sign has to be opposite in both ends of the interval. In particular, it entails that the polynomial is square-free. We can not compute the exact root, but it is simple to refine root to an arbitrary floating point precision by the Descartes rule of signs (the bisection method). A classical description of this method gives Uspensky [38]. Hemmer [36] uses a variant of bisection method called *bitstream Descartes* by Eigenwillig et al. [39]. During the research, we came to the conclusion it is a bottleneck in our algorithm, thus we follow a different approach. There exist an algorithm of Akritas [40] which was shown in [41] to be far more efficient than the best known bisection method by Rouillier [42]. Akritas method is based on continued fractions. Due to a limited space, we will not describe the details of this method. An implementation is available in Mathemagix [43] in the *realroot* module. We extended the existing implementation to support coefficients of type $\mathbb{Q}(\sqrt{\delta})$. Our tests show that the Akritas method is about over a dozen times faster than the method of Uspensky and a few times faster than Rouillier's method.

III. Implementation and testing

A. New algorithm in contrast to Hemmer's algorithm

Hemmer [36] proposed a complete algorithm for computing an intersection graph of quadrics in $\mathbb{R}^3$. The algorithm uses QI library for quadric intersection. There are similarities and differences between our proposed algorithm and Hemmer's algorithm. It is presented in table I. Note: n is a number of quadrics in an arrangement.

Some of the procedures described in Hemmer algorithm exist in the new algorithm. Some of these were reimplemented with new algorithm to improve overall performance.

TABLE I
Comparision of the new algorithm and Hemmer's algorithm

Property	Motion planning in SO(3)	Hemmer
Topology	Spin(3)	$\mathbb{R}^3$
Dimension	4	3
Constraints	1	0
Objects	quadrics in Spin(3)	quadrics in $\mathbb{R}^3$
Running time	$O(n^3 log(n))$	$O(n^3 log(n))$

These include: an improved algorithm for polynomial root isolation and a method of determining a polynomial sign at a given point.

Currently, the implementation of Hemmer's algorithm is still not publicly available. Is is a part of Exacus project.

B. Triangle scenes or ball-only scenes

There are several reasons that ball-only scenes can be chosen instead of triangle scenes. In some usages a ball geometry is simply more natural than a triangulated one. We have tested various scenes: triangulated by design, ball-only scenes by design and scenes for which a sphere approximation was automatically generated by means of algorithm [44]. A general observation is that the complexity of the configuration space of ball-only scenes is usually lower. For a triangular scenes, there are a lot of small pieces of configuration space defined by every single $\mathcal{S}$ predicate. These fragments rarely overlap. Also, many surfaces are unnecessarily generated in triangle scenes.

We have collected the list of advantages of ball-only scenes over triangle scenes:

- much fewer predicates are needed, so the running times are better
- much fewer spin-surfaces are generated, because larger areas are defined by a single $\mathcal{BB}$ predicate
- duplicated surfaces are rare (by design)
- sphere-trees can be used to adaptive computations
- a $\mathcal{BB}$ predicate can enclose many $\mathcal{TT}$ predicates - it can be used as a faster method of approximation, but still triangles can be used as well
- it may simply be useful to describe a scene by balls only (for the safety of the mechanical tools used)

On the other hand, there are also a few disadvantages of using ball-only scenes:

- precise and mostly linear scenes are not suitable for ball-only scenes
- with a $\mathcal{BB}$ predicate, one can not design a flat geometry which is possible with $\mathcal{TT}$ predicates

Several good algorithms for sphere-tree construction have been developed, as in [45], [46] and [44]. The latter one, was used by us in our tests.

C. Implementation

All presented results were implemented in the author's library *libcs*. The library is designed in a generic fashion, so the most of underlying algorithms can be parametrized by different number types. The bottom layer of the library is CGAL [16] on top of which a spin kernel is introduced. For a number type we employ *bigint* type from LiDIA library [47], because the *libqi* [29] library requires that. Internally,

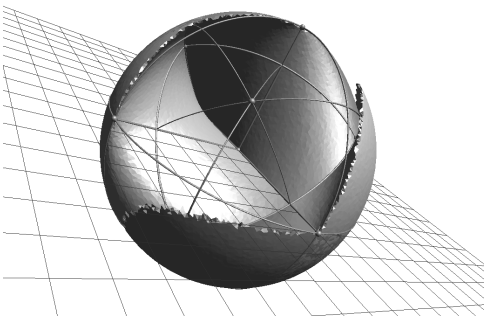

Fig. 8. An example projection of Spin(3) configuration space contents

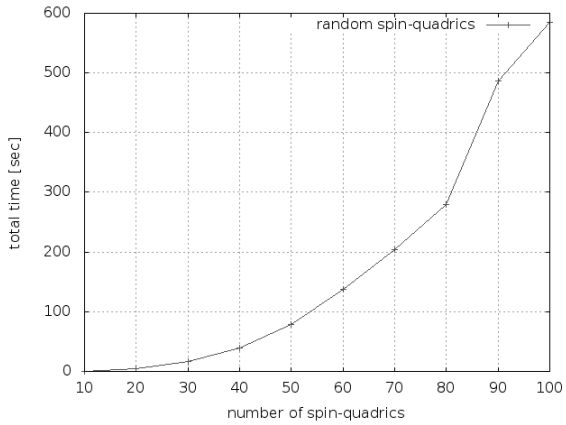

Fig. 9. *libcs* running times

LiDIA's *bigint* type is implemented with GMP [15] library, as most of similar solutions. Spin kernel extends CGAL's *Cartesian* kernel with a *Filtered kernel* on top of it.

The *libcs* class diagram is depicted in figure 7.

We also implemented a visualization application, which can display configuration spaces with their contents. An example configuration space is presented in figure 8.

During our research, we have tested many different scenes for the correctness of our implementation. At the same time, we have to note, our algorithm is not sensitive to any specific nature of a scene. Because of that, for the performance tests, we just use random scenes with the desired triangle or ball count. The running times are presented in plot 9. The implementation was tested on a Linux 3.2.0-3-amd64 SMP on a 2.40 GHz processor with 4 GB RAM and compiled with g++ 4.7.1.

D. The proposed algorithm is output-sensitive

We proved that it is possible to construct a simple test, indicating whether a pair of a stationary and a rotating triangle induce any geometry in the configuration space. As a result of this, our new algorithm is preceded by a this test for all triangle pairs, giving out a minimum list of predicates that we have to consider to go get the complete configuration space.

We start with the following lemma, for $\mathcal{TT}$ scenes:

Lemma 2 (Configuration space geometry)**.** *Triangle-triangle predicate or ball-ball predicate generate non-empty geometry in configuration space if and only if the intersection of the sweep volume of a rotating object f and a stationary object g is not empty:*

$$sweep(f) \cap g \neq \emptyset$$

Proof: If the sweep volume of rotating object intersects a stationary object, there exist a subset $U \in \mathrm{Spin}(3)$ of rotations $\mathscr{R}_s$ of the rotationg object that causes the intersection. The border ∂U is a part of the border of configuration space $\partial U \ \in \partial C_{free}$ and thus, is included. ■

Depending on scene type, we have two different tests for sweep volume intersection. In both, we use the previous lemma.

Proposition 6 ($\mathcal{TT}$ predicate inclusion test)**.** *A $\mathcal{TT}(K, L, M, A, B, C)$ predicate for a rotating triangle $f = ABC$ and a stationary triangle $g = KLM$ shall be included in configuration space computations if the following holds:*

$$intersect\_ball\_triangle(f, B_{max}(g)) \wedge$$
$$(g_K > r_{min}(f) \vee g_L > r_{min}(f) \vee g_M > r_{min}(f))$$

where

$$r_{min}(f) = min(\|f_A\|, \|f_B\|, \|f_C\|)$$
$$r_{max}(f) = max(\|f_A\|, \|f_B\|, \|f_C\|)$$
$$B_{min}(f) = \mathbb{B}(0, r_{min}(f))$$
$$B_{max}(f) = \mathbb{B}(0, r_{max}(f))$$

and $intersect\_ball\_triangle$ is a routine testing for a ball and a triangle intersection.

Proof: From lemma 2 we have:

$$sweep(f) \cap g \neq \emptyset$$

we can rewrite the above constraint as:

$$\begin{aligned}
sweep(f) \cap g \neq \emptyset &\iff \\
(B_{max}(f) \backslash B_{min}(f)) \cap g \neq \emptyset &\iff \\
((B_{max}(f) \cap g) \backslash (B_{min}(f) \cap g)) \neq \emptyset &\iff \\
(B_{max}(f) \cap g \neq \emptyset) \wedge \neg(B_{min}(f) \cap g \neq \emptyset) &\iff \\
(B_{max}(f) \cap g \neq \emptyset) \wedge (B_{min}(f) \cap g = \emptyset) &\iff \\
intersect_ball_triangle(g, B_{max}(f)) \wedge & \\
(g_K > r_{min}(f) \vee g_L > r_{min}(f) \vee g_M > r_{min}(f)) &
\end{aligned}$$

■

The routine $intersect\_ball\_triangle$ is taken from [48] and generalized to support arbitrary number type. All computations are performed in the number ring (no divisions and no square roots). In figure 10 a schematic view of $\mathcal{TT}$ inclusion test is depicted.

For a ball-only scene, we prove the following theorem:

Proposition 7 ($\mathcal{BB}$ predicate inclusion test)**.** *A $\mathcal{BB}(B, l, A, r)$ predicate for a rotating ball $f = \mathbb{B}(A, r)$ and a stationary ball $g = \mathbb{B}(B, l)$ shall be included in configuration space computations if the following holds:*

$$(lhs \leq 0) \vee (lhs^2 \leq rhs)$$

where

$$lhs := \|A\|^2 + \|B\|^2 - (r + l)^2$$
$$rhs := 4\|A\|^2\|B\|^2$$

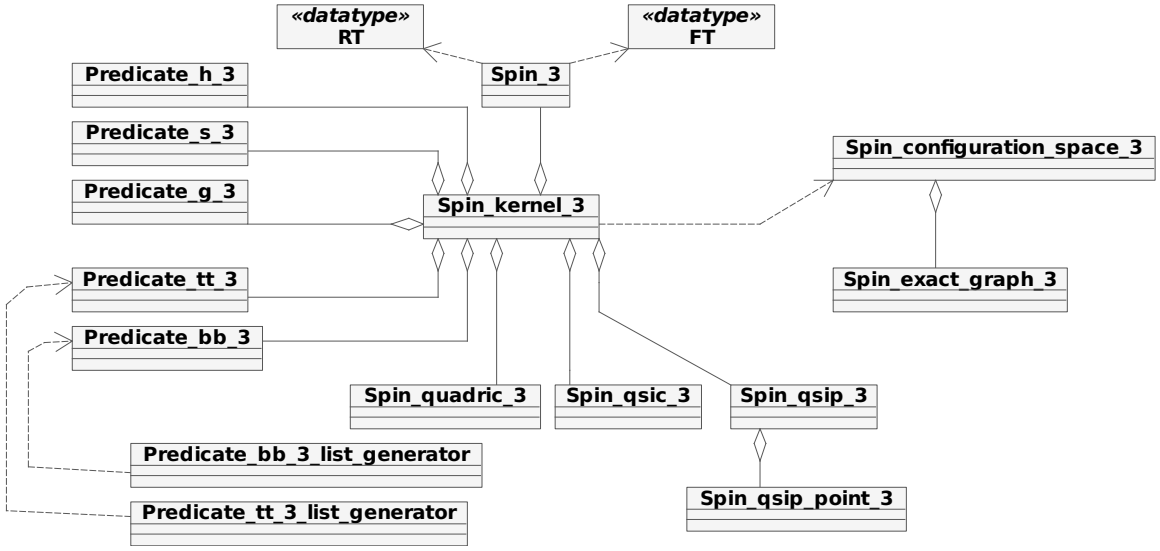

Fig. 7. *libcs* class diagram

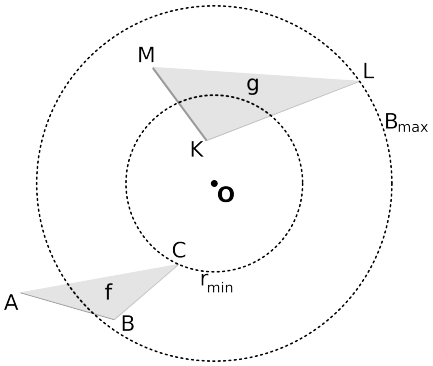

Fig. 10. $\mathcal{TT}$ predicate inclusion test

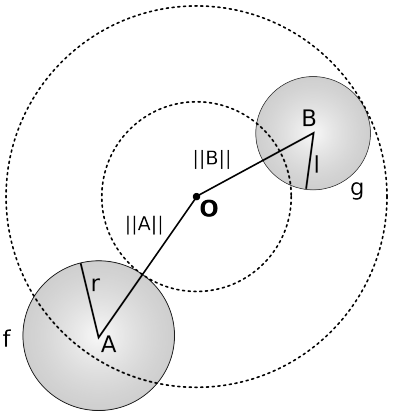

Fig. 11. $\mathcal{BB}$ predicate inclusion test

Proof: From lemma 2 we have:

$$sweep(f) \cap g \neq \emptyset$$

we can rewrite the above constraint as:

$$\begin{aligned}
sweep(f) \cap g \neq \emptyset &\iff \\
(\|A\| - r \leq \|B\| + l) \wedge (\|A\| + r \geq \|B\| - l) &\iff \\
(\|A\| - \|B\| \leq r + l) \wedge (\|A\| - \|B\| \geq -r - l) &\iff \\
(\|A\| - \|B\| \leq r + l) \wedge (-(\|A\| - \|B\|) \leq r + l) &\iff \\
|\|A\| - \|B\|| \leq r + l &\iff \\
\|A\|^2 + \|B\|^2 - 2\|A\|\|B\| \leq (r + l)^2 &\iff \\
\|A\|^2 + \|B\|^2 - (r + l)^2 \leq 2\|A\|\|B\| &\iff \\
lhs := \|A\|^2 + \|B\|^2 - (r + l)^2 & \\
rhs := 4\|A\|^2\|B\|^2 &
\end{aligned}$$

the last inequality holds iff $lhs \leq 0$ or $lhs^2 \leq rhs$. ■

Once again, in the test all computations are performed in the number ring (no divisions and no square roots). In figure 11 a schematic view of $\mathcal{BB}$ inclusion test is depicted.

IV. Conclusion and future work

We have shown that by using some latest algorithms related to 3D arrangement of quadrics, we are able to construct an exact configuration space for a motion involving 3D rotations. Such an algorithm allows one to construct motion planning algorithms of a new class, currently solved only in an approximate way. The algorithm can also be a part of more complex algorithm for motion planning with more than three degrees of freedom. The presented algorithm currently does not support the removal of duplicated features (edges and vertices), as in Hemmer's algorithm. It also does not support polyhedra with its interior included. We are going to address these additional features in our further research. Finally, it should be noted that it is possible to extend the structure of the configuration space by introducing cells in the arrangement. There has already been some progress done for arrangements of quadrics in $\mathbb{R}^3$. Examples include [49] and [50]. This task is one of the subjects of the author's further work.

References

[1] S. LaValle, *Planning algorithms*. Cambridge Univ Pr, 2006.
[2] J.-C. Latombe, *Robot motion planning*. Springer Verlag, 1990.
[3] P. Dobrowolski, "Exact Motion Planning with Rotations: the Case of a Rotating Polyhedron," in *Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering 2012*, London, U.K., 4-6 July 2012, pp. 792–797.
[4] G. Vegter, "The visibility diagram: a data structure for visibility problems and motion planning," *SWAT 90*, pp. 97–110, 1990.

[5] P. Agarwal, B. Aronov, and M. Sharir, "Motion planning for a convex polygon in a polygonal environment," *Discrete & Computational Geometry*, vol. 22, no. 2, pp. 201–221, 1999.

[6] F. Avnaim, J. Boissonnat, and B. Faverjon, "A practical exact motion planning algorithm for polygonal objects amidst polygonal obstacles," in *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*. IEEE, 1988, pp. 1656–1661.

[7] V. Koltun, "Pianos are not flat: Rigid motion planning in three dimensions," in *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2005, pp. 505–514.

[8] M. Foskey, M. Garber, M. Lin, and D. Manocha, "A Voronoi-based hybrid motion planner," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 1. IEEE, 2001, pp. 55–60.

[9] S. Hirsch and D. Halperin, "Hybrid motion planning: Coordinating two discs moving among polygonal obstacles in the plane," *Algorithmic Foundations of Robotics V*, pp. 239–256, 2004.

[10] D. Hsu, G. Sánchez-Ante, and Z. Sun, "Hybrid PRM sampling with a cost-sensitive adaptive strategy," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 3874–3880.

[11] L. Zhang, Y. Kim, and D. Manocha, "A hybrid approach for complete motion planning," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. IEEE, 2007, pp. 7–14.

[12] J. Lien, "Hybrid motion planning using Minkowski sums," *Proceedings of Robotics: Science and Systems IV*, 2008.

[13] L. Dorst and D. Fontijne, *Geometric algebra for computer science: an object-oriented approach to geometry*. Morgan Kaufmann, 2007.

[14] P. Lounesto, *Clifford algebras and spinors*. Cambridge Univ Pr, 2001, vol. 286.

[15] *GMP - The GNU Multiple Precision Arithmetic Library*. [Online]. Available: http://gmplib.org

[16] *CGAL - Computational Geometry Algorithms Library*. [Online]. Available: http://www.cgal.org

[17] J. Canny, *The complexity of robot motion planning*. The MIT Press, 1988.

[18] F. Thomas and C. Torras, "Interference detection between non-convex polyhedra revisited with a practical aim," in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*. IEEE, 1994, pp. 587–594.

[19] O. Devillers, P. Guigue *et al.*, "Faster triangle-triangle intersection tests," INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE, Tech. Rep., 2002.

[20] M. Held, "ERIT - A Collection of Efficient and Reliable Intersection Tests," *Journal of Graphics Tools*, vol. 2, pp. 25–44, 1998.

[21] T. Möller, "A Fast Triangle-Triangle Intersection Test," *Journal of Graphics Tools*, vol. 2, pp. 25–30, 1997.

[22] M. Henc and J. Porter-Sobieraj, "Parallel Decomposition of Three-dimensional Rotation Space," in *CS&P*, M. Szczuka, L. Czaja, A. Skowron, and M. Kacprzak, Eds. Pułtusk, Poland: Białystok University of Technology, 2011, pp. 206–214, electronic edition.

[23] L. Dupont, D. Lazard, S. Lazard, and S. Petitjean, "A new algorithm for the robust intersection of two general quadrics," in *Proc. 19th Annu. ACM Sympos. Comput. Geom*, 2003, pp. 246–255.

[24] Z. Xu, X. Wang, X. Chen, and J. Sun, "A robust algorithm for finding the real intersections of three quadric surfaces," *Computer aided geometric design*, vol. 22, no. 6, pp. 515–530, 2005.

[25] B. Mourrain, J. Técourt, and M. Teillaud, "On the computation of an arrangement of quadrics in 3D," *Computational Geometry*, vol. 30, no. 2, pp. 145–164, 2005.

[26] E. Berberich, M. Hemmer, L. Kettner, E. Schömer, and N. Wolpert, "An exact, complete and efficient implementation for computing planar maps of quadric intersection curves," in *Proceedings of the twenty-first annual symposium on Computational geometry*. ACM, 2005, pp. 99–106.

[27] S. Lazard, L. Peñaranda, and S. Petitjean, "Intersecting quadrics: An efficient and exact implementation," *Computational Geometry*, vol. 35, no. 1-2, pp. 74–99, 2006.

[28] J. Levin, "A Parametric Algorithm for Drawing Pictures of Solid Objects Composed of Quadric Surfaces." *Commun. ACM*, vol. 19, no. 10, pp. 555–563, 1976. [Online]. Available: http://dblp.uni-trier.de/db/journals/cacm/cacm19.html#Levin76

[29] *QI - Quadric intersection*. [Online]. Available: http://vegas.loria.fr/qi

[30] C. Tu, W. Wang, and J. Wang, "Classifying the nonsingular intersection curve of two quadric surfaces," in *Geometric Modeling and Processing, 2002. Proceedings*. IEEE, 2002, pp. 23–32.

[31] W. Wang, B. Joe, and R. Goldman, "Computing quadric surface intersections based on an analysis of plane cubic curves," *Graphical Models*, vol. 64, no. 6, pp. 335–367, 2002.

[32] L. Dupont, D. Lazard, S. Lazard, and S. Petitjean, "Near-optimal parameterization of the intersection of quadrics: I. The generic algorithm," *J. Symb. Comput.*, vol. 43, no. 3, pp. 168–191, 2008.

[33] ——, "Near-optimal parameterization of the intersection of quadrics: II. A classification of pencils," *J. Symb. Comput.*, vol. 43, no. 3, pp. 192–215, 2008.

[34] ——, "Near-optimal parameterization of the intersection of quadrics: III. Parameterizing singular intersections," *J. Symb. Comput.*, vol. 43, no. 3, pp. 216–232, 2008. [Online]. Available: http://dx.doi.org/10.1016/j.jsc.2007.10.007

[35] W. H. Beyer, "CRC Standard Mathematical Tables," Boca Raton, FL: CRC Press, Inc., Tech. Rep., 1987.

[36] M. Hemmer, L. Dupont, S. Petitjean, and E. Schömer, "A complete, exact and efficient implementation for computing the edge-adjacency graph of an arrangement of quadrics." *J. Symb. Comput.*, vol. 46, no. 4, pp. 467–494, 2011. [Online]. Available: http://dblp.uni-trier.de/db/journals/jsc/jsc46.html#HemmerDPS11

[37] M. Hemmer, "Exact computation of the adjacency graph of an arrangement of quadrics," Ph.D. dissertation, Johannes Gutenberg-UniversitÃt, 2008.

[38] J. V. Uspensky, *Theory of Equations*. McGraw-Hill, 1948.

[39] A. Eigenwillig, L. Kettner, W. Krandick, K. Mehlhorn, S. Schmitt, and N. Wolpert, "A Descartes Algorithm for Polynomials with Bit-Stream Coefficients." in *CASC*, ser. Lecture Notes in Computer Science, V. G. Ganzha, E. W. Mayr, and E. V. Vorozhtsov, Eds., vol. 3718. Springer, 2005, pp. 138–149. [Online]. Available: http://dblp.uni-trier.de/db/conf/casc/casc2005.html#EigenwilligKKMSW05

[40] A. Akritas, A. Bocharov, and A. Strzebonski, "Implementation of real root isolation algorithms in Mathematica," *Abstracts of the International Conference on Interval and Computer-Algebraic Methods in Science and Engineering (Interval'94)*, pp. 23–27, March 7–10 1994.

[41] A. Akritas and A. Strzebonski, "A comparative study of two real root isolation methods," *Nonlinear Analysis: Modelling and Control*, vol. 10(4), pp. 297–304, 2005.

[42] F. Rouillier and P. Zimmermann, "Efficient isolation of polynomial's real roots," *J. Comput. Appl. Math.*, vol. 162, no. 1, pp. 33–50, Jan. 2004. [Online]. Available: http://dx.doi.org/10.1016/j.cam.2003.08.015

[43] *Mathemagix - A free computer algebra system*. [Online]. Available: http://mathemagix.org

[44] G. Bradshaw and C. O'Sullivan, "Adaptive medial-axis approximation for sphere-tree construction." *ACM Trans. Graph.*, vol. 23, no. 1, pp. 1–26, 2004. [Online]. Available: http://dblp.uni-trier.de/db/journals/tog/tog23.html#BradshawO04

[45] P. M. Hubbard, "Collision Detection for Interactive Graphics Applications." *IEEE Trans. Vis. Comput. Graph.*, vol. 1, no. 3, pp. 218–230, 1995. [Online]. Available: http://dblp.uni-trier.de/db/journals/tvcg/tvcg1.html#Hubbard95

[46] I. J. Palmer and R. L. Grimsdale, "Collision Detection for Animation using Sphere-Trees." *Comput. Graph. Forum*, vol. 14, no. 2, pp. 105–116, 1995. [Online]. Available: http://dblp.uni-trier.de/db/journals/cgf/cgf14.html#PalmerG95

[47] *LiDIA: A C++ Library For Computational Number Theory*. [Online]. Available: http://www.cdc.informatik.tu-darmstadt.de/TI/LiDIA

[48] C. Ericson, *Optimizing a sphere-triangle intersection test*. [Online]. Available: http://realtimecollisiondetection.net/blog/?p=103

[49] N. Geismann, M. Hemmer, and E. Schömer, "Computing a 3-dimensional Cell in an Arrangement of Quadrics: Exactly and Actually!" in *Proceedings of the seventeenth annual symposium on Computational geometry*. ACM, 2001, pp. 264–273.

[50] E. Schömer and N. Wolpert, "An exact and efficient approach for computing a cell in an arrangement of quadrics." *Comput. Geom.*, vol. 33, no. 1-2, pp. 65–97, 2006. [Online]. Available: http://dblp.uni-trier.de/db/journals/comgeo/comgeo33.html#SchomerW06