Calculating the rigid rotations that move three vectors orthogonal to three fixed vectors

H. Bekker

Department of Mathematics and Computing Science, University of Groningen, P.O.B. 800, 9700 AV Groningen, The Netherlands, Email: bekker@cs.rug.nl

Abstract

In $\mathbb{R}^3$ three vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$, and three vectors $\mathbf{k}, \mathbf{l}, \mathbf{m}$ are given. In this article a method and an algorithm are presented to calculate the rigid rotations $\mathbf{R}$ that rotate the vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ so that the vector $\mathbf{Ra}$ is orthogonal to $\mathbf{k}$, $\mathbf{Rb}$ is orthogonal to $\mathbf{l}$ and $\mathbf{Rc}$ is orthogonal to $\mathbf{m}$. The geometry of the problem is discussed and numerical issues treated. It is shown that the number of solutions is $0, 2, 4, 6$ or 8.

Key words: linear algebra, spherical trigonometry, rigid rotation, alignment, inscribed spherical triangle.

1 Introduction

Consider in $\mathbb{R}^3$ three vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ and three vectors $\mathbf{k}, \mathbf{l}, \mathbf{m}$. We want to calculate the real proper orthogonal transformation $\mathbf{R}$ with the property

$$\mathbf{k}.\mathbf{Ra} = 0, \quad \mathbf{l}.\mathbf{Rb} = 0, \quad \mathbf{m}.\mathbf{Rc} = 0. \tag{1}$$

The matrix $\mathbf{R}$ represents a *proper orthogonal transformation* when it holds that $\Sigma_i \; R_{ij}R_{ik} = \delta_{jk}, \quad j, k = 1, 2, 3$ and $det(\mathbf{R}) = 1$. A proper orthogonal transformation is sometimes called a *rigid rotation* because it does not change the dimensions or shape of the transformed object.

In this article we are only interested in solving problem instances with a finite number of solutions. This set of problems is called the set of *acceptable problems*. The condition that the number of solutions is finite does not strongly limit the set of acceptable problems, many degenerate problem instances have a finite number of solutions. Here, *degenerate* means that $\mathbf{a}, \mathbf{b}, \mathbf{c}$ are not linearly independent but coplanar, or that two of the vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ are linearly dependent. In the same way the vectors $\mathbf{k}, \mathbf{l}, \mathbf{m}$ may be degenerate. In spite of the fact that there are, based on the type of degeneracy, nine types of problems, the method we present is a generic method, i.e. it handles all problem instances in the same way.

As the problem in (1) is about Two Vector Triples, in the following we call this problem the TVT problem. As will be shown later (1) can be written as a set of three second degree polynomial equations in three variables. By applying a preconditioning technique we transform this set of equations into a set of two polynomial equations in two variables. These and other techniques transform the problem to a simpler problem and reduce numerical complications. This is relevant because, besides implementing the method in a computer algebra system, for the sake of speed we implement the method in a conventional programming language, in our case C++.

The TVT problem is an instance of the more general problem

$$\mathbf{k}.\mathbf{R}\mathbf{a} = \alpha_1, \quad \mathbf{l}.\mathbf{R}\mathbf{b} = \alpha_2, \quad \mathbf{m}.\mathbf{R}\mathbf{c} = \alpha_3 \tag{2}$$

where $\alpha_1, \alpha_2, \alpha_3$ are real numbers. The assumption that (1) is solved, not (2), makes it possible to transform the problem into a simpler problem. As will become clear in the following subsection, for our goal it is enough to solve (1) and therefore we will not further consider (2).

In the rest of this section it is shown how the TVT problem is related to aligning convex polyhedra, the TVT problem is visualized, three reformulations are given and the unacceptable problems are listed. In section two the problem is preconditioned and in section three two polynomial equations are derived. Finally the implementation and results are discussed.

1.1 Problem motivation

The reason that we solve the TVT problem has to do with our research on calculating the similarity measure of two convex polyhedra A and B in 3D. In [1] a similarity measure for convex polyhedra is proposed, based on Minkowski addition. It is calculated by evaluating the volume or mixed volume of the Minkowski sum of two convex polyhedra A and B over a specific set of relative orientations of A and B. The relative orientations are characterized by the fact that faces and edges of A and B are are maximally aligned, i.e. are parallel as much as possible. Representing the direction of an edge e by the vector $\mathbf{e}$, and representing the orientation of face f by its normal vector $\mathbf{f}$, edge e is parallel to face f when it holds that $\mathbf{e}.\mathbf{f} = \mathbf{0}$. In 3D a rigid rotation $\mathbf{R}$ is defined by three parameters, so a maximum alignment of A and B can be calculated by choosing three edges e_1, e_2, e_3 of A (or B) and three faces f_1, f_2, f_3 of B (or A), and to calculate for the corresponding vectors $\mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3}, \mathbf{f_1}, \mathbf{f_2}, \mathbf{f_3}$ a rigid rotation $\mathbf{R}$ with the property

$$\mathbf{f_1}.\mathbf{R}\mathbf{e_1} = 0, \quad \mathbf{f_2}.\mathbf{R}\mathbf{e_2} = 0, \quad \mathbf{f_3}.\mathbf{R}\mathbf{e_3} = 0. \tag{3}$$

In this way alignments with parallel faces are also found: when $\mathbf{e_1}$ and $\mathbf{e_2}$ both lie in face f of A, and f_1 equals f_2 , then A and B are aligned so that face f of A is parallel to face f_1 of B.

To calculate the similarity of A and B, for all edge triples and all face triples all maximum alignments of A and B are calculated and for every of these alignments the similarity functional is evaluated. The similarity measure of A and B is the maximum of the similarity functional over all maximum alignments of A and B. The number of combinations of three edges and three faces of two tetrahedra, even when the geometry of the tetrahedra is used to skip unfeasible cases, is of the order 100..1000 [henk axel]. For more complex polyhedra this number is much larger. This shows that the TVT algorithm and its implementation should be efficient.

In this article we are not concerned with polyhedra and calculating similarity measures, we concentrate on solving (1).

1.2 Some properties of the TVT problem

The following simple properties of the TVT problem will be used.

-The lengths of the vectors in (1) does not influence the solutions $\mathbf{R}$, the only condition is that the lengths are non-zero. In the following it is assumed that the vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{k}, \mathbf{l}, \mathbf{m}$ have been scaled to unit length. The sign of the vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{k}, \mathbf{l}, \mathbf{m}$ does not matter. That is because when $\mathbf{k}.\mathbf{Ra} = \mathbf{0}$ it also holds that $\mathbf{k}.\mathbf{R}(-\mathbf{a}) = \mathbf{0}$, $-\mathbf{k}.\mathbf{Ra} = \mathbf{0}$ and $-\mathbf{k}.\mathbf{R}(-\mathbf{a}) = \mathbf{0}$. A similar statement holds for $\mathbf{l}.\mathbf{Rb} = \mathbf{0}$ and $\mathbf{m}.\mathbf{Rc} = \mathbf{0}$.

-Instead of moving the vector triple $\mathbf{a}, \mathbf{b}, \mathbf{c}$ the vector triple $\mathbf{k}, \mathbf{l}, \mathbf{m}$ may be moved, that is, we could solve the *inverse problem*

$$\mathbf{a}.\tilde{\mathbf{R}}\mathbf{k} = 0, \quad \mathbf{b}.\tilde{\mathbf{R}}\mathbf{l} = 0, \quad \mathbf{c}.\tilde{\mathbf{R}}\mathbf{m} = 0. \tag{4}$$

Obviously, $\tilde{\mathbf{R}} = \mathbf{R}^{-1}$, hence, for a given problem instance the number of solutions of (4) and (1) is the same.

-The relative orientation of the triple $\mathbf{a}, \mathbf{b}, \mathbf{c}$ w.r.t. the triple $\mathbf{k}, \mathbf{l}, \mathbf{m}$ does not influence the number of solutions of a TVT problem.

1.3 Reformulations

The TVT problem may be reformulated as a problem on the unitsphere and as a restricted projective geometry problem. Let us start with the reformulation on the unitsphere centered at the origin. The vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ are represented by three points a, b, c on the unitsphere. Point a is the the intersection of the unitsphere with the halfline from the origin in the direction of $\mathbf{a}$, or, more crudely, a is the endpoint of the vector $\mathbf{a}$. The points b and c are defined analogously. The vectors $\mathbf{k}, \mathbf{l}, \mathbf{m}$ are represented by three great circles k, l, m on the unitsphere. The great circle k is the polar circle of the point on the sphere defined by the vector $\mathbf{k}$, thus, k is the intersection of the sphere with the plane through the origin perpendicular to the vector $\mathbf{k}$. The great circles l and m are defined analogously. Denoting the spherical triangle defined by a, b, c as $\triangle(a, b, c)$, we can reformulate the problem as: calculate the proper orthogonal rotations $\mathbf{R}$ that move $\triangle(a, b, c)$ so that a lies on the great circle k, b lies on l and c lies on m, see figure 1.

The TVT problem may also be formulated as a problem in the projective plane using restricted projective transformations. Let us explain. In plane projective geometry objects (e.g. points and lines) in the plane are are represented

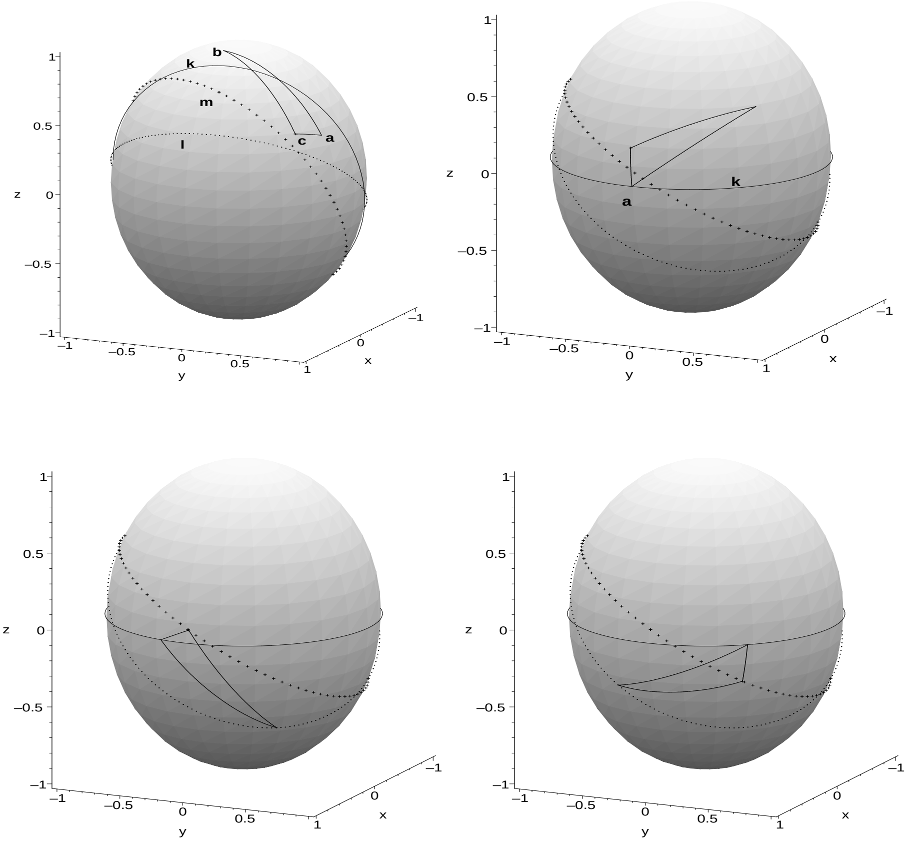

Fig. 1. In section (2) it is explained how the TVT problem may be represented on the sphere.
Top left: a TVT problem instance. The problem is to find the rotations $\mathbf{R}$ that move the spherical triangle $\triangle(a, b, c)$ so that a lies on the great circle k, b lies on l and c lies on m.
Top right: the preconditioned TVT problem. The vector $\mathbf{a}$ is in the direction of the x axis and the great circle k is in the x, y plane. Thus, $\mathbf{a}$ lies on k.
Bottom: two solutions of the preconditioned problem. The great circles k, l, m remain in the preconditioned position and $\triangle(a, b, c)$ has moved so that $\mathbf{a}$ lies on k, $\mathbf{b}$ lies on l and $\mathbf{c}$ lies on m. The problem shown here has four solutions. The two other solutions are not visible because then $\triangle(a, b, c)$ is at the backside of the sphere.

as objects in 3D. In 3D space these objects are transformed by a linear transformation and the resulting objects are backprojected to the plane. In conventional projective geometry linear transformations in 3D are used. We now restrict the linear transformations to proper orthogonal transformations in 3D and call this geometry *rigid projective geometry.* The problem can now be stated as follows: in the projective plane three points a, b, c are given and three lines k, l, m. Calculate the rigid projective transformations $\mathbf{R}$ that map a onto k, b onto l and c onto m.

The analogous problem in the Euclidean plane is as follows. Consider in the Euclidean plane three lines k, l, m, and a triangle with vertices a, b, c, see figure 2. Find the Euclidean transformations $\mathbf{R}$ that move the triangle so that

$$\mathbf{R}a \text{ lies on } k, \qquad \mathbf{R}b \text{ lies on } l, \qquad \mathbf{R}c \text{ lies on } m.$$

In [2] it is shown that this problem has at most two solutions. The Euclidean problem in the plane is a limit case of the TVT problem.

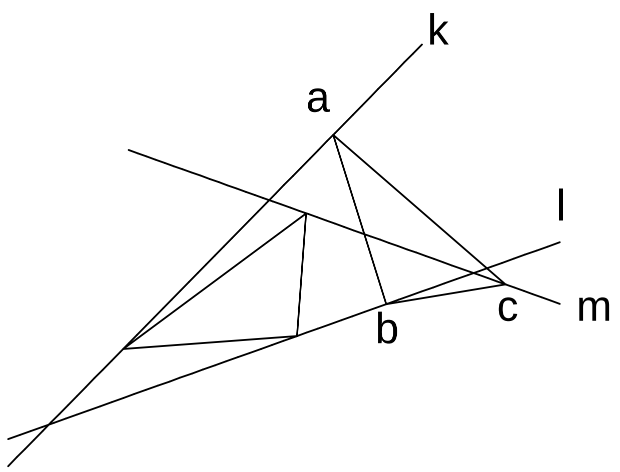

Fig. 2. The Euclidean geometry problem related to the TVT problem. Three lines k, l, m are given and three points a, b, c. To be better visible the points are connected by three line segments, so the points a, b, c are the vertices of $\triangle(a, b, c)$. The problem is to find the Euclidean transformations, that is, shift- translate transformations, that move $\triangle(a, b, c)$ so that a lies on k, b lies on l and c lies on m. This problem instance has two solutions. Both positions of $\triangle(a, b, c)$ are shown. The left $\triangle(a, b, c)$ is inscribed in in the triangle defined by k, l, m.

Yet another reformulation of the TVT problem can be given, again on the sphere. A curve cc on the sphere is defined as the locus of the point c of $\triangle(a, b, c)$ while the point a coincides with the great circle k and the point b coincides with the great circle l, see figure 3. Solving the TVT problem is equivalent with calculating the intersections of cc with the great circle m.

The reformulations on the sphere are useful to visualize the TVT problem and the reformulation in the projective plane shows how the TVT problem is related to a known problem in the Euclidean plane. However, the reformulations do not help to solve the TVT problem. To solve (1) we approach the problem in an algebraic way.

In the literature the TVT problem is not mentioned, not in the form of (1) nor as reformulation.

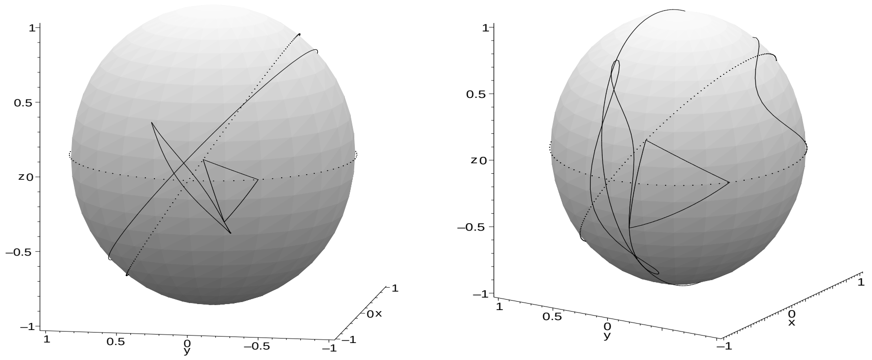

Fig. 3. The TVT problem may be defined in terms of calculating the intersection of a curve cc on the sphere with the great circle m (line). The curve cc is the locus of the point c of $\triangle(a, b, c)$ while the point a coincides with the great circle k (sparsely doted), and the point b coincides with the great circle l (densely dotted). In general cc consists of two branches, one on each side of the sphere. Two different problem instances are shown. In the left figure only one branch is visible, intersecting the great circle m twice. In the right figure one branch is completely visible, intersecting the great circle m six times and part of the other branch is visible intersecting m at the backside of the sphere twice. So this case has eight solutions.

2 Special problems and unacceptable problems

2.1 Some special problems

Many special problem types of the TVT problem exist, some of them are degenerate and some are general and yet special. Here we want to give an impression of the many shapes of the TVT problem in a visual way. In figure 1 a general problem instance with four solutions was shown. A problem instance with eight solutions was shown in figure 3b. One branch of cc intersects m six times, the other branch (not visible) intersects m two times.

In figure 4 the solutions of two degenerate problem instances are shown. In the left figure the vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ are coplanar and the vectors $\mathbf{k}, \mathbf{l}, \mathbf{m}$ are coplanar. One of four solutions is shown. In the right figure the most degenerate acceptable problem type is shown. The vectors $\mathbf{a}$ and $\mathbf{c}$ coincide, and the vectors $\mathbf{l}$ and $\mathbf{m}$ coincide. One of four solutions is shown.

Finally, in figure 5 two almost identical problem instances are shown, the left one is unacceptable, the right one is acceptable. In both cases the great circles k and l coincide and the points a and b do not coincide. The angle between the vectors $\mathbf{a}$ and $\mathbf{c}$ is $\pi/2$ and the angle between the vectors $\mathbf{b}$ and $\mathbf{c}$ is $\pi/2$. As a result the curve cc consists of two opposite points. In the left figure $\mathbf{m} = (1, -3, 0)$, so cc lies on the great circle m, thus the number of solutions is infinite. In the right figure $\mathbf{m} = (1, -3, -0.2)$, so cc lies not on the great circle m, thus the number of solutions is finite, in this case zero.

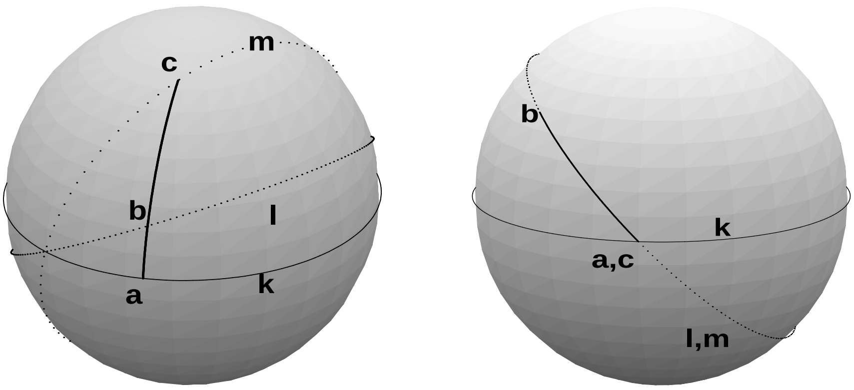

Fig. 4. Two degenerate problem instances. Left: the vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ are coplanar and the vectors $\mathbf{k}, \mathbf{l}, \mathbf{m}$ are coplanar. One of four solutions is shown. Right: the most degenerate acceptable problem type. The vectors $\mathbf{a}$ and $\mathbf{c}$ coincide, and the vectors $\mathbf{l}$ and $\mathbf{m}$ coincide. One of four solutions is shown.

2.2 Unacceptable problems

To prevent the TVT algorithm from breaking down on unacceptable problem instances they should be detected timely. In appendix A unacceptable TVT problems are derived in a systematic way, here only the results are given.

Unacceptable problems are characterized by specific combinations of coplanar vectors, perpendicular vectors and parallel vectors, where in this context parallel means in the same or opposite direction. As mentioned before, the relative orientation of the triple $\mathbf{a}, \mathbf{b}, \mathbf{c}$ w.r.t. the triple $\mathbf{k}, \mathbf{l}, \mathbf{m}$ does not influence whether a problem is acceptable or not, only the relative orientation of the vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ and the relative orientation of the vectors $\mathbf{k}, \mathbf{l}, \mathbf{m}$ has to be considered.

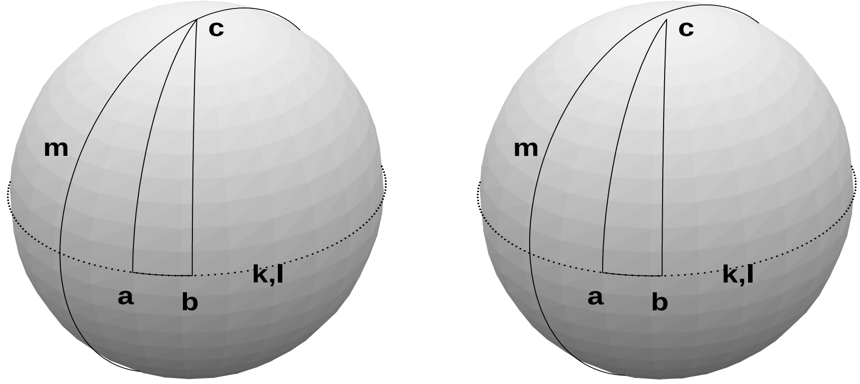

Fig. 5. Two almost identical problem instances, the left one is unacceptable, the right one is acceptable but has zero solutions. In both cases the great circles k and l lie in the $x-y$ plane and coincide. The points a and b lie on k and l and do not coincide. Because $\mathbf{c}=(0,0,1)$ the angle between the vectors $\mathbf{a}$ and $\mathbf{c}$ is $\pi/2$ and the angle between the vectors $\mathbf{b}$ and $\mathbf{c}$ is $\pi/2$. In the left figure $\mathbf{m}=(1,-3,-0)$ so any rotation of $\triangle(a,b,c)$ around the z-axis leaves $\mathbf{a},\mathbf{b},\mathbf{c}$ perpendicular to $\mathbf{k},\mathbf{l},\mathbf{m}$ respectively, thus this problem is unacceptable. In the right figure $\mathbf{m}=(1,-3,-0.2)$, thus when a and b lie on k and l the point c never lies on m, which makes the problem acceptable with zero solutions.

In order to represent unacceptable cases the following notation is used. When all three vectors $\mathbf{a},\mathbf{b},\mathbf{c}$ are in the same direction this is represented as $\mathbf{a}\|\mathbf{b}\|\mathbf{c}$. When the vectors $\mathbf{a}$ and $\mathbf{b}$ are parallel and they are both perpendicular to $\mathbf{c}$ this is represented as $(\mathbf{a}\|\mathbf{b})\perp\mathbf{c}$. When $\mathbf{b}$ is perpendicular to $\mathbf{a}$, and $\mathbf{c}$ is perpendicular to $\mathbf{a}$ this is represented as $\mathbf{b}\perp\mathbf{a}\wedge\mathbf{c}\perp\mathbf{a}$. The situation that $\mathbf{a},\mathbf{b},\mathbf{c}$ are coplanar is represented by $copl(\mathbf{a},\mathbf{b},\mathbf{c})$.

Using this representation the unacceptable cases are as follows, see table 1. The case shown in figure 5a corresponds with [2,2,1] in this table.

1,1,1	$\mathbf{a}\|\mathbf{b}\|\mathbf{c}$ and $copl(\mathbf{k},\mathbf{l},\mathbf{m})$
1,1,2	$(\mathbf{a}\|\mathbf{b})\perp\mathbf{c}$ and $\mathbf{k}\perp\mathbf{m}\wedge\ \mathbf{l}\perp\mathbf{m}$
1,2,1	$(\mathbf{a}\|\mathbf{c})\perp\mathbf{b}$ and $\mathbf{k}\perp\mathbf{l}\wedge\mathbf{m}\perp\mathbf{l}$
1,2,2	$(\mathbf{b}\|\mathbf{c})\perp\mathbf{a}$ and $\mathbf{l}\perp\mathbf{k}\wedge\ \mathbf{m}\perp\mathbf{k}$
2,1,1	$\mathbf{b}\perp\mathbf{a}\wedge\ \mathbf{c}\perp\mathbf{a}$ and $(\mathbf{l}\|\mathbf{m})\perp\mathbf{k}$
2,1,2	$\mathbf{a}\perp\mathbf{b}\wedge\ \mathbf{c}\perp\mathbf{b}$ and $(\mathbf{k}\|\mathbf{m})\perp\mathbf{l}$
2,2,1	$\mathbf{a}\perp\mathbf{c}\wedge\mathbf{b}\perp\mathbf{c}$ and $(\mathbf{k}\|\mathbf{l})\perp\mathbf{m}$
2,2,2	$copl(\mathbf{a},\mathbf{b},\mathbf{c})$ and $\mathbf{k}\|\mathbf{l}\|\mathbf{mc}$

Table 1. Unacceptable problem types

From table 1 it can be seen that, apart from permutations and problem inversion, there are actually only two unacceptable problem types. Case $[1, 1, 1]$ is the inverse of problem $[2, 2, 2]$. The cases $[1, 1, 2], [1, 2, 1]$ and $[1, 2, 2]$ are identical, apart from a permutation of the vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ and a corresponding permutation of the vectors $\mathbf{k}, \mathbf{l}, \mathbf{m}$. Similarly, the cases $[1, 1, 2], [1, 2, 1]$ and $[1, 2, 2]$ are identical, apart from a permutation of the vectors. Finally, the cases $[1, 1, 2], [1, 2, 1]$ and $[1, 2, 2]$ are inverse cases of $[1, 1, 2], [1, 2, 1]$ and $[1, 2, 2]$ respectively. Thus only two classes of unacceptable problems remain, the class consisting of $[1, 1, 1]$ and $[2, 2, 2]$, and the class consisting of $[1, 1, 2], [1, 2, 1], [1, 2, 2], [1, 1, 2], [1, 2, 1]$ and $[1, 2, 2]$. This makes it easy to implement checking for unacceptable problem instances. Only two types of checks have to be implemented and these are used to test all permuted and inverted instances of the problem at hand.

3 Preconditioning

After aborting unacceptable problems, the first two steps of the TVT algorithm are preconditioning techniques. The first step, if required, consists of a permutation of $\mathbf{a}, \mathbf{b}, \mathbf{c}$ and a corresponding permutation of $\mathbf{k}, \mathbf{l}, \mathbf{m}$. In the second preconditioning step the triples $\mathbf{a}, \mathbf{b}, \mathbf{c}$ and $\mathbf{k}, \mathbf{l}, \mathbf{m}$ are moved to well-chosen randomized positions. The goal of preconditioning is to keep the the algorithm simple, in spite of the various problem types.

3.1 Permutations of $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{k}, \mathbf{l}, \mathbf{m}$

The algorithm solving the TVT problem is implemented in a procedure $tvt(...)$ and it is called as $tvt(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{k}, \mathbf{l}, \mathbf{m}, \mathbf{R})$. The order of the equations in (1) does not matter, thus the solutions $\mathbf{R}$ are invariant under any permutation of $\mathbf{a}, \mathbf{b}, \mathbf{c}$ and a corresponding permutation of $\mathbf{k}, \mathbf{l}, \mathbf{m}$. This property is used to create a situation where $\mathbf{a}, \mathbf{b}$ are not linearly dependent and $\mathbf{k}, \mathbf{l}$ are not linearly dependent. Thus, after aborting unacceptable problem instances, if required, the values of the variables $\mathbf{a}, \mathbf{b}, \mathbf{c}$ are permuted and the values of the variables $\mathbf{k}, \mathbf{l}, \mathbf{m}$ are permuted correspondingly, so that $\mathbf{a}, \mathbf{b}$ are linearly independent and $\mathbf{k}, \mathbf{l}$ are linearly independent. This is always possible, problems instances for which this is not possible have been aborted at an earlier stage. In fact, because we are only interested in the angle between $\mathbf{a}$ and $\mathbf{b}$ not all six permutations of $\mathbf{a}, \mathbf{b}, \mathbf{c}$ have to be considered but only the three cyclic shifts of $\mathbf{a}, \mathbf{b}, \mathbf{c}$.

3.2 The preconditioning transformations S and T

We will now see how the solutions of two problem instances $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{k}, \mathbf{l}, \mathbf{m}$ and $\mathbf{a}', \mathbf{b}', \mathbf{c}', \mathbf{k}', \mathbf{l}', \mathbf{m}'$ are related, assuming that the primed vectors $\mathbf{a}', \mathbf{b}', \mathbf{c}', \mathbf{k}', \mathbf{l}', \mathbf{m}'$ are related to the unprimed vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{k}, \mathbf{l}, \mathbf{m}$ by two proper orthogonal transformations $\mathbf{S}$ and $\mathbf{T}$ as

$$\mathbf{a} = \mathbf{S}\mathbf{a}', \quad \mathbf{b} = \mathbf{S}\mathbf{b}', \quad \mathbf{c} = \mathbf{S}\mathbf{c}' \tag{5}$$

$$\mathbf{k} = \mathbf{T}\mathbf{k}', \quad \mathbf{l} = \mathbf{T}\mathbf{l}', \quad \mathbf{m} = \mathbf{T}\mathbf{m}'. \tag{6}$$

Assume that we want to solve

$$\mathbf{k}'.\mathbf{R}'\mathbf{a}' = 0, \quad \mathbf{l}'.\mathbf{R}'\mathbf{b}' = 0, \quad \mathbf{m}'.\mathbf{R}'\mathbf{c}' = 0. \tag{7}$$

and assume that we have been able to find a rotation $\mathbf{R}$ fulfilling (1). Using (5) and (6) to substitute $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{k}, \mathbf{l}, \mathbf{m}$ in (1) gives

$$\mathbf{k}'.\mathbf{T}^{-1}\mathbf{R}\mathbf{S}\mathbf{a}' = 0 \quad \mathbf{k}'.\mathbf{T}^{-1}\mathbf{R}\mathbf{S}\mathbf{b}' = 0 \quad \mathbf{k}'.\mathbf{T}^{-1}\mathbf{R}\mathbf{S}\mathbf{c}' = 0$$

Thus, by solving (1) we obtain the solution of (7), where the solution is given by

$$\mathbf{R}' = \mathbf{T}^{-1}\mathbf{R}\mathbf{S}.$$

The proper orthogonal transformations $\mathbf{S}$ and $\mathbf{T}$ are completely free, hence they may be chosen so that the triples $\mathbf{a}, \mathbf{b}, \mathbf{c}$ and $\mathbf{k}, \mathbf{l}, \mathbf{m}$ are positioned in such a way that the equations in (1) are simplified. Now we choose $\mathbf{S}$ so that $\mathbf{a}$ is oriented along the positive $\mathbf{x}$ axis, and $\mathbf{T}$ is chosen so that $\mathbf{k}$ is oriented along the positive $\mathbf{z}$ axis, see figure 1. Thus, more explicit, the vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{k}, \mathbf{l}, \mathbf{m}$ are given by

$$\mathbf{a} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix}, \mathbf{c} = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \end{pmatrix}, \mathbf{k} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \mathbf{l} = \begin{pmatrix} l_0 \\ l_1 \\ l_2 \end{pmatrix}, \mathbf{m} = \begin{pmatrix} m_0 \\ m_1 \\ m_2 \end{pmatrix} \tag{8}$$

Notice that $\mathbf{a}$ and $\mathbf{k}$ are orthogonal.

The condition that $\mathbf{a}$ is in the positive x-direction and $\mathbf{k}$ is in the positive z-direction does not completely determine $\mathbf{S}$ and $\mathbf{T}$. Later, by choosing the directions of $\mathbf{b}$ and $\mathbf{l}$, the transformations $\mathbf{S}$ and $\mathbf{T}$ will be defined completely.

4 Calculating rotations

4.1 The Rodrigues matrix

Every orthogonal transformation may be considered as a rotation through some angle around an axis through the origin. We represent this rotation axis by the vector

$$\mathbf{r} = \begin{pmatrix} u \\ v \\ w \end{pmatrix}.$$

The length $r \equiv |\mathbf{r}|$ of this vector is used to define the angle θ of the rotation around this axis as

$$\tan(\frac{\theta}{2}) = r.$$

Accordingly to Rodrigues [3] this rotation may be represented by a matrix

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

Hence, a rotation $\mathbf{R}_x$ around the $\mathbf{x}$ axis is given by

$$\mathbf{R}_x = (\frac{1}{1+u^2}) \begin{pmatrix} 1+u^2 & 0 & 0 \\ 0 & 1-u^2 & -2u \\ 0 & 2u & 1-u^2 \end{pmatrix} \tag{9}$$

and a rotation $\mathbf{R}_z$ around the $\mathbf{z}$ axis is given by

$$\mathbf{R}_z = (\frac{1}{1+w^2}) \begin{pmatrix} 1-w^2 & -2w & 0 \\ 2w & 1-w^2 & 0 \\ 0 & 0 & 1+w^2 \end{pmatrix}. \tag{10}$$

(9) and (10) may be combined to a sequence of rotations $\mathbf{R}_{zx}$, that is, first a rotation around the $\mathbf{x}$ axis followed by a rotation around the $\mathbf{z}$ axis. The combined rotation is given by

$$\mathbf{R}_{zx} = \frac{1}{(1+u^2)(1+w^2)} \begin{pmatrix} (1-w^2)(1+u^2) & -2w(1-u^2)) & 4uw \\ 2w(1+u^2) & (1-w^2)(1-u^2) & -2u(1-w^2) \\ 0 & 2u(1+w^2) & (1+w^2)(1-u^2) \end{pmatrix}. \tag{11}$$

Earlier we chose $\mathbf{S}$ and $\mathbf{T}$ so that $\mathbf{a}$ is perpendicular to $\mathbf{k}$. The rotation $\mathbf{R}_{zx}$ has the property that $\mathbf{k}.\mathbf{R}_{zx}\mathbf{a}= 0$ for every $\mathbf{R}_{zx}$, so, the vector $\mathbf{a}$ transformed by $\mathbf{R}_{zx}$ remains perpendicular to $\mathbf{k}$. I.e., the first equation in (1) is fulfilled for every rotation $\mathbf{R}_{zx}$. Therefore we only have to consider the second and third equation in (1). Moreover, the rotation matrix only depends on u and w. In this way, without loss of generality, three non-linear equations (1) in three variables are transformed into two non-linear equation in two variables.

Summarizing, our approach to solve the TVT problem consists of two steps. First the triples $\mathbf{a}, \mathbf{b}, \mathbf{c}$ and $\mathbf{k}, \mathbf{l}, \mathbf{m}$ are moved to well-chosen positions. Then the triple $\mathbf{a}, \mathbf{b}, \mathbf{c}$ is rotated around the x-axis, followed by a rotation around the z-axis. The parameters u and w represent the angles of these rotations.

The Rodrigues representation of a proper orthogonal transformation is closely related to the quaternion representation. However, instead of four variables in the quaternion representation, the Rodrigues representation is defined by three variables. A disadvantage of the Rodrigues representation is that, in order to represent a rotation over $\pm\pi$, one or more components of $\mathbf{r}$ go to infinity, i.e. $|r| \to \infty$. As will be shown later, we avoid this situation by a randomization technique.

4.2 Deriving the equations and randomization

Assuming that no rotations over $\pm\pi$ occur the scalar factor $\frac{1}{(1+u^2)(1+w^2)}$ in (11) is non-zero. As it only influences the lengths of the transformed vectors, not their directions, this factor has not to be taken into account. Substituting (11) without this factor in the last two equations of (1), and substituting the vectors $\mathbf{b}, \mathbf{c}, \mathbf{l}, \mathbf{m}$ given in (8) gives two polynomial equations in u and w, see figure 6 for an example,

$$(-l0\ b0 + l1\ b1 - l2\ b2)\, u^2 w^2 + (2\, l1\ b0 + 2\, l0\ b1)\, u^2 w + (2\, l1\ b2 + 2\, l2\ b1)\, u w^2$$
$$+ (-l1\ b1 - l2\ b2 + l0\ b0)\, u^2 + 4\, l0\ b2\ uw + (l2\ b2 - l0\ b0 - l1\ b1)\, w^2$$
$$+ (-2\, l1\ b2 + 2\, l2\ b1)\, u + (2\, l1\ b0 - 2\, l0\ b1)\, w + l0\ b0 + l1\ b1 + l2\ b2 = 0 \quad (12)$$

and

$$(-m0\ c0 + m1\ c1 - m2\ c2)\, u^2 w^2 + (2\, m1\ c0 + 2\, m0\ c1)\, u^2 w + (2\, m1\ c2 + 2\, m2\ c1)\, u w^2$$
$$+ (-m1\ c1 - m2\ c2 + m0\ c0)\, u^2 + 4\, m0\ c2\ uw + (m2\ c2 - m0\ c0 - m1\ c1)\, w^2$$
$$+ (-2\, m1\ c2 + 2\, m2\ c1)\, u + (2\, m1\ c0 - 2\, m0\ c1)\, w + m0\ c0 + m1\ c1 + m2\ c2 = 0 \quad (13)$$

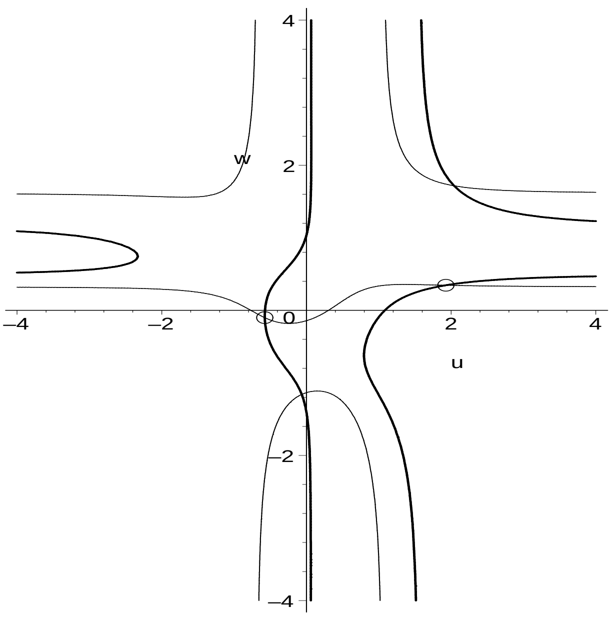

Fig. 6. Equation (12) and (13) (bold) for the problem instance shown in figure 1. In order to make the graph more compact the scaling is non-linear. It can be seen that the curves intersect four times which corresponds with four solutions. Two intersections are marked with a circle. These solutions are shown in figure 1.

From (12) and (13) two univariate equations are derived, one polynomial equation in u, denoted as *eqnu*, and one polynomial equation in w, denoted as *eqnw*. The equations *eqnu* and *eqnw* are derived by eliminating the variable w respectively u from (12) and (13). The computer algebra system MAPLE [maple] performs this task by calculating the determinant of the Sylvester matrix of (12) and (13). This gives two eighth degree polynomial equations

$$polu \equiv f_0 + f_1 u + f_2 u^2 + f_3 u^3 + f_4 u^4 + f_5 u^5 + f_6 u^6 + f_7 u^7 + f_8 u^8 = 0 \quad (14)$$

and

$$polw \equiv g_0 + g_1 w + g_2 w^2 + g_3 w^3 + g_4 w^4 + g_5 w^5 + g_6 w^6 + g_7 w^7 + g_8 w^8 = 0. \quad (15)$$

In these equations the coefficients $f_0 ... f_8$ and $g_0 ... g_8$ are polynomials in $b_0, b_1, b_2, c_0, c_1, c_2, l_0, l_1, l_2, m_0, m_1, m_2$ and thus are real-valued. (14) and (15) are both of degree eight. That means that the system of equations (12) and (13) has eight solutions in $\mathbb{C}$. Because the coefficients of (14) and (15) are real valued the solutions of (12) and (13) will be real valued or occur in complex conjugate pairs.

Remark: to gain more insight we tried to calculate the Groebner basis of (12) and (13). Current computer algebra systems did not succeed. Possibly that is because (12) and (13) are too general, all problem types, non-degenerate and degenerate, are represented by (12) and (13).

In the foregoing it was assumed that no solutions $u = \pm\infty$ or $w = \pm\infty$ occur. This condition is fulfilled when *eqnu* and *eqnw* are monic, i.e. when in *eqnu* $f_8 \neq 0$ and in *eqnw* $g_8 \neq 0$. We accomplish this by determining the direction of $\mathbf{b}$ and $\mathbf{l}$ by a randomization process. Let us explain this for *eqnu*. In the foregoing it was decided that $\mathbf{a}$ should be in the direction of the positive x-axis, so the orientation of $\mathbf{b}$ and $\mathbf{c}$ around the x-axis is free. We now choose this orientation angle randomly. When for this orientation f_8 proves to be zero, the vectors $\mathbf{b}, \mathbf{c}$ are rotated by an additional random angle around the x-axis. Because the parameter u represents the rotation of $\mathbf{b}, \mathbf{c}$ around the x-axis, as a result all roots of *eqnu* are shifted, so, very probably, now $f_8 \neq 0$. When still $f_8 = 0$ the randomization is repeated until $f_8 \neq 0$. In a similar way, by rotating $\mathbf{l}, \mathbf{m}$ randomly around the z-axis *polw* is made monic. In the following subsection the randomization is discussed in more detail.

4.3 Randomization and calculating S and T

Let us first see how the rigid rotation $\mathbf{S}$ is determined that moves the vectors $\mathbf{a}', \mathbf{b}', \mathbf{c}'$ so that $\mathbf{a} = \mathbf{S}\mathbf{a}', \quad \mathbf{b} = \mathbf{S}\mathbf{b}', \quad \mathbf{c} = \mathbf{S}\mathbf{c}'$, where $\mathbf{a}$ coincides with the positive x-axis and the orientation of $\mathbf{b}, \mathbf{c}$ around the x-axis is randomly chosen. To that end first a local frame consisting of the orthogonal unitvectors $\mathbf{d}_{x'}$, $\mathbf{d}_{y'}$, $\mathbf{d}_{z'}$ is constructed around $\mathbf{a}', \mathbf{b}', \mathbf{c}'$. The vector $\mathbf{d}_{x'}$ is chosen so that it coincides with $\mathbf{a}'$. The vector $\mathbf{d}_{y'}$ is chosen so that it is perpendicular to $\mathbf{d}_{x'}$, so that it is in the plane defined by $\mathbf{a}', \mathbf{b}'$, and so that the inner product $\mathbf{b}'.\mathbf{d}_{y'} > 0$. This is always possible because the angle between $\mathbf{a}'$ and $\mathbf{b}'$ is non-zero. Finally, $\mathbf{d}_{z'}$ is defined by $\mathbf{d}_{z'} = \mathbf{d}_{x'} \times \mathbf{d}_{y'}$.

Another local frame $\mathbf{d}_x$, $\mathbf{d}_y$, $\mathbf{d}_z$ is constructed, around the the (not yet existing) vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$. The vector $\mathbf{d}_x$ is chosen so that it coincides with the positive x-axis. The vector $\mathbf{d}_y$ is chosen so that it is perpendicular to $\mathbf{d}_x$ and has a random orientation around $\mathbf{d}_x$, so, $\mathbf{d}_y$ is in the y-z plane. Finally, $\mathbf{d}_z$ is defined by $\mathbf{d}_z = \mathbf{d}_x \times \mathbf{d}_y$.

From the vectors $\mathbf{d}_{x'}$, $\mathbf{d}_{y'}$, $\mathbf{d}_{z'}$ a matrix $\mathbf{M}$ is constructed as

$$M = (\mathbf{d}_{x'}, \mathbf{d}_{y'}, \mathbf{d}_{z'})$$

thus, the columns of $\mathbf{M}$ consist of the vectors $\mathbf{d}_{x'}$, $\mathbf{d}_{y'}$, $\mathbf{d}_{z'}$. Similarly from the vectors $\mathbf{d}_x$, $\mathbf{d}_y$, $\mathbf{d}_z$ a matrix $\mathbf{N}$ is constructed

$$N = (\mathbf{d}_x, \mathbf{d}_y, \mathbf{d}_z).$$

Using the property that for an orthogonal matrix $\mathbf{M}$ it holds that $\mathbf{M}^{-1}$=$\mathbf{M}^T$ it is easily checked that the linear transformation $\mathbf{S}$ mapping the frame $\mathbf{d}_{x'}$, $\mathbf{d}_{y'}$, $\mathbf{d}_{z'}$ on the frame $\mathbf{d}_x$, $\mathbf{d}_y$, $\mathbf{d}_z$ is given by

$$S = NM^T.$$

Now the vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ are constructed with (5).

In a similar way $\mathbf{T}$ is calculated, that is, first two local frames $\mathbf{e}_{x'}$, $\mathbf{e}_{y'}$, $\mathbf{e}_{z'}$ and $\mathbf{e}_x$, $\mathbf{e}_y$, $\mathbf{e}_z$ are constructed. The vector $\mathbf{e}_{x'}$ coincides with $\mathbf{k'}$. The vector $\mathbf{e}_{y'}$ is chosen so that it is perpendicular to $\mathbf{e}_{x'}$, so that it is in the plane defined by $\mathbf{k'}, \mathbf{l'}$, and so that the inner product $\mathbf{l'}.\mathbf{e}_{y'} > 0$. The vector $\mathbf{e}_{z'}$ is defined by $\mathbf{e}_{z'} = \mathbf{e}_{x'} \times \mathbf{e}_{y'}$. The vector $\mathbf{e}_x$ is chosen so that it coincides with the positive z-axis. The vector $\mathbf{e}_y$ is chosen so that it is perpendicular to $\mathbf{e}_x$ and has a random orientation around $\mathbf{e}_x$, so, $\mathbf{e}_y$ is in the x-y plane. Finally, $\mathbf{d}_z$ is defined by $\mathbf{d}_z = \mathbf{d}_x \times \mathbf{d}_y$.

From the vectors $\mathbf{e}_{x'}$, $\mathbf{e}_{y'}$, $\mathbf{e}_{z'}$ a matrix $\mathbf{O}$ is constructed

$$O = (\mathbf{e}_{x'}, \mathbf{e}_{y'}, \mathbf{e}_{z'})$$

and from the vectors $\mathbf{e}_x$, $\mathbf{e}_y$, $\mathbf{e}_z$ a matrix $\mathbf{P}$ is constructed

$$P = (\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z).$$

The linear transformation $\mathbf{T}$ mapping the frame $\mathbf{e}_{x'}$, $\mathbf{e}_{y'}$, $\mathbf{e}_{z'}$ on the frame $\mathbf{e}_x$, $\mathbf{e}_y$, $\mathbf{e}_z$ is given by

$$T = PO^T.$$

The vectors $\mathbf{k}, \mathbf{l}, \mathbf{m}$ are constructed with (6).

4.4 Solving the equations

As was explained in section 1.1 our reason to solve the TVT problem is that we want to align two polyhedra, so, in principle we are only interested in real solutions of the system of equations (12) and (13). However, most degenerate problem instances have multiple roots with multiplicity up to four. The roots with even multiplicity are very sensitive to noise in the input data, so are easily missed. This is intolerable. Therefore we calculate all eight complex roots of the system (14), (15) and select the real roots and the roots with a small complex part.

By aborting unacceptable problem instances and by preconditioning the problem, the system of equations (12), (13) is guaranteed to be well behaved, that is, it has eight solutions in $\mathbb{C}$ and it is not too close to degeneracy. Therefore, solving (12), (13) with a computer algebra system gives no problems. However, solving (12), (13) in C++ with a numerical library specialized in solving systems of polynomial equations proves to be non-trivial. To overcome these problems we devised a new method to solve small systems of polynomial equations. This method will be explained briefly later.

Remark: the reason that solving the system of equations (12), (13) is not always trivial is that for some problem instances one of these equations becomes degenerate, that is, it reduces to $0 = 0$.

4.5 The complete TVT algorithm

AlgorithmTVT Calculate $\mathbf{R}'$ so that $\mathbf{k}'.\mathbf{R}'\mathbf{a}' = 0,\ \mathbf{l}'.\mathbf{R}'\mathbf{b}' = 0,\ \mathbf{m}'.\mathbf{R}'\mathbf{c}' = 0$

if unacceptable-problem (see section1.3) **then**
  exit(Error: unacceptable problem instance)
end if
while $\angle(\mathbf{a}', \mathbf{b}') == 0$ or $\angle(\mathbf{k}', \mathbf{l}') == 0$ **do**
  cyclic_shift($\mathbf{a}', \mathbf{b}', \mathbf{c}'$)
  cyclic_shift($\mathbf{k}', \mathbf{l}', \mathbf{m}'$)
end while
repeat
  generate-randomized($\mathbf{S}$) generate-randomized($\mathbf{T}$)
  $\mathbf{a} = \mathbf{S}\mathbf{a}'$ $\mathbf{b} = \mathbf{S}\mathbf{b}'$ $\mathbf{c} = \mathbf{S}\mathbf{c}'$
  $\mathbf{k} = \mathbf{T}\mathbf{k}'$ $\mathbf{l} = \mathbf{T}\mathbf{l}'$ $\mathbf{c} = \mathbf{T}\mathbf{m}'$
  calculate the coefficients of *eqnu* and *eqnw*
until *eqnu* is monic and *eqnw* is monic
solve *eqnu* and *eqnw* for u and w
select real solutions
calculate $\mathbf{R}_{zx}$
calculate $\mathbf{R}'$

Comments on the TVT algorithm

- In section 1.3 a list of unacceptable cases was given for the unprimed vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ and $\mathbf{k}, \mathbf{l}, \mathbf{m}$. Because the absolute orientation of $\mathbf{a}, \mathbf{b}, \mathbf{c}$ and of $\mathbf{k}, \mathbf{l}, \mathbf{m}$ does not influence the acceptability, and the relative orientation of $\mathbf{a}, \mathbf{b}, \mathbf{c}$ w.r.t. $\mathbf{k}, \mathbf{l}, \mathbf{m}$ does not influence the acceptability, in the TVT algorithm the acceptability is tested for primed vectors.

– In principle, instead of calculating in the repeat loop all of the coefficients of *eqnu* and *eqnw*, it is sufficient to calculate f_8 and g_8. However, the probability that a rerandomization is required is very small, and therefore it is more efficient to calculate all coefficients right away.

5 Implementation, tests, results and discussion

First we implemented the TVT algorithm in MAPLE and used it for visualization and experiments. It works well but takes in the order of one second to solve a single TVT problem instance.

To achieve a higher performance we implemented the algorithm in C++. All floating point numbers are represented with 64 bit precession, i.e. the DOUBLE type of C++. Initially the SYNAPS C++ package for solving systems of polynomial equations was used, however, a small but significant number of solutions was missed by SYNAPS and SYNAPS was slow. Therefore we designed our own method to obtain the roots of the system (12), (13), designated the CORS method [6]. CORS proved to be robust and efficient. All further experiments with the C++ implementation of TVT were performed using the CORS method.

Essence of the CORS method is that first all eight solutions of (14) are calculated and all eight solutions of (15). For this we use Laguerre's method [4]. These solutions are combined to 64 solutions. From these solutions, with a combinatorial optimization technique, eight optimal roots are selected. The name CORS stands for "Combinatorial Optimization Root Selection".

The error of a solution is defined as

$$err = \sqrt{(\mathbf{k}'.\mathbf{R}'\mathbf{a}')^2 + (\mathbf{l}'.\mathbf{R}'\mathbf{b}')^2 + (\mathbf{m}'.\mathbf{R}'\mathbf{c}')^2}.$$

In the C++ implementation two constants are used. The first constant sa represents a small angle. It is used to define unacceptable problem instances, as given in section 1.3, in the following way. Two unitvectors $\mathbf{a}, \mathbf{b}$ are called linearly dependant when $\angle(\mathbf{a}, \mathbf{b}) \leq sa$ or $\angle(-\mathbf{a}, \mathbf{b}) \leq sa$. Analogously, two unitvectors $\mathbf{a}, \mathbf{b}$ are called perpendicular when $|\angle(\mathbf{a}, \mathbf{b}) - \pi/2| \leq sa$. Experiments showed that the value $sa = 10^{-7}$ is optimal in the sense that a smaller angle leads to unstable numerical calculations and a larger angle has the effect that problem instances with well-defined solutions are aborted. The second constant, denoted as cy, which stands for complexity, determines which complex valued solutions are acceptable. More precise, when the pair u, w is a solution of the system of equations (12), (13), where u or w is complex valued, it is considered as a solution when $|arg(u)| \leq cy$ and $|arg(w)| \leq cy$. In that case the real part of u and w is used for further calculations. Experiments showed that the value $cy = 10^{-4}$ is optimal in the sense that a smaller value may result in the loss of multiple roots and a larger value introduces unacceptable errors.

In section 4.3 it was explained that the vector $\mathbf{d}_y$ is perpendicular to the x axis an that it is given a random direction in the y-z plane. In our implementation we do not generate during runtime random vectors $\mathbf{d}_y$ but use a precomputed fixed list with 10 vectors in the y-z plane with a random orientation. In the same way the random vector $\mathbf{e}_y$ is chosen from a fixed list.

We tested the TVT algorithm extensively on various problem types and various problem dimensions. Nine types of problems are generated, that is, the vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ may be independent or coplanar or two of the vectors may be linearly dependent. The vectors $\mathbf{k}, \mathbf{l}, \mathbf{m}$ are generated with the same types of degeneracies. Combining these cases gives nine types of problems. In our random problem generator the range of the angles between the vectors can be controlled. In this way the method is tested on a range of problem dimensions. We use three ranges. In large angle problems the angle between the vectors is large, which means that it ranges from $\frac{1}{100}\pi$ to $\frac{99}{100}\pi$. The angles may also be medium, which means that they range from $10^{-4}\pi$ to $10^{-2}\pi$, and from $(1-10^{-2})\pi$ to $(1-10^{-4})\pi$. Small angles range from 0 to $10^{-4}\pi$, and from $(1-10^{-4})\pi$ to π.

Results For all problem types and dimension the method works accurate, stable and efficient. Acceptable problems are handled correctly and unacceptable problems are aborted. The number of solutions is always even and the error is small. As an example, we ran 10^5 problem instances where the angles between $\mathbf{a}, \mathbf{b}, \mathbf{c}$ are large and the angles between $\mathbf{k}, \mathbf{l}, \mathbf{m}$ are large. In the table below it is listed how many problems had ns solutions. This shows that, as expected, no problem has an uneven number of solutions.

number of solutions	0	1	2	3	4	5	6	7	8
nr. of problems	10581	0	9684	0	75616	0	3321	0	798

For these problems, less than one in a thousand problems requires rerandomization. Thus the fixed lists with 10 random vectors are adequate. The average error of these solutions is 6.5 10^{-11}. For problems with a small dimensions the error is on average 3.8 10^{-19}.

One of our goals was to devise a generic method, that is, a method that handles all problem types in the same way, irrespective of the type of degeneracy of the problem. We tested the method extensively on degenerate problem types and did not encounter any problem. Below the possible number of solutions is listed for some combinations of degenerate triples $\mathbf{a}, \mathbf{b}, \mathbf{c}$ and $\mathbf{k}, \mathbf{l}, \mathbf{m}$.

	l. i. $\mathbf{a}, \mathbf{b}, \mathbf{c}$	copl. $\mathbf{a}, \mathbf{b}, \mathbf{c}$	$(\mathbf{a} \parallel \mathbf{b}) \vee (\mathbf{a} \parallel \mathbf{c}) \vee (\mathbf{b} \parallel \mathbf{c})$	$\mathbf{a} \parallel \mathbf{b} \parallel \mathbf{c}$
l.i. $\mathbf{k}, \mathbf{l}, \mathbf{m}$	0,2,4,6,8	0,4,8	0,4	0
copl. $\mathbf{k}, \mathbf{l}, \mathbf{m}$	0,4,8	4	4	∞
$(\mathbf{k} \parallel \mathbf{l}) \vee (\mathbf{k} \parallel \mathbf{m}) \vee (\mathbf{l} \parallel \mathbf{m})$	0,4	4	4	∞
$\mathbf{k} \parallel \mathbf{l} \parallel \mathbf{m}$	0	∞	∞	∞

Table 2.
The possible number of solutions of the TVT problem for combinations of degenerate triples $\mathbf{a}, \mathbf{b}, \mathbf{c}$ and $\mathbf{k}, \mathbf{l}, \mathbf{m}$. The abbriviation l.i. means linearly independant and copl. means coplanar. In this table unacceptable problems are not considered.

Finally we tested the TVT implementation on our initial problem, i.e. aligning convex polyhedra. The general and degenerate problem instances occurring in this application were handled without problems.

The speed of the C++ implementation on a 1Ghz. PC. is roughly 1000 calls of $tvt()$ per second.

6 Conclusion

When (1) is directly formulated as a set of three polynomial equations, without considering unacceptable problem instances, roots at infinity, double and quadruple roots, and without preconditioning, it leads to a complicated implementation and unstable numerical calculations. When, on the other hand, the problem is preconditioned properly, only two polynomial equations are left. Also, by randomizing the preconditioning, rotations over $\pm\pi$ are avoided and thus the Rodrigues representation of a rotation may be used. By a permutation of $\mathbf{a}, \mathbf{b}, \mathbf{c}$ and a corresponding permutation of $\mathbf{k}, \mathbf{l}, \mathbf{m}$ the algorithm can handle all problem types in the same way, which simplifies the implementation significantly. After taking these precautions and using a robust polynomial solver, solving the TVT problem is without problems. The resulting C++ implementation is generic, stable and efficient. It may be used to calculate alignments of convex polyhedra and to calculate inscribed spherical triangles.

7 Appendix A

In this appendix we identify unacceptable TVT problems in a systematic way and show that there are eight types of unacceptable problems. Unacceptable problem instances may be detected in two ways, geometrically and algebraically. The first option means that the geometry of the triples $\mathbf{a}, \mathbf{b}, \mathbf{c}$ and $\mathbf{k}, \mathbf{l}, \mathbf{m}$ is analyzed, the second option means that polynomial equations are analyzed. We have chosen the first option because then it is more clear which cases are aborted. Unacceptable problems are characterized by specific combinations of coplanar vectors, perpendicular vectors and parallel vectors, where in this context parallel means in the same or opposite direction.

Let us first define three situations.

I $\mathbf{a}$ is perpendicular to $\mathbf{k}$.
II $\mathbf{a}$ is perpendicular to $\mathbf{k}$ and $\mathbf{b}$ is perpendicular to $\mathbf{l}$.
III $\mathbf{a}$ is perpendicular to $\mathbf{k}$, $\mathbf{b}$ is perpendicular to $\mathbf{l}$ and $\mathbf{c}$ is perpendicular to $\mathbf{m}$.

Now we consider situation III. When there are infinitely many proper orthogonal transformations leaving this situation invariant we are dealing with an unacceptable problem instance. It is well known that any proper orthogonal transformation may be written as a rotation around an axis ρ over some angle. So an unacceptable problem instance is characterized by the fact that there is an axis ρ about which $\mathbf{a}, \mathbf{b}, \mathbf{c}$ may be rotated over an infinite number of angles, leaving

III invariant. Our approach to identify unacceptable problem instances is to look for these axes, first for the pair $\mathbf{a}, \mathbf{k}$ then for the pairs $\mathbf{a}, \mathbf{k}$ and $\mathbf{b}, \mathbf{l}$ and finally for $\mathbf{a}, \mathbf{k}$, $\mathbf{b}, \mathbf{l}$ and $\mathbf{c}, \mathbf{m}$.

Consider the vectors $\mathbf{a}$ and $\mathbf{k}$ and assume that I is fulfilled. Then there are two axes leaving I invariant, see figure 7.1 and 7.2.

1 The axis ρ_1, coinciding with $\mathbf{a}$. Rotating $\mathbf{a}$ around ρ_1 does not move $\mathbf{a}$ so I is still fulfilled.

2 The axis ρ_2 coinciding with $\mathbf{k}$. Rotating $\mathbf{a}$ around ρ_2 leaves $\mathbf{a}$ perpendicular to $\mathbf{k}$ so I is still fulfilled.

Now the vectors $\mathbf{b}$ and $\mathbf{l}$ are added and we assume that II is fulfilled. There are four possible positions of $\mathbf{b}$ leaving II invariant under an infinite number of rotations. Each of the previous cases branches in two cases, see figure 7.1,1...7.2,2.

1,1 ρ_1, coincides with $\mathbf{a}$ and $\mathbf{b}$ coincides with ρ_1.
1,2 ρ_1, coincides with $\mathbf{a}$ and $\mathbf{b}$ is perpendicular to ρ_1.
2,1 ρ_2 coincides with $\mathbf{k}$ and $\mathbf{b}$ coincides with ρ_2.
2,2 ρ_2 coincides with $\mathbf{k}$ and $\mathbf{b}$ is perpendicular to ρ_2.

Finally the vectors $\mathbf{c}$ and $\mathbf{m}$ are added and we assume that III is fulfilled. For each of the previous four cases there are two positions of $\mathbf{c}$ leaving III invariant under an infinite number of rotations. This gives eight cases, see figure 7.1,1,1...7.2,2,2.

1,1,1 ρ_1 coincides with $\mathbf{a}$, $\mathbf{b}$ coincides with ρ_1, $\mathbf{c}$ coincides with ρ_1.
1,1,2 ρ_1 coincides with $\mathbf{a}$, $\mathbf{b}$ coincides with ρ_1, $\mathbf{c}$ is perpendicular to ρ_1.
1,2,1 ρ_1 coincides with $\mathbf{a}$, $\mathbf{b}$ is perpendicular to ρ_1, $\mathbf{c}$ coincides with ρ_1.
1,2,2 ρ_1 coincides with $\mathbf{a}$, $\mathbf{b}$ is perpendicular to ρ_1, $\mathbf{c}$ is perpendicular to ρ_1.
2,1,1 ρ_2 coincides with $\mathbf{k}$, $\mathbf{b}$ coincides with ρ_2, $\mathbf{c}$ coincides with ρ_2.
2,1,2 ρ_2 coincides with $\mathbf{k}$, $\mathbf{b}$ coincides with ρ_2, $\mathbf{c}$ is perpendicular to ρ_2.
2,2,1 ρ_2 coincides with $\mathbf{k}$, $\mathbf{b}$ is perpendicular to ρ_2, $\mathbf{c}$ coincides with ρ_2.
2,2,2 ρ_2 coincides with $\mathbf{k}$, $\mathbf{b}$ is perpendicular to ρ_2, $\mathbf{c}$ is perpendicular to ρ_2.

This may be represented in a more compact notation. In case [1,1,1] all three vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ are in the same direction which we represent by $\mathbf{a}\|\mathbf{b}\|\mathbf{c}$. In case [1,1,2] the vectors $\mathbf{a}$ and $\mathbf{b}$ are parallel and they are both perpendicular to $\mathbf{c}$ which we represent by $(\mathbf{a}\|\mathbf{b})\ \perp\mathbf{c}$. In case [1,2,2] $\mathbf{b}$ is perpendicular to $\mathbf{a}$, and $\mathbf{c}$ is perpendicular to $\mathbf{a}$ which we represent by $\mathbf{b}\perp\mathbf{a} \wedge \mathbf{c}\perp\mathbf{a}$. In case [2,2,2] $\mathbf{a}, \mathbf{b}, \mathbf{c}$ are coplanar which we represent by $\mathit{copl}(\mathbf{a}, \mathbf{b}, \mathbf{c})$.

Until now we only considered the vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$, let us now look at $\mathbf{k}, \mathbf{l}, \mathbf{m}$. When III holds then for every case in [1,1,1]...[2,2,2] the configuration of $\mathbf{k}, \mathbf{l}, \mathbf{m}$ is almost determined by the positions of $\mathbf{a}, \mathbf{b}, \mathbf{c}$. For example, for [1,1,1] it holds that $\mathit{copl}(\mathbf{k}, \mathbf{l}, \mathbf{m})$, for [1,1,2] it holds that $\mathbf{k}\perp\mathbf{m} \wedge \mathbf{l}\perp\mathbf{m}$, for [1,2,2] it holds that $(\mathbf{l}\|\mathbf{m}) \perp \mathbf{k}$ and for [2,2,2] it holds that $\mathbf{k}\|\mathbf{l}\|\mathbf{m}$. With that we are in a position to give the complete list of eight unacceptable problem types, showing the combined configuration of $\mathbf{a}, \mathbf{b}, \mathbf{c}$ and $\mathbf{k}, \mathbf{l}, \mathbf{m}$.

1,1,1 $\mathbf{a}\|\mathbf{b}\|\mathbf{c}$ and $\mathit{copl}(\mathbf{k}, \mathbf{l}, \mathbf{m})$

1,1,2 $(\mathbf{a}\|\mathbf{b})\perp\mathbf{c}$ and $\mathbf{k}\perp\mathbf{m}\wedge\ \mathbf{l}\perp\mathbf{m}$
1,2,1 $(\mathbf{a}\|\mathbf{c})\perp\mathbf{b}$ and $\mathbf{k}\perp\mathbf{l}\wedge\mathbf{m}\perp\mathbf{l}$
1,2,2 $(\mathbf{b}\|\mathbf{c})\perp\mathbf{a}$ and $\mathbf{l}\perp\mathbf{k}\wedge\ \mathbf{m}\perp\mathbf{k}$
2,1,1 $\mathbf{b}\perp\mathbf{a}\wedge\ \mathbf{c}\perp\mathbf{a}$ and $(\mathbf{l}\|\mathbf{m})\perp\mathbf{k}$
2,1,2 $\mathbf{a}\perp\mathbf{b}\wedge\ \mathbf{c}\perp\mathbf{b}$ and $(\mathbf{k}\|\mathbf{m})\perp\mathbf{l}$
2,2,1 $\mathbf{a}\perp\mathbf{c}\wedge\mathbf{b}\perp\mathbf{c}$ and $(\mathbf{k}\|\mathbf{l})\perp\mathbf{m}$
2,2,2 $\mathit{copl}(\mathbf{a},\mathbf{b},\mathbf{c})$ and $\mathbf{k}\|\mathbf{l}\|\mathbf{mc}$

Literature

[1] Tuzikov, A. V., Roerdink, J. B. T. M., Heijmans, H. J. A. M., Similarity Measures for Convex Polyhedra Based on Minkowski Addition. Pattern Recognition 33 (2000) 979-995
[2] J. L. Coolidge, A Treatise on the Circle and the Sphere, Clarendon press, Oxford, 1916.
[3] The Rodrigues rotation matrix is discussed in most books on classical mechanics, for example:
Rosenberg, R. M., Analytical Dynamics of Discrete Systems. Plenum Press, New York and London, or
Hamel G., Theorethische Mechanik. Springer-Verlag 1978, or
Goldstein H., Classical Mechanics. Addison-Wesley publishing company.
[4] W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling. Numerical Recipes in C, Cambr. Univ. Press, New York, 1986.
[5] SYNAPS: Available at: http://www.inria.fr/galaad/logiciels/synaps/inex.html
[6] CORS. Accepted for the ICCSA05 conference.
[7] Henk Bekker, Axel Brink, Reducing the time complexity of Minkowski-sum based similarity measures by using geometric inequalities, Proceedings of the international conference on computational science and its applications, ICCSA 2004, LNCS vol. 3045 part III, 32-41, Springer Verlag.

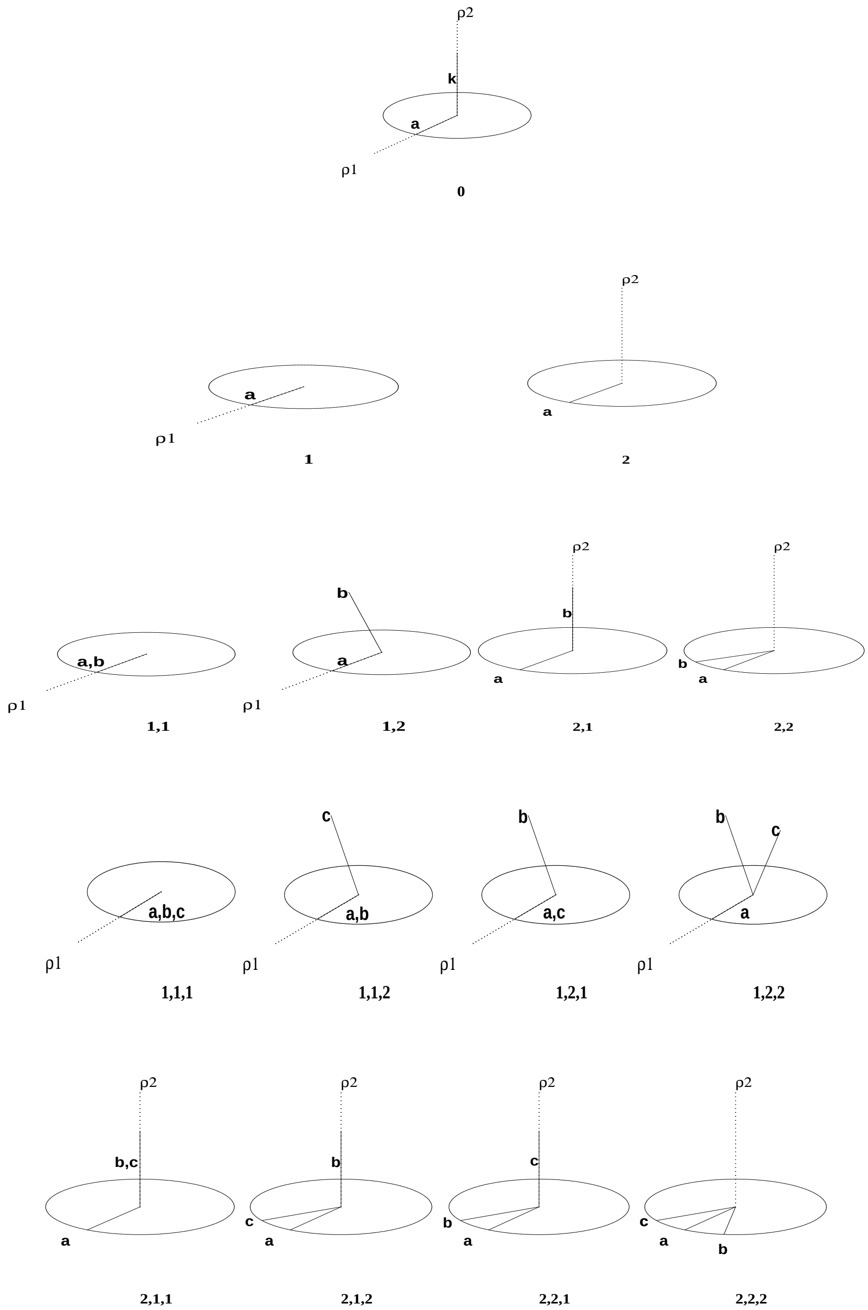

Fig. 7. Unacceptable problem types derived systematically. Figure 0 shows how the axes ρ_1 and ρ_2 are positioned w.r.t. the vectors $\mathbf{a}$ and $\mathbf{k}$, the other figures show a sequence of cases. Only the positions of $\mathbf{a}, \mathbf{b}, \mathbf{c}$ are shown. The sequence $1, 1, 1 \ldots 2, 2, 2$ represents the actual unacceptable problem types. For more details see the text.