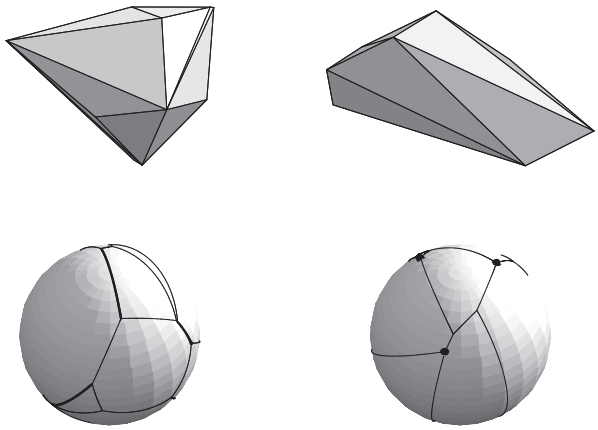
Speeding up the computation of similarity measures based on Minkowski addition in 3D

Axel Brink

April 2004



Master's thesis

Supervisor: Henk Bekker

Scientific Visualization and Computer Graphics
Department of Mathematics and Computing Science
Rijks*universiteit* Groningen

RuG

Abstract

The shape of two objects can be compared by a computer program using similarity measures. The family of similarity measures based on Minkowski addition provides a well-founded way to do this. In practice, they are restricted to comparing the shapes of convex polyhedra. We describe two variants of these measures. The first variant uses the volume of the Minkowski sum, and the second variant uses the so-called mixed volume of the Minkowski sum.

Current implementations have the disadvantage that they are very time-consuming: their time complexity is proportional to n^6, where n is proportional to the complexity of the polyhedra. We developed, implemented and tested improved algorithms for both variants. These algorithms proved to reduce the time complexity to $n^{4.5}$.

Contents

Chapter 1

Introduction

One of the subjects of current research is object recognition by machines. This can for example be applied in business: it enables automatically determining whether an object is in store, when somebody holds a duplicate of it in front of a camera.

This object recognition requires three steps: first, an input device (a camera or a laser beam) observes a real-world object and stores the visually perceived information. Second, the three-dimensional shape of the real-world object is inferred from the visual information, and is represented as a three-dimensional object in the computer. Third, the shape of this object is *compared* to a database of known objects, in order to determine which object was held in front of the input device.

When two objects are compared, the result is a number that indicates to what degree the objects are similar. This number is computed by a function called a *similarity measure*. Many similarity measures exist; each is based on a specific principle.

1.1 Similarity measures based on Minkowski addition

In this thesis, we describe similarity measures based on Minkowski addition. This is a kind of addition that allows us to *add* two objects, resulting in a new object. This new object is referred to as the *Minkowski sum* of the original objects. Minkowski addition is explained in chapter 2, along with other basic concepts.

Tuzikov et al. [6] recently suggested that the Minkowski sum of two three-dimensional objects has some properties that can be used to create a similarity measure. These properties include both its volume and its *mixed volume*. While the volume of an object is an ordinary concept, its mixed volume is quite abstract. In chapter 3, we describe these two properties and how they can each be used for a specific similarity measure.

Given two objects, their similarity can be computed using either one of these similarity measures. However, it is not possible to compare *any* two objects using similarity measures based on Minkowski addition. It is only known how to compute the similarity of *convex polyhedra*. We quickly explain the terms

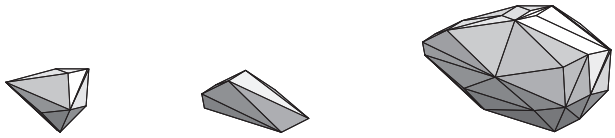


Figure 1.1: Two convex polyhedra (left, middle), which can be compared, and their Minkowski sum (right)

"polyhedron" and "convex". A *polyhedron* is a three-dimensional solid, which has a surface consisting of flat pieces, called *faces*. The faces are connected by line segments, called *edges*. Polyhedra thus do not have a curved surface. However, a curved surface can be approximated by a large number of small faces. Furthermore, a polyhedron is *convex*, in short, when it has no dents. Convex polyhedra are described into more detail in chapter 2.

In some way it is possible to compare polyhedra that are not convex. Convex versions of these polyhedra can be derived before they are compared. This is done by computing their *convex hull*. In the newly formed objects, all dents of the original objects are covered by faces, just like all dents of a gift are covered when it is wrapped in paper.

1.2 Program outline

In order to describe how a computer program can compare two convex polyhedra, we first describe how humans compare objects in daily life. Consider for example two cups, one standing straight up and one lying on its side. A human would pick up the lying cup and rotate it to align its handle with the handle of the other cup. After this, he is able to judge the similarity of the cups.

Notice that it would not make any difference to the human whether the cups were lying or standing straight up from the beginning; that has no influence on the human's judgement on the similarity of the cups. In other words, the similarity of the two objects is not dependent on their orientation. This is called *rotation invariance*.

We describe an algorithm that follows the same procedure with convex polyhedra. It rotates one of the two polyhedra while keeping the orientation of the other constant. For each rotation, a value specific for a similarity measure is computed. In this way, all relative orientations of two convex polyhedra are evaluated, searching for the *best match*. The best match is the orientation that yields the highest value. This value is the similarity value of the two convex polyhedra.

1.3 Critical orientations

Computing a similarity value for each relative orientation would be impossible, because an infinite number of relative orientations exist. However, it was shown [6] [4] that the highest similarity value can be found in a set of relative orientations that is limited: the set of *critical orientations*. These orientations are characterised by a certain combination of edges and faces of both objects that

must be *parallel*. In chapter 3, we explain how exactly edges and faces need to be parallel. It suffices to search only through this limited set of orientations, so computing a similarity value in finite time is possible.

Currently, the first algorithms implementing similarity measures based on Minkowski addition construct a set of critical orientations by first constructing another set. This set contains combinations of edges and faces of the objects. For each of those combinations, it is tried to rotate one of the objects in order to get the edges and faces parallel in the correct way. Bekker et al. [2] designed an algorithm to find one or more such rotations, given a combination of edges and faces. Sometimes this succeeds, in which case the found rotation (or rotations) is a critical orientation. In the other cases, such a rotation doesn't exist.

The set of combinations grows very fast when the complexity of the polyhedra increases, because complex polyhedra by definition have a lot of edges and faces. Furthermore, Bekker's algorithm takes a lot of time. The combination of these two aspects causes that current algorithms are very slow.

1.4 Improved method

We present a method to speed up those algorithms. The idea of the method is that for some of the combinations of edges and faces, it can be detected that they can never become parallel. This saves a call to Bekker's algorithm, which would otherwise try to find an orientation that doesn't exist.

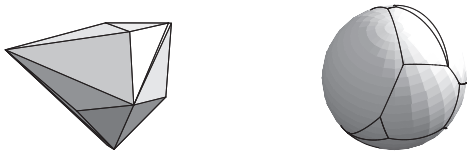


Figure 1.2: A polyhedron and its SDR

In order to detect which edges and faces cannot become parallel, the polyhedra are transformed into a new representation: the *slope diagram representation (SDR)*. An SDR looks like a sphere covered with connected arcs. The arcs somehow describe angles between edges and faces of the original polyhedra. By comparing distances between arcs and endpoints of arcs, combinations of edges and faces that cannot become parallel can be identified. These distances only need to be computed once, so this is done in a preprocessing step, before the best match is searched. We describe the method into more detail in chapter 4.

1.5 Summary

Summarising, current algorithms that compute a similarity measure based on Minkowski addition are very slow. We introduce an improved algorithm that does the following:

1. Compute distances on the SDRs of the polyhedra
2. Create a set of combinations of edges and faces

3. Detect combinations that cannot lead to critical orientations by comparing distances on the SDRs

4. Find critical orientations by calling Bekker's algorithm for each the remaining combinations

5. Compute a similarity value for each critical orientation

6. Yield the highest similarity value

In this algorithm, item number 3 describes the improvement to current algorithms. For this improvement, distances on SDRs are compared in item number 1. Because SDRs are spheres, measuring these distances boils down to measuring distances on the surface of a sphere. This is not trivial. A great deal of our work was dedicated to determining how those distances can be calculated. Chapter 5 describes how it can be done.

We did two series of speed tests: one for the volume measure and one for the mixed volume measure. Chapter 6 presents the results. Chapter 7 discusses these results and concludes the thesis.

Chapter 2

Preliminaries

The shape of convex polyhedra is compared using similarity measures based on Minkowski addition. To compute such a measure, a set of critical orientations must be run through, which can be found using the slope diagram representations (SDRs) of the polyhedra. They consist of arcs covering a unit sphere. The comparison can be speeded up when some distances in an SDR have been computed in advance.

This chapter introduces basic concepts, which are needed by the other chapters: convex polyhedra, Minkowski addition, objects on a sphere, the slope diagram representation and distances in an SDR.

2.1 Convex polyhedra

A polyhedron is a three-dimensional solid which has a surface consisting of flat pieces, called *faces*. The faces are polygons, connected at their edges, which are the line segments where two faces meet. The points where three or more of these edges meet, are called *vertices*.

A polyhedron is convex, in short, when it has no dents. More precisely, a polyhedron is convex if the intersection of every line with the polyhedron is either empty or only a single line segment.

Figure 2.1: Convex polyhedron (left) and non-convex polyhedron (right)

One property of a polyhedron is its *volume*. The volume of a polyhedron A is denoted by $V(A)$. It can be computed by standard methods, for example by dividing the polyhedron in pyramids, each having a face as base, and summing the volumes of all pyramids.

2.2 Minkowski addition

Minkowski addition allows us to *add* two polyhedra, resulting in a new polyhedron:

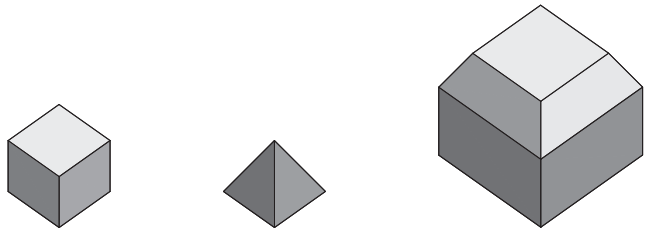


Figure 2.2: Cube (left), pyramid (middle) and their Minkowski sum (right)

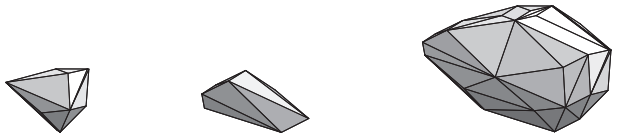


Figure 2.3: Two arbitrary convex polyhedra (left, middle) and their Minkowski sum (right)

The Minkowski sum operation is denoted by the symbol $\oplus$. Formally, the Minkowski sum of objects A and B is defined as:

$$A \oplus B \equiv \{a + b : a \in A, b \in B\} \tag{2.1}$$

Let's take a closer look at this definition, in order to understand what happens. It states that we can find the Minkowski sum C by adding all vectors in A to all vectors in B. Consider a vector a to some point in A. Now add all (infinitely many) vectors in B to a. The resulting vectors define a set of points that together form a duplicate of B, translated by the vector a. All these points belong to C. Now repeat this process for all (infinitely many) choices of a in A. The resulting shape C is the Minkowski sum of A and B.

A few properties of the Minkowski sum are worth mentioning:

- $A \oplus B$ is a polyhedron, so it is possible to compute its volume, denoted by $V(A \oplus B)$.
- $A \oplus B \equiv B \oplus A$.
- When A and B are convex, then $A \oplus B$ is also convex.
- The positions of A and B determine the position of $A \oplus B$, but not its shape.

2.3 Objects on a sphere

The slope diagram representation (SDR) of a convex polyhedron looks like a sphere covered with arcs. Before we explain SDRs exactly, we first need to introduce some concepts. In this section we define "objects" that can be identified on a sphere: *sphere points*, *great circles* and *arcs*. Some of the characteristics of these objects are also described.

2.3.1 Sphere points

A *sphere point* is a vector of length 1. It thus points to a position on a unit sphere.

2.3.2 Great circles

A *great circle* is a circle on a unit sphere, with radius 1. Consequently, its century coincides with the centre of the unit sphere. We introduce some notions about great circles:

- The *plane* of a great circle is the unique plane that contains the great circle.
- Two great circles are *perpendicular* if the planes that contain them are perpendicular. That is, $V \perp W \Rightarrow C_V \perp C_W$, where V and W are planes that contain the circles C_V and C_W, respectively.
- The *poles* of a great circle C, are the two unique points that are defined by the inward and and outward unit normal vectors of the plane of C.

Figure 2.4: A great circle (left) and an arc (right)

2.3.3 Arcs

We define an *arc* to be a piece of a great circle. An arc consists of two *endpoints* enclosing an infinite number of *interior points*. We introduce a couple of notions involving arcs:

- The *plane* of an arc is the unique plane that contains the arc.
- The *poles* of an arc A are the two unique points that are defined by the positive and and negative unit normal vectors of the plane of A.

- The *inverse* of an arc A, A^{-1}, is the arc formed by projecting it through the centre of the unit sphere onto the other side of it. In other words, A^{-1} is the arc that results when applying the inversion symmetry operation $(x, y, z) \rightarrow (-x, -y, -z)$ to all points of A.
- The *lune* of an arc A, $LUNE(A)$, is the region on the sphere that contains A and is bounded by two planes that contain the poles of A, each containing one of the endpoints of A.
- The *antilune* of an arc A, $ALUNE(A)$, is the lune of the inverse of A. That is, $ALUNE(A) \equiv LUNE(A^{-1})$. Furthermore, $LUNE(A) \equiv -ALUNE(A)$.

Figure 2.5: The lune (left, dotted) and antilune (right, dotted) of an arc

2.4 Slope diagram representation

A slope diagram representation (SDR) is a partial description of a convex polyhedron. The slope diagram of a convex polyhedron A is denoted by $SDR(A)$. It consists of a collection of sphere points, called SDR points, and arcs, called SDR arcs. They cover a unit sphere in a particular way.

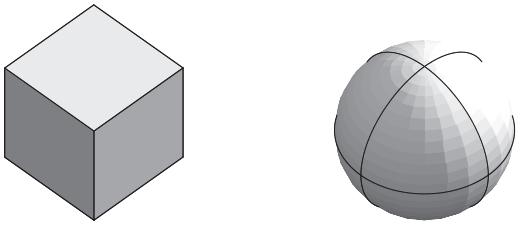


Figure 2.6: A cube and its SDR

All SDR arcs are connected to other SDR arcs; they meet at SDR points. In this way, the SDR arcs divide the surface of the sphere in regions, called *spherical polygons*. As such, an SDR is a subdivision of a unit sphere. The relation between an object and its SDR is as follows:

- An *SDR point* is a sphere point that represents a face of the object. It is the endpoint of the outward unit normal vector of the face it represents. Because any unit vector has length 1, an SDR point is a point on the surface of a unit sphere.

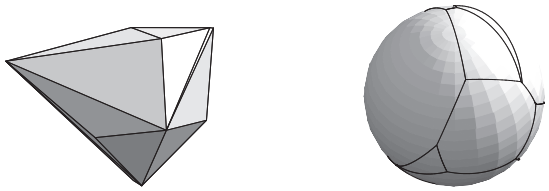


Figure 2.7: An arbitrary polyhedron and its SDR

- An *SDR arc* is an arc between two SDR points, representing the edge between the two faces in the object that are represented by the two SDR points. The angle between these endpoints is equal to the angle between the corresponding faces.
- The arcs enclose *spherical polygons*; each corresponds to a vertex of the polyhedron. This is the vertex where the edges meet to which the enclosing arcs in the SDR correspond. In our implementation, SDR polygons are not stored as a separate structure, however, because they can be computed from the SDR points and arcs. Besides, our method doesn't use SDR polygons.

Two observations about SDRs are worth mentioning:

- *an SDR arc is uniquely identified by its endpoints* In special cases, two sphere points e_1, e_2 do not uniquely identify an arc that connects these points. This is the case when $\angle(e_1, e_2) = 0$ or when $\angle(e_1, e_2) = \pi$. In these cases, there are an infinite number of arcs connecting the sphere points. We prove that this cannot happen with the endpoints of SDR arcs.

  Proof The endpoints e_1, e_2 of an arc A represent adjacent faces of a convex polyhedron. The angle between two adjacent faces in a polyhedron is always less than π. Consequently, $\angle(e_1, e_2) < \pi$. Furthermore, endpoints of an SDR arc must be distinct, so $\angle(e_1, e_2) > 0$.

  One unique great circle C through e_1 and e_2 exists. When C is cut at e_1 and e_2, then A is the unique piece of C of which the angle between the endpoints is less than π. Thus, an SDR arc is uniquely identified by its two endpoints. □

  This allows a definition of the word "between": a sphere point p is *between* the endpoints of an SDR arc when it is on the SDR arc.

- *SDRs can be used to detect parallelness of faces and edges*
  The SDRs of two polyhedra can be overlayed. The overlay may be used to detect parallelness of faces and edges. For example, if faces in A and B are parallel, then the corresponding SDR points of $SDR(A)$ and $SDR(B)$ coincide. Similarly, if an edge in A is parallel to a face in B, then an SDR arc of $SDR(A)$ coincides with an SDR point in $SDR(B)$.

2.5 Distances in a slope diagram representation

The improved method skips certain cases, based on distance inequalities which require that some distances in SDRs are computed in advance. Because SDRs are unit spheres, all our distance calculations concern unit spheres. We will refer to the unit sphere by simply *sphere*. There are four kinds of distances we want to measure: the distance between two sphere points, between a sphere point and a great circle, between a point and an arc and between two arcs.

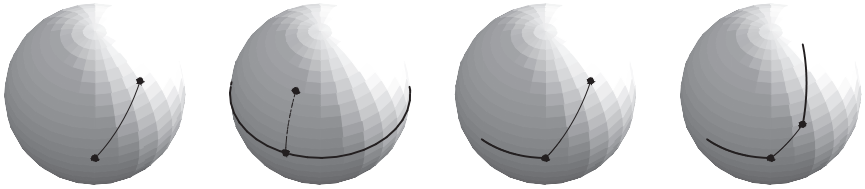


Figure 2.8: From left to right: distance between two sphere points, between a sphere point and a great circle, between a point and an arc and between two arcs

The *distance between two sphere points* $d(p_1, p_2)$ is defined as the length of the shortest path between them, following the surface of the unit sphere. In the literature, this distance is also called "geodesic distance". This path is a segment of a great circle, thus an arc. Concluding, the distance between two points is equal to the length of the arc between these points. Note that the distance is measured in radians.

Apart from the distance between two sphere points, the other three distances cannot be formulated in a single number. A range of distances exists, bounded by a *minimum distance* and *maximum distance*. We denote these distances by:

- $d_{\min}(p, C)$ and $d_{\max}(p, C)$ for a sphere point and a great circle;
- $d_{\min}(p, A)$ and $d_{\max}(p, A)$ for a sphere point and an arc;
- $d_{\min}(A_1, A_2)$ and $d_{\max}(A_1, A_2)$ for two arcs.

In chapter 5, we show how these four kinds of distances can be computed.

Summary This chapter introduced some basic concepts, which are needed by the other chapters: convex polyhedra, Minkowski addition, objects on a sphere, the slope diagram representation and distances in an SDR.

Chapter 3

Similarity measures

When two objects are compared, the result is a number that indicates to what degree the objects are similar. This number is computed by a function called a *similarity measure*. The function yields by convention a number between 0 and 1, where 1 indicates exact resemblance. In the literature, sometimes the notion *dissimilarity measure* is used for a function that works the same but yields 0 for exact resemblance.

Many similarity or dissimilarity measures exist [7]. For instance, the most well-known dissimilarity measure is the *Hausdorff distance*. We first describe this measure for illustrative purposes. Then we describe the family of similarity measures based on Minkowski addition.

3.1 Hausdorff distance

The Hausdorff distance between two point sets A and B is defined as follows:

$$H(A,B) \equiv \max(\vec{h}(A,B), \vec{h}(B,A))$$

where $\vec{h}(A,B)$ denotes the *directed Hausdorff distance* which is defined as follows:

$$\vec{h}(A,B) \equiv \max_{a \in A}\{\min_{b \in B}\{d(a,b)\}\}$$

where d denotes any distance function of two points a and b, for example, Euclidean distance. So the directed Hausdorff distance $\vec{h}(A,B)$ is the distance between two specially chosen points a and b. a is the point in A that has maximum distance to all points in B, and b is the point in B that has minimum distance to a. Similarly, $\vec{h}(B,A)$ can be computed by interchanging A and B.

Summarising, the general Hausdorff distance $H(A,B)$ is the maximum of two distances, each between two points. This causes an important disadvantage of this (dis)similarity measure: it is very sensitive to noise. For instance, when one of the objects is modified such that it contains a "hair", it is probable that a or b in one of the directed Hausdorff distances gets into this hair. In that case, the measure can produce a completely different result.

A characteristic of the Hausdorff distance is that the similarity of two objects depends on three factors:

1. shape

2. relative orientation

3. relative position

Thus, apart from the shape, the similarity value changes when one of the objects is moved or rotated. This phenomenon is often not wanted when only the *shape* of the objects has to be compared. Therefore, it is needed to try all orientations and positions of the objects, searching for the best match. The *best match* is that relative orientation and position, for which the Hausdorff distance produces the lowest value.

3.2 Measures based on Minkowski addition

Tuzikov et al. [6] recently introduced similarity measures based on Minkowski addition for three-dimensional convex polyhedra. Because the shape of the Minkowski sum of two objects is not dependent on their position, these similarity measures are naturally independent on the relative position of the objects. In other words, they are *translation invariant*, which is a great advantage. Thus, in order to find the best match, only relative orientations have to be searched. Furthermore, it was shown [6] [4] that the best match for convex polyhedra can be found in a set of relative orientations that is limited. Thus finding the *exact* relative orientation that produces the best match, is possible.

We describe two variants of similarity measures based on Minkowski addition. The first variant is based on the *volume* of the Minkowski sum of the two compared objects; we name it the *volume measure*. The second variant is based on its *mixed volume*; we name it the *mixed volume measure*.

3.2.1 Volume measure

The volume measure of two objects A and B makes use of a known relation between the volume of $A \oplus B$ and the individual volumes of A and B:

$$V(A \oplus B) \geq 8V(A)^{\frac{1}{2}}V(B)^{\frac{1}{2}} \tag{3.1}$$

This formula is derived from the *Brunn-Minkowski inequality* [5][6]. Generally inequality holds, except for the special case where A and B have the same shape and orientation. In that case, the left-hand side is equal to the right-hand side. This observation suggests dividing the right-hand side by the left-hand side, yielding a starting point for a similarity measure:

$$\frac{8V(A)^{\frac{1}{2}}V(B)^{\frac{1}{2}}}{V(A \oplus B)} \tag{3.2}$$

This formula is called the *volume function*. It yields a number between 0 and 1, inclusive: it yields at most 1, because the denominator is greater than or equal to the numerator; it yields at least 0, because volumes are involved, and volumes cannot be negative. Note also that because only the *volumes* of A, B and their Minkowski sum are involved, relative position plays no role. The formula thus establishes translation invariance.

However, the Minkowski sum of A and B changes when A or B is rotated, as does its volume. So the volume function (3.2) does not satisfy independence of

the relative orientation of A and B, or *rotation invariance*. It is thus required to search through all relative orientations to find the best match. The *best match* is in this case the relative orientation for which the volume function yields the highest value. Any relative orientation can be established by rotating object B while keeping the rotation of object A constant. We thus obtain the following formula:

$$\sigma_1(A,B) \equiv \max_{R \in \Re} \frac{8V(A)^{\frac{1}{2}}V(B)^{\frac{1}{2}}}{V(A \oplus R(B))} \tag{3.3}$$

$\Re$ denotes the set of all rotations in R^3. There are an infinite number of relative orientations, so this formula would be useless. However, Tuzikov et al. [6] [4] proved that the maximum value in formula (3.3) can be found in only a limited set of relative orientations. This set consists of all relative orientations of A and B such that one of the following conditions holds:

- Three edges of A are parallel to three faces of B
- Two edges of A are parallel to two faces of B, and one face of A is parallel to one edge of B
- One edge of A is parallel to one face of B, and two faces of A are parallel to two edges of B
- Three faces of A are parallel to three edges of B

The set of all relative orientations for which one of these conditions holds, is called the set of *critical orientations* for the volume measure. In order to find the maximum specified by formula (3.3), it is only required to evaluate the volume function (3.2) for each of these relative orientations. Chapter 5 describes how all these critical orientations can be found efficiently.

3.2.2 Mixed volume measure

The mixed volume measure is based on the so-called "mixed volume" of two objects. This concept was introduced by Minkowski [3]. It allows an alternative way of computing the volume of the Minkowski sum:

$$V(A \oplus B) \equiv V(A) + 3V(A,A,B) + 3V(A,B,B) + V(B)$$

This formula is based on Minkowski's theorem on mixed volumes, applied to the case of two three-dimensional objects, see [6]. The terms $V(A,A,B)$ and $V(A,B,B)$ denote *mixed volumes*. These mixed volumes are quantities that somehow contribute to the volume of $A \oplus B$, but this concept is quite abstract. We will not try to give a concrete clarification. However, quoting [1], it can be shown that $V(A,A,B)$ is proportional to the area of A and the linear dimension of B. Similarly, $V(A,B,B)$ is proportional to the linear dimension of A and the area of B.

A relation between the mixed volume $V(A,A,B)$ and the volumes of A and B is known:

$$V(A,A,B) \geq V(A)^{\frac{2}{3}}V(B)^{\frac{1}{3}} \tag{3.4}$$

This inequality was also derived from the Minkowski inequality. It is possible to use this inequality to create a *mixed volume function*, just like the volume function we described for the volume measure:

$$\frac{V(A)^{\frac{2}{3}}V(B)^{\frac{1}{3}}}{V(A,A,R(B))} \tag{3.5}$$

The similarity measure then becomes:

$$\sigma_2(A,B) \equiv \max_{R\in\Re} \frac{V(A)^{\frac{2}{3}}V(B)^{\frac{1}{3}}}{V(A,A,R(B))} \tag{3.6}$$

It has been shown [4] that the maximum of this similarity measure can also be found in a limited set of orientations; this is the set of critical orientations for the mixed volume measure. It consists of all relative orientations of A and B such that the following condition holds:

- Three edges of A are parallel to three faces of B

This condition is equal to one of the conditions for the critical orientations for the volume measure, which we described before. Thus, the volume measure and the mixed volume measure are very similar. In order to find the maximum specified by formula (3.6), it is only needed to evaluate the mixed volume function (3.5) for each of the critical orientations of the mixed volume measure. The next chapter describes how all these critical orientations can be found efficiently.

Summary We described two similarity measures: the volume measure and the mixed volume measure. They can both be computed by searching for the maximum of a function that is evaluated for a limited number of critical orientations.

Chapter 4

Method

When the shapes of two convex polyhedra are compared, the result is a number that indicates to what degree the shapes are similar. This number is called a similarity value. To compute this value, a function must be evaluated for a limited set of critical orientations. The orientation that yields the highest value of this function, is called the best match. The similarity value of the convex polyhedra is the function value of this best match.

The critical orientations are characterised by combinations of edges and faces in the polyhedra that must be parallel, as described in 3.2.1 and 3.2.2. In this chapter, we first discuss a primitive method, and then an improved method.

The primitive method is discussed in order to have a reference to compare the improved method to. It searches for critical orientations by treating all of the combinations of edges and faces. For each combination, it tries to rotate one of the polyhedra such that the edges and faces in the combination get parallel.

The improved method finds the critical orientations faster, because it skips some of the combinations. That can be done because for these combinations, it can be determined that it is not possible to get the edges and faces parallel.

4.1 Primitive method

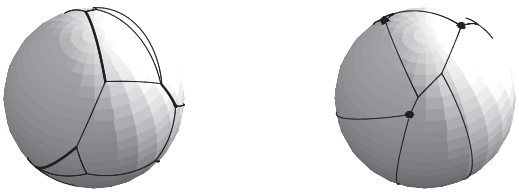


Figure 4.1: The critical orientations are the relative orientations of A and B such that in the overlay of $SDR(A)$ and $SDR(B)$ three arcs in $SDR(A)$ coincide with three SDR points in $SDR(B)$.

The critical orientations are characterised by conditions about parallel faces and edges. These orientations can be identified using the slope diagram representation (SDR). When the SDRs of convex polyhedra A and B are overlayed,

parallelness of faces and edges in A and B appears as SDR points coinciding with SDR arcs.

The primitive method has two variants, one for the volume measure and one for the mixed volume measure. The variants treat different kinds of combinations of arcs and SDR points, because the volume measure and the mixed volume measure have different parallelness conditions. We first describe the variant for the mixed volume measure, because it has only one parallelness condition. After that, we describe the volume measure, which is basically an extension to the variant for the mixed volume measure.

4.1.1 Mixed volume measure

When using slope diagram representations, the condition for the mixed volume as described in 3.2.2 becomes:

- Three SDR arcs of $SDR(A)$ coincide with three SDR points of $SDR(B)$

Note that this condition also includes the special case that SDR points of $SDR(A)$ coincide with SDR points of $SDR(B)$, because an SDR points are endpoints of SDR arcs.

A straightforward approach to finding all orientations satisfying the above condition, is treating all combinations of three SDR arcs of A and three SDR points of B. Each combination looks like a six-tuple $(a_1^A, a_2^A, a_3^A, p_1^B, p_2^B, p_3^B)$, where a_i^A are arcs in $SDR(A)$ and p_i^B are points in $SDR(B)$, and each a_i^A must coincide with p_i^B. For each of these combinations, we have to try to rotate B such that the points fall on the arcs. If we managed to find such a rotation R, we have to evaluate the mixed volume function (3.5). We repeat this process for all combinations, searching for the maximum of the mixed volume function. This maximum is the similarity of the compared polyhedra.

Finding a rotation R such that three SDR points p_1^B, p_2^B, p_3^B of $R(SDR(B))$ coincide with three arcs a_1^A, a_2^A, a_3^A of A can be done using an algorithm developed by Bekker [2], called `tvt`. It uses numerical approximation techniques to yield zero or more rotations R.

The primitive method is implemented by the following algorithm:

$\sigma = 0$
`for all` a_1^A
  `for all` a_2^A `>` a_1
    `for all` a_3^A `>` a_2
      `for all` p_1^B
        `for all` p_2^B
          `for all` p_3^B
            R `= tvt(`$a_1^A, a_2^A, a_3^A, p_1^B, p_2^B, p_3^B$`)`
            σ_R `=` $\frac{V(A)^{\frac{2}{3}} V(B)^{\frac{1}{3}}}{V(A,A,R(B))}$
            `if` σ_R `>` σ `then` σ `=` σ_R

Algorithm 1a: primitive method for the mixed volume measure

There are two reasons why this algorithm is slow:

1. The function `tvt` is quite slow: on current home PCs, it can be executed only about 1000 times per second.

2. There are a lot of combinations, so `tvt` is called many times.

Small optimisations Note that it does not matter in which order we say "a_1^A coincides with p_1^B", "a_2^A coincides with p_2^B" and "a_3^A coincides with p_3^B". So for example, $(a_1^A, a_2^A, a_3^A, p_1^B, p_2^B, p_3^B)$ is equivalent to $(a_2^A, a_1^A, a_3^A, p_2^B, p_1^B, p_3^B)$. There are six of those equivalent permutations. In this algorithm, the permutations of SDR arcs are omitted. In this way, a constant factor of six is gained. This is established by the >'s; they use the fact that SDR arcs are linearly ordered in our implementation.

There are other combinations of SDR arcs and SDR points that can be left out. Those are the combinations for which one of the following conditions hold:

- $p_1^B = p_2^B = p_3^B$
- $a_1^A = a_2^A = a_3^A$
- $p_i^B = p_j^B$ while $a_i^A = a_j^A$

For these combinations always an infinite number of critical orientations exist. For example, the first condition corresponds to the case that the three chosen points are equal. We show that this combination can be left out.

$p_1^B = p_2^B = p_3^B$ means that there is effectively only one SDR point of B that is restricted in its position, leaving an infinite number of satisfying relative orientations. Let M denote the set of these relative orientations. In another iteration of the nested `for`-loops, a combination will occur for which two of the p's are the same as before, and the third different. Let N denote the set of relative orientations satisfying this combination. N is a subset of M, because it contains the orientations that satisfy the same condition and one extra condition. So these orientations will not result in a lower value of the (mixed) volume function. □

All combinations of SDR arcs and SDR points that satisfy the above conditions, are left out in our implementation of the primitive method. However, it can be argued that these combinations should not be left out in the primitive method, because the primitive method wouldn't be primitive anymore.

4.1.2 Volume measure

Using the slope diagram representation, the critical orientations of the volume measure can be defined as the set of orientations satisfying one of the following conditions:

- Three SDR arcs of $SDR(A)$ coincide with three SDR points of $SDR(B)$;
- Two SDR arcs of $SDR(A)$ coincide with two SDR points of $SDR(B)$, and one SDR point of $SDR(A)$ coincides with one arc of $SDR(B)$;
- One SDR arc of $SDR(A)$ coincides with one SDR point of $SDR(B)$, and two SDR points of $SDR(A)$ coincide with two arcs of $SDR(B)$;
- Three SDR points of $SDR(A)$ coincide with three SDR arcs of $SDR(B)$.

The primitive method works essentially the same for the volume measure as for the mixed volume measure. For each of the conditions, the primitive method has six nested `for`-loops:

$\sigma = 0$
`for all` a_1^A
  `for all` a_2^A `>` a_1^A
    `for all` a_3^A `>` a_2^A
      `for all` p_1^B
        `for all` p_2^B
          `for all` p_3^B
            R `= tvt(`$a_1^A, a_2^A, a_3^A, p_1^B, p_2^B, p_3^B$`)`
            σ_R `=` $\frac{8V(A)^{\frac{1}{2}}V(B)^{\frac{1}{2}}}{V(A \oplus R(B))}$
            `if` σ_R `>` σ `then` σ `=` σ_R
`for all` a_1^A
  `for all` a_2^A `>` a_1^A
    `for all` p_3^A
      `for all` p_1^B
        `for all` p_2^B
          `for all` a_3^B
            R `= tvt(`$a_1^A, a_2^A, p_3^A, p_1^B, p_2^B, a_3^B$`)`
            σ_R `=` $\frac{8V(A)^{\frac{1}{2}}V(B)^{\frac{1}{2}}}{V(A \oplus R(B))}$
            `if` σ_R `>` σ `then` σ `=` σ_R
`for all` a_1^A
  `for all` p_2^A
    `for all` p_3^A
      `for all` p_1^B
        `for all` a_2^B
          `for all` a_3^B
            R `= tvt(`$a_1^A, p_2^A, p_3^A, p_1^B, a_2^B, a_3^B$`)`
            σ_R `=` $\frac{8V(A)^{\frac{1}{2}}V(B)^{\frac{1}{2}}}{V(A \oplus R(B))}$
            `if` σ_R `>` σ `then` σ `=` σ_R
`for all` p_1^A
  `for all` p_2^A
    `for all` p_3^A
      `for all` a_1^B
        `for all` a_2^B
          `for all` a_3^B
            R `= tvt(`$p_1^A, p_2^A, p_3^A, a_1^B, a_2^B, a_3^B$`)`
            σ_R `=` $\frac{8V(A)^{\frac{1}{2}}V(B)^{\frac{1}{2}}}{V(A \oplus R(B))}$
            `if` σ_R `>` σ `then` σ `=` σ_R

Algorithm 1b: primitive method for the volume measure

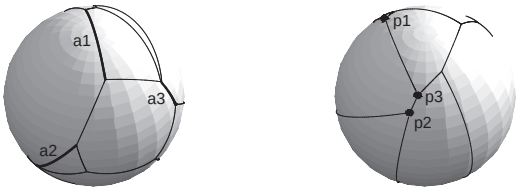


Figure 4.2: It is not always possible to rotate B such that three SDR points of $SDR(B)$ coincide with three SDR arcs of $SDR(A)$. This combination of SDR arcs and SDR points can thus be skipped. This observation is the essence of the improved method.

4.2 Improved method

4.2.1 Mixed volume measure

Finding critical orientations for the mixed volume measure requires rotating B such that three SDR points of $SDR(B)$ coincide with three SDR arcs of $SDR(A)$. In some cases there is no such orientation. Consider for example figure 4.2. It is clear that the marked SDR points of $SDR(B)$ can never fall on the marked arcs of $SDR(A)$. Points p_1 and p_2 could fit on a_1 and a_2, but then p_3 cannot fall on a_3, because p_3 is close to p_2 while a_3 is not close to a_2. Generally speaking, the distances do not agree.

There is no need to call the slow function `tvt` with this combination, because we already know that `tvt` cannot find a rotation for it. Skipping this combination saves time. When the distances between SDR arcs in $SDR(A)$ and distances between SDR points in $SDR(B)$ are known, the remaining combinations are identified by the following condition:

$$\begin{array}{ll} d_{\min}(a_1^A, a_2^A) \le d(p_1^B, p_2^B) \le d_{\max}(a_1^A, a_2^A) & \text{and} \\ d_{\min}(a_2^A, a_3^A) \le d(p_2^B, p_3^B) \le d_{\max}(a_2^A, a_3^A) & \text{and} \\ d_{\min}(a_1^A, a_3^A) \le d(p_1^B, p_3^B) \le d_{\max}(a_1^A, a_3^A) & \end{array} \tag{4.1}$$

These inequalities are called *distance inequalities*. The following algorithm skips combinations for which these distance inequalities do not hold:

$\sigma = 0$
`for all` a_1^A
  `for all` a_2^A `>` a_1^A
    `for all` a_3^A `>` a_2^A
      `for all` p_1^B
        `for all` p_2^B
          `for all` p_3^B
            `if` $d_{\min}(a_1^A, a_2^A) \le d(p_1^B, p_2^B) \le d_{\max}(a_1^A, a_2^A)$ `and`
               $d_{\min}(a_2^A, a_3^A) \le d(p_2^B, p_3^B) \le d_{\max}(a_2^A, a_3^A)$ `and`
               $d_{\min}(a_1^A, a_3^A) \le d(p_1^B, p_3^B) \le d_{\max}(a_1^A, a_3^A)$ `then`
              R `= tvt(`$a_1^A, a_2^A, a_3^A, p_1^B, p_2^B, p_3^B$`)`
              σ_R `=` $\frac{V(A)^{\frac{2}{3}} V(B)^{\frac{1}{3}}}{V(A,A,R(B))}$
              `if` σ_R `>` σ `then` σ `=` σ_R

Algorithm 2: improved method for the mixed volume measure

Summarising, in order to determine which combinations can be skipped, the distances between the SDR points of $SDR(B)$ must be compared to the distances between the SDR arcs of $SDR(A)$. These distances can be computed and stored in tables during a preprocessing step. In chapter 5, it is explained how these distances can be computed.

4.2.2 Volume measure

For the volume measure, similar improvements can be done based on distance inequalities. The inequalities in formula (4.1) are applicable again for the first nesting of `for`-loops in algorithm 1b. The distances between three SDR arcs of $SDR(A)$ are compared to the distances between three SDR points of $SDR(B)$. This combination of SDR arcs and SDR points can be written as $(a_1^A, a_2^A, a_3^A, p_1^B, p_2^B, p_3^B)$. Similar inequalities apply to the the fourth nesting, but this time the SDR points come from $SDR(A)$ and the SDR arcs come from $SDR(B)$.

For the second and third nesting, the combinations are more complicated. The combinations of the second nesting are of the form $(a_1^A, a_2^A, p_3^A, p_1^B, p_2^B, a_3^B)$. For such combinations, also distances between an SDR point and an SDR arc must be compared. For these comparisons, it must be tested whether the ranges of distances overlap:

$$
\begin{array}{ll}
d_{\min}(a_1^A, a_2^A) \leq d(p_1^B, p_2^B) \leq d_{\max}(a_1^A, a_2^A) & \text{and} \\
d_{\max}(p_3^A, a_2^A) \geq d_{\min}(p_2^B, a_3^B) & \text{and} \\
d_{\min}(p_3^A, a_2^A) \leq d_{\max}(p_2^B, a_3^B) & \text{and} \\
d_{\max}(p_3^A, a_1^A) \geq d_{\min}(p_1^B, a_3^B) & \text{and} \\
d_{\min}(p_3^A, a_1^A) \leq d_{\max}(p_1^B, a_3^B) &
\end{array}
$$

The combinations of the third nesting are of the form $(a_1^A, p_2^A, p_3^A, p_1^B, a_2^B, a_3^B)$, and are thus similar to the combinations for the second nesting.

All mentioned distance inequalities can be easily plugged into the primitive method for the volume measure by adding `if`-statements. The resulting improved method is not printed.

Summary We introduced a primitive and an improved method for computing the similarity of two convex polyhedra. The improved method is faster because it takes distance inequalities in account. Both methods can compute two variants of similarity measures based on Minkowski addition: the volume measure and the mixed volume measure.

Chapter 5

Distance computations

In this chapter we explain how the distance between two sphere points and the distance between two arcs can be computed. Computing the distance between two points will prove to be simple, but computing the distance between two arcs requires more work. We explain how to compute the distance between two arcs in four steps. The first step is the distance between two sphere points. Then we consider the distance between a sphere point and a great circle, the distance between a sphere point and an arc, and finally the distance between two arcs.

5.1 Distance between two points

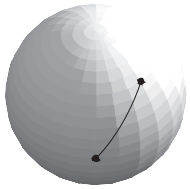


Figure 5.1: Distance between two points

The distance between two sphere points p_1 and p_2 is equal to the length of the arc D between these points. The length of D is proportional to the angle θ between p_1 and p_2. Because sphere points lie on the surface of the unit sphere and we measure in radians, the length of D is *equal* to the angle between p_1 and p_2. Computing this angle in Cartesian coordinates can be done using the inner product: $\theta = \cos^{-1}(p_1 \cdot p_2)$. Thus:

$$d(p_1, p_2) \equiv \cos^{-1}(p_1 \cdot p_2)$$

5.2 Distance between a point and a great circle

Now that we know how to compute the distance between two sphere points, we go to the next step: computing the minimum and maximum distance between

a sphere point p and a great circle C. First, we search for the points on C that are the closest to and the farthest from p. We call these points $q_{\min}$ and $q_{\max}$, respectively. The minimum distance between p and C is then equal to the distance between p and $q_{\min}$. Thus, $d_{\min}(p, C) = d(p, q_{\min})$. Similarly, $d_{\max}(p, C) = d(p, q_{\max})$. We first describe how to find $q_{\min}$, and then $q_{\max}$.

5.2.1 Minimum distance

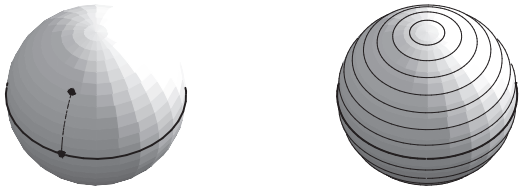


Figure 5.2: Minimum distance between a point and a great circle (left); curves of equal (minimum) distance to a great circle (right). The great circle is drawn thick.

$q_{\min}$ is the point on C that has minimum distance to p. This distance is equal to the length of the arc D between p and $q_{\min}$.

Theorem 5.2.1 $D \perp C$

Proof Without loss of generality, we define an orthogonal set of base vectors such that the centre of the unit sphere is in the origin, C is in the xz-plane and p is in the xy-plane. For each $d \in (0, \pi)$, the points on the sphere having distance d to the point p define a circle $C_D(d)$. When d increases, starting from 0, there is a unique point where $C_D(d)$ first touches C. This is the point $q_{\min}$. p, $q_{\min}$, and the arc D between them, are all in the xy-plane. This plane is perpendicular to the xz-plane, which is the plane of C. Because the planes of D and C are perpendicular, $D \perp C$. □

From $D \perp C$ it follows that $q_{\min}$ can be found by moving from p along the surface of the sphere to C, perpendicularly to C. This can be done computationally by a projection: first, p is projected orthogonally to the plane of C. Two distinct vectors e_1 and e_2 in the plane of C are needed for the projection. The projection can now be done using the following formula:

$$p' = (e_1 \cdot p)e_1 + (e_2 \cdot p)e_2$$

The resulting point p' is on the intersection line of the planes of C and D. Next, p' is normalised, yielding $q_{\min}$:

$$q_{\min} = \frac{p'}{|p'|}$$

A problem arises when p is a pole of A, because then p' is the origin of the sphere and $|p'| = 0$. This vector cannot be normalised. However, we do not need a projection in this case because the minimum distance between an equator and its pole is known to be $\frac{1}{2}\pi$.

5.2.2 Maximum distance

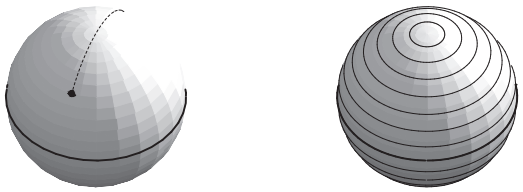


Figure 5.3: Maximum distance between a sphere point and a great circle (left); curves of equal maximum distance to a great circle (right). The great circle is drawn thick.

$q_{\max}$ is the point on C that has maximum distance to p. We now prove that $q_{\max}$ is exactly on the other side on C with respect to $q_{\min}$.

Theorem 5.2.2 $q_{\max} = -q_{\min}$

Proof Like in the proof in the former section, we increase d, but now continuing from the distance where we stopped then. The intersection of $C_D(d)$ and C then consists of two points, until these points meet in the point the farthest from p: this is $q_{\max}$. Like $q_{\min}$, this point is in the xy-plane and on C. The xy-plane intersects C at exactly two points, which are each other's inverse because C is centred at the origin. These are the points $q_{\min}$ and $q_{\max}$, so $q_{\max} = -q_{\min}$. □

Summary Computing the minimum and maximum distance between a sphere point p and a circle C first requires finding the points $q_{\min}$ and $q_{\max}$. These can be computed using a projection and a normalisation. Then, the distances can be computed as distances between sphere points: $d_{\min}(p, C) = d(p, q_{\min})$ and $d_{\max}(p, C) = d(p, q_{\max})$.

5.3 Distance between a point and an arc

Now that we have found the minimum and maximum distance from a point to a great circle, we can use this information to compute the minimum and maximum distance between a sphere point p and an arc A. The arc A is part of a great circle C_A. We compute the points $q_{\min}$ and $q_{\max}$ on the circle C_A as described in section 5.2. The endpoints of A serve as vectors defining the plane of C, needed for the projection. We first consider the minimum distance, and then the maximum distance.

5.3.1 Minimum distance

If $q_{\min}$ happens to be an interior point of A, then this point of A must be the closest to p, because $q_{\min}$ is the point on C_A that is closest to p, and A is a part of C_A. This happens when $p \in LUNE(A)$. The minimum distance between p and A is then equal to the distance between the points p and $q_{\min}$. Thus:

$$q_{\min} \in A \Rightarrow d_{\min}(p, A) \equiv d(p, q_{\min}).$$

Figure 5.4: Minimum distance between a sphere point and an arc. In the left picture, the point is in the lune of the arc; in the right picture, it is not.

If $q_{\min}$ is not an interior point of A, then there is no interior point of A that has minimum distance to p. So one of the endpoints of A must have minimum distance to p. When a_1 and a_2 are the endpoints of A, we can compute the distance of each to p and take the minimum:

$$q_{\min} \notin A \Rightarrow d_{\min}(p, A) \equiv \min(d(p, e_1), d(p, e_2))$$

Determining whether $q_{\min} \in A$ can be done using the following formula:

$$\left.\begin{array}{ll} (e_1 \times q_{\min}) \cdot (q_{\min} \times e_2) > 0 & \text{and} \\ (e_1 \times q_{\min}) \cdot (e_1 \times e_2) > 0 & \end{array}\right\} \Rightarrow q_{\min} \in A$$

The first part is true if and only if $q_{\min}$ is a vector between the endpoints of A, or between their negative counterparts. The second part is true if and only if $q_{\min}$ is not between the negative counterparts.

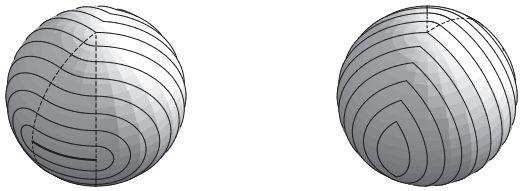


Figure 5.5: Curves of equal minimum distance to an arc, and its lune. Front view (left) and back view (right). The arc is drawn thick.

Figure 5.5 shows curves of equal minimum distance to an arc A. The curves have a different shape in the lune of A than outside the lune. Inside the lune, the shape is like a part of the shape of the curves of minimum distance to the great circle carrying A. This shows that the minimum distance from a point p in $L(A)$ is equal to the minimum distance between the sphere points p and $q_{\min}$, the point on the circle containing A that has minimum distance to p.

5.3.2 Maximum distance

Similarly, if $q_{\max}$ is an interior point of A, then the maximum distance between p and A is equal to the distance between p and $q_{\max}$. This happens when $p \in ALUNE(A)$. Otherwise, the maximum distance is equal to the distance between p and one of the endpoints of A:

Figure 5.6: Maximum distance between a sphere point and an arc. In the left picture, the point is in the antilune of the arc; in the right picture, it is not.

$$q_{\max} \in A \Rightarrow d_{\max}(p, A) \equiv d(p, q_{\max})$$
$$q_{\max} \notin A \Rightarrow d_{\max}(p, A) \equiv \max(d(p, a_1), d(p, a_2))$$

Determining whether $q_{\max} \in A$ now becomes:

$$\left.\begin{array}{l} (e_1 \times q_{\min}) \cdot (q_{\min} \times e_2) > 0 \quad \text{and} \\ (e_1 \times q_{\min}) \cdot (e_1 \times e_2) < 0 \end{array}\right\} \Rightarrow q_{\max} \in A$$

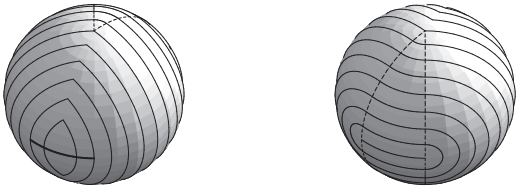


Figure 5.7: Curves of equal maximum distance to an arc, and its antilune. Front view (left) and back view (right). The arc is drawn thick.

Figure 5.7 shows curves of equal maximum distance to an arc A. It is clear that the distance curves in the antilune have the same shape as a part of the curves of maximum distance to a great circle; the great circle of which A is a part. This shows that the maximum distance from a point p in $ALUNE(A)$ is equal to the maximum distance between p and $q_{\max}$, the point on the circle containing A that has maximum distance to p.

More remarkable is the fact that figure 5.7 and figure 5.5 are very similar. The curves of maximum distance to A are equal to the curves of minimum distance to A^{-1}. The arc A^{-1} represents the set of points that have maximum distance to A, which is π. This leads to an alternative definition of the maximum distance:

$$d_{\max}(p, A) \equiv \pi - d_{\min}(p, A^{-1}) \tag{5.1}$$

Summary Computing the minimum and maximum distance between a point and an arc first requires finding the points $q_{\min}$ and $q_{\max}$. Then, it is needed to test if $q_{\min}$ or $q_{\max}$ is an interior point of A. If $q_{\min}$ is an interior point of A, then the minimum distance is equal to the distance between the sphere points p and $q_{\min}$. Otherwise, the minimum distance is equal to the distance between p and an endpoint of A. The maximum distance can be found similarly.

5.4 Distance between two arcs

Now that we know how to compute the distance between a point and an arc, we are able to compute the distance between two arcs. To this end, we consider the variable points q_1 and q_2 on the arcs A_1 and A_2, respectively. The minimum distance between A_1 and A_2 is the minimum distance between q_1 and q_2. Therefore, we first search for the points $q_{1,\min}$ and $q_{2,\min}$, which are the positions of q_1 and q_2 with minimum distance. After that, the minimum distance can be easily computed by $d_{\min}(A_1, A_2) = d(q_{1,\min}, q_{2,\min})$. Similarly, in order to compute the maximum distance between A_1 and A_2, we search for $q_{1,\max}$ and $q_{2,\max}$ and compute their distance by $d_{\max}(A_1, A_2) = d(q_{1,\max}, q_{2,\max})$. We first discuss the minimum distance, and then the maximum distance.

5.4.1 Minimum distance

We identify two cases that can apply:

1. $q_{1,\min}$ and $q_{2,\min}$ are endpoints of A_1 and A_2
2. $q_{1,\min}$ is an endpoint of A_1 and $q_{2,\min}$ is an interior point of A_2, or vice versa

One might expect a third case in which $q_{1,\min}$ and $q_{2,\min}$ are both interior points of A_1 and A_2, but this case can never occur:

Theorem 5.4.1 *$q_{1,\min}$ and $q_{2,\min}$ are not both interior points of A_1 and A_2*

Proof If $q_{1,\min}$ and $q_{2,\min}$ are interior points of A_1 and A_2, then there are two possible causes:

1. the arcs are curved towards each other, or
2. the arcs intersect, implying a distance of zero.

We show that both situations cannot happen. First, two arcs cannot intersect, because they represent edges of a polyhedron and edges never intersect. Second, the arcs cannot be curved towards each other, because in that case the circles of which the arcs are a part could not have the same centre. But arcs are parts of great circles, which all have the same centre, so this can never be the case. □

Concluding, the minimum distance between two arcs is equal to the minimum distance between an arc and an endpoint of the other arc. We already know how to compute the minimum distance between a point and an arc: it is discussed in 5.3.1. However, it is not known which endpoint to take. So all four possibilities must be checked, searching for the minimum:

$$d_{\min}(A_1, A_2) = \min(d_{\min}(e_1^1, A_2), d_{\min}(e_2^1, A_2), d_{\min}(e_1^2, A_1), d_{\min}(e_2^2, A_1)) \tag{5.2}$$

Where e_j^i denotes endpoint i of arc j. Formula 5.2 is defined in terms of the minimum distance between a point and an arc. Figure 5.8 shows that the position of one endpoint of an arc relative to the lune of the other arc plays a role, just like the case of the distance between a point and an arc.

Figure 5.8: Minimum distance between two arcs. In the right picture, an endpoint of one arc is in the lune of the other arc.

5.4.2 Maximum distance

Figure 5.9: Maximum distance between two arcs. In the right picture, an endpoint of one arc is in the antilune of the other arc.

For computing the maximum distance, three cases can apply:

1. $q_{1,\max}$ and $q_{2,\max}$ are endpoints of A_1 and A_2
2. $q_{1,\max}$ is an endpoint of A_1 and $q_{2,\max}$ is an interior point of A_2, or vice versa
3. $q_{1,\max}$ and $q_{2,\max}$ are interior points of A_1 and A_2

The first two cases are similar to the two cases for the minimum distance, as is the computation of the maximum distance in these cases:

$$d_{\max}(A_1, A_2) = \max(d_{\max}(e_1^1, A_2), d_{\max}(e_2^1, A_2), d_{\max}(e_1^2, A_1), d_{\max}(e_2^2, A_1)) \tag{5.3}$$

But the third case deserves special attention. In 5.4.1 we argued that two arcs A_1 and A_2 cannot intersect, which would otherwise imply a minimum distance of zero. Recall that formula 5.1 states that the maximum distance between a point and an arc can also be reformulated as π minus the minimum distance between the point and the inverse of the arc:

$$d_{\max}(p, A_1) \equiv \pi - d_{\min}(p, A_1^{-1}) \tag{5.4}$$

Because we now need to compute the minimum distance between two arcs, we choose p on A_2 such that it has minimum distance to A_1^{-1}. Key observation is that A_1^{-1} *and* A_2 *can intersect.* Thus, point p can be an internal point of A_1^{-1} and A_2 at the same time, implying $d_{\min}(p, A_1^{-1}) = 0$. In this case, formula 5.4 yields a maximum distance of π. Concluding:

$$A_1^{-1} \cap A_2 = \emptyset \Rightarrow d_{\max}(A_1, A_2)$$
$$= \max(d_{\max}(e_1^1, A_2), d_{\max}(e_2^1, A_2), d_{\max}(e_1^2, A_1), d_{\max}(e_2^2, A_1))$$
$$A_1^{-1} \cap A_2 \neq \emptyset \Rightarrow d_{\max}(A_1, A_2) = \pi$$

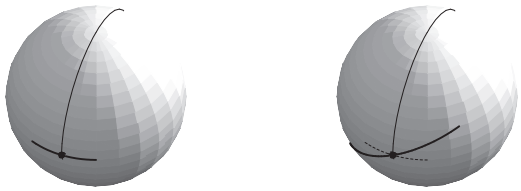


Figure 5.10: Maximum distance between two arcs, when one arc intersects the inverse of the other arc. The inverse arc is drawn dotted. Left: front view; right: back view.

So it must be checked whether the arcs A_1^{-1} and A_2 intersect or not. In the following paragraph, we describe how this can be done.

Intersection testing First, let l_1 be the straight line segment that connects the endpoints e_1^1 and e_1^2 of arc A_1^{-1}. Similarly, let l_2 be the straight line segment that connects the endpoints e_2^1 and e_2^2 of arc A_2. A_1^{-1} and A_2 intersect if and only if a unique half-line l_{int} from the origin through the intersection point exists, which intersects l_1 in the point p_{l_1} and l_2 in the point p_{l_2}. So, A_1^{-1} and A_2 intersect if a point p_{l_1} on l_1 is a linear multiple of a point p_{l_2} on l_2. To check whether this is the case, we first parametrise l_1 with parameter λ ($0 \leq \lambda \leq 1$) and l_2 with parameter μ ($0 \leq \mu \leq 1$):

$$l_1 = \lambda e_1^2 + (1 - \lambda) e_1^1$$
$$l_2 = \mu e_2^2 + (1 - \mu) e_2^1$$

By equating l_1 and $\sigma \cdot l_2$, with $\sigma > 0$, we establish that p_{l_1} and p_{l_2} are on one half-line:

$$\lambda e_1^2 + (1 - \lambda) e_1^1 = \sigma(\mu e_2^2 + (1 - \mu) e_2^1)$$

This equation can be rewritten as follows:

$$(e_1^2 - e_1^1)\lambda + (e_2^2 - e_2^1)\mu\sigma - e_2^2\sigma = -e_1^1 \tag{5.5}$$

Because all e_j^i are three dimensional vectors, formula 5.5 can be seen as a 3x3 linear system of equations, with unknowns λ, $\mu\sigma$ and σ. These parameters can be found by standard methods, such as Gaussian elimination. When the found values satisfy $\lambda > 0$, $0 \leq \lambda \leq 1$ and $0 \leq \mu \leq 1$, A_1^{-1} and A_2 intersect.

Summary The minimum distance between two arcs can be defined in terms of the minimum distance between one endpoint and one arc. The same generally holds for the maximum distance between two arcs, except when one arc intersects the inverse of the other arc. In that case, the maximum distance is π.

Chapter 6

Results

In order to test how much faster the improved method is than the primitive method, we implemented a computer program that simulates the comparison of two convex polyhedra. This program takes two polyhedra as input. The polyhedra are transformed into convex versions by computing their convex hull. After that, the program computes a desired similarity measure. This can be the volume or the mixed volume measure. The output of the program consists of two numbers, one for the primitive method and one for the improved method. These numbers indicate for both methods the number of calls to `tvt` needed. Recall that this is the slow function that tries to find one or more orientations for which specific parallelness conditions hold, for a given combination of arcs and SDR points. By recording both numbers, the improved method can be compared with the primitive method.

We performed two series of performance tests. In the first series, the methods computed the volume measure; in the second the methods computed the mixed volume measure. This was done by repeatedly executing the program with as input a pair of generated polyhedra of increasing complexity, where complexity is defined as the number of faces. The polyhedra are generated randomly, such that both polyhedra are equally complex.

We first present the results for the volume measure, and then those for the mixed volume measure. For both we show how the number of calls to `tvt` relates to the complexity of the polyhedra.

6.1 Volume measure

The left graph of figure 6.1 shows the number of calls to `tvt` needed for the primitive method. It shows that the time needed to compare two polyhedra is more than proportional to their complexity. Notice the huge scale and recall that executing `tvt` 1000 times takes one second on current home PCs. This means for example that polyhedra with 24 faces require about *ten days* to compare using the primitive method. Even when we use a computer that is a hundred times faster, the time required to compare such polyhedra still is far beyond practicable.

The graph on the right shows the performance of the improved method. This graph looks similar to that of the left graph, but there are two differences:

the range on the y-axis is smaller and the slope is lower. With this method, polyhedra with 24 faces require about half a day to compare.

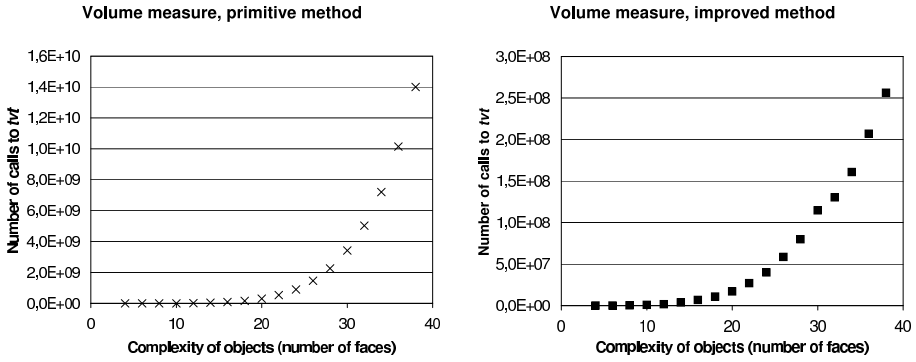


Figure 6.1: Performance of methods for the volume measure. Primitive method (left) and improved method (right). Notice the difference in scale.

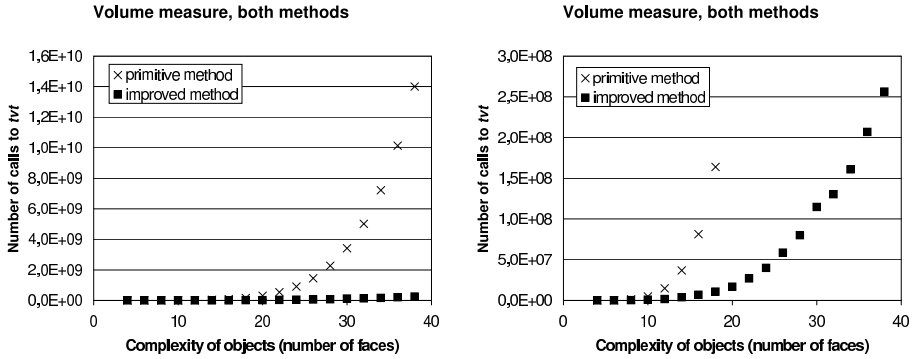


Figure 6.2: Performance of methods for the volume measure. Primitive and improved method combined. Large scale on the y-axis (left) and smaller scale (right).

Figure 6.2 shows the performance of both methods combined in one graph. The left graph shows not only that the improved method performs better than the primitive method, but also that the improvement increases when the complexity of the polyhedra increases. The right graph is a copy of the left one, but with a smaller range on the y-axis. It shows that the improved method also performs better with polyhedra of very low complexity. For example, when two polyhedra with 12 faces are compared, the primitive method makes 149 million calls to `tvt` (4 hours) while the improved method only needs 17 million (30 minutes).

Curve fitting The left graph of figure 6.3 shows the performance of the methods for the volume measure on a logarithmic scale. The fact that the data points of each method lie approximately on a straight line, indicates an underlying

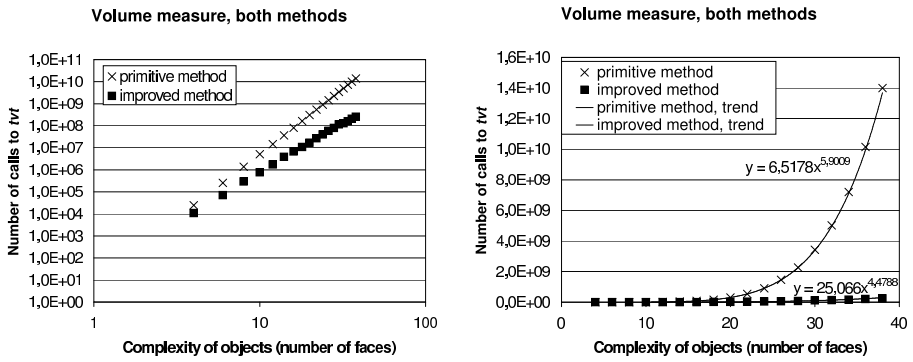


Figure 6.3: Performance of methods for the volume measure. Primitive and improved method combined on a logarithmic scale (left) and on a linear scale with fitted curves (right).

function of the form $y = a \cdot x^b$. We used the least squares method to fit curves on the data points, based on this function. The functions of the resulting curves are:

- $y = 6.52x^{5.90}$ for the primitive method;
- $y = 25.07x^{4.48}$ for the improved method.

The curves of these functions on a linear scale are shown in the right graph of figure 6.3. Note that in these functions, x denotes the number of faces in the polyhedra. In the following, we will also use the letter n for this quantity. For the the time complexity, only the power of x (or n) matters: $y \propto x^{5.90}$ for the primitive method and $y \propto x^{4.48}$ for the improved method.

The primitive method contains four times a nesting of six `for`-loops, which contain a call to `tvt`. All four nestings consist of three loops running through SDR arcs or SDR points in $SDR(A)$, and three loops running through arcs or SDR points in $SDR(B)$. The number of SDR arcs and SDR points in each loop is proportional to n. So the number of calls to `tvt` must be proportional to n^6. The fact that there are four of these nested structures has no influence on the time complexity of the method; it only matters a constant factor.

We found a complexity of $n^{5.90}$, so there is a slight difference. This difference can be explained by the fact that some combinations of SDR points and SDR arcs are left out structurally, as explained in section 4.1.

The improved method performs better when the compared objects are more complex. This can be explained as follows. The method uses distance inequalities to skip combinations of SDR points that cannot coincide with SDR arcs. When a polyhedron becomes more complex, the SDR arcs in its SDR become shorter. Then the chance is smaller that two SDR points of another SDR can fit between two of these SDR arcs. So the more complex the objects, the more combinations can be skipped. In [1] it is shown that the time complexity of this method is $n^{4.5}$, which agrees reasonably well with the value of our fitted curve.

6.2 Mixed volume measure

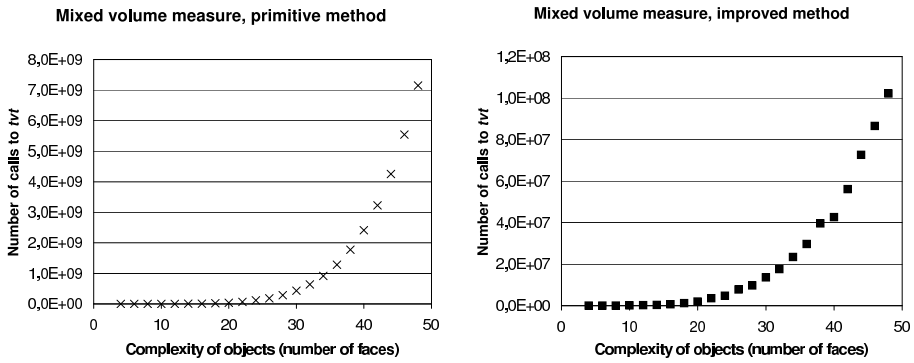


Figure 6.4: Performance of methods for the mixed volume measure. Primitive method (left) and improved method (right).

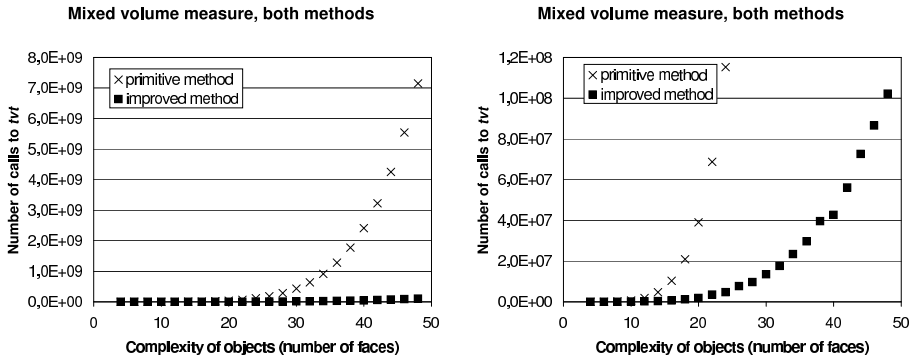


Figure 6.5: Performance of methods for the mixed volume measure. Primitive and improved method combined. Large scale on the y-axis (left) and smaller scale (right).

We did the same performance tests for the mixed volume measure. The results are shown in figure 6.4 and 6.5. The graphs are similar to those for the volume measure. However, as may be expected, the number of calls to `tvt` is lower. For example, for polyhedra with 12 faces, the primitive method needs 1,9 million calls to `tvt` (31 min) and the improved method needs 192.000 (3 min).

Curve fitting The left graph of figure 6.6 shows the performance of the methods for the mixed volume measure on a logarithmic scale. The data points of both methods lie again approximately on a straight line. Again we fitted curves, which are shown by the right graph of figure 6.6. The functions of these curves are:

- $y = 0.71x^{5.95}$ for the primitive method;

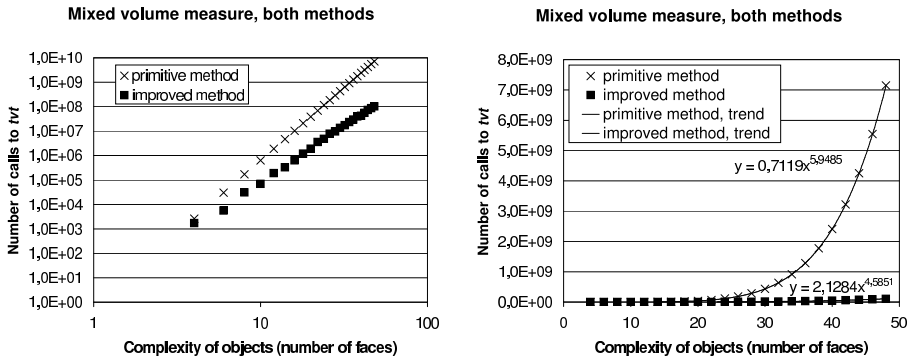


Figure 6.6: Performance of methods for the mixed volume measure. Primitive and improved method combined on a logarithmic scale (left) and on a linear scale with fitted curves (right).

- $y = 2.13x^{4.59}$ for the improved method.

Concluding, the time complexity of both methods for the mixed volume is approximately equal to their complexity for the volume measure. These complexities are roughly proportional to n^6 for the primitive method and $n^{4.5}$ for the improved method. We didn't find a satisfying explanation for this found complexity yet; it will be left as future work.

The primitive and improved methods perform a constant factor better for the mixed volume measure, because for this method the parallelness conditions are more strict, so less combinations need to be checked.

Data structure optimisation In our implementation, the distances between SDR points and SDR arcs are stored in ordinary tables. The main loop runs through all combinations of points and arcs and uses distance inequalities for every combination to determine whether the combination can be skipped or not. Because usually many combinations can be skipped, many iterations in the `for`-loops are done without a call to `tvt`.

To avoid this, we also tried another implementation. This implementations has a more sophisticated data structure, in which SDR points and SDR arcs are sorted in several ways. The main loop uses this ordering to efficiently yield combinations of SDR points and SDR arcs that cannot be skipped. This saves iterations in `for`-loops.

We tested this implementation and found that this saving became significant with polyhedra of more than 24 faces. However, we chose to abandon the idea, because it is more complex and it has no influence on the number of calls to `tvt` needed.

Summary The improved method works well. While the complexity of the primitive method is n^6, the complexity of the improved method is $n^{4.5}$.

Chapter 7

Conclusion

The shape of two objects can be compared by a computer program using similarity measures. The family of similarity measures based on Minkowski addition provides a well-founded way to do this. We described two variants of these measures. The first variant uses the volume of the Minkowski sum, and the second variant uses the mixed volume of the Minkowski sum.

Current implementations have the disadvantage that they are very time-consuming: their time complexity is proportional to n^6, where n is proportional to the complexity of the polyhedra. We developed, implemented and tested improved algorithms for both variants. These algorithms proved to reduce the time complexity to $n^{4.5}$.

Although this is a significant improvement, a time complexity of $n^{4.5}$ is still very high. For example, simple polyhedra with 12 faces still need 3 minutes to compare on current home PCs, using the mixed volume measure. Comparing using the volume measure takes 31 minutes. Even for these simple polyhedra, the time needed to compare is far beyond usable. Particularly, when one polyhedron needs to be compared to a complete *database* of polyhedra, then fast comparison is important. So the use of these improved methods is still very limited.

7.1 Future work

We found a time complexity of $n^{4.5}$ for both variants of similarity measures computed by the improved algorithm. That the volume measure apparently has this time complexity still asks for an explanation.

The method presented in this thesis skips combinations using distance inequalities. These inequalities take distances along the surface in account, but not angles. A further improvement would be to implement an algorithm that also takes angles in account.

A major disadvantage of the method is that it can only be used for convex polyhedra. It would be an improvement if the method could be altered such that it also allows for polyhedra that are not convex.

Furthermore, it is not really known how to apply similarity measures based on Minkowski addition. An application might be object recognition: a computer program compares an object to a database of objects, and possibly finds the

object that is most similar.

The similarity of two convex polyhedra is expressed in one similarity value. However, there are uncertainties about this similarity value:

- Has it the power to find the correct object in a database, when we compare to another object that is not an exact duplicate?
- What does it say about the similarity of two objects when only their convex hulls are compared?
- What does it say about the similarity of two objects when they are approximated by polyhedra with very few faces?

Thus, the method can be further improved and it needs more testing in order to determine how well similarity measures based on Minkowski additions can be applied in practice.

Bibliography

[1] Henk Bekker and Axel Brink. Reducing the time complexity of minkowski-sum based similarity calculations by using geometric inequalities. In *Proceedings of the international conference on computational science and its applications (ICCSA 2004)*, pages 31–40, Perugia, Italy, 2004.

[2] Henk Bekker and Jos B.T.M. Roerdink. Calculating critical orientations of polyhedra for similarity measure evaluation. In *Proceedings of the 2nd annual IASTED international conference on computer graphics and imaging*, pages 106–111, Palm Springs, California USA, February 1999.

[3] P.M. Gruber and J.M. Wills, editors. *Mixed volumes*, chapter 1.2. Elsevier science publishers B.V., 1993. Chapter written by J.R. Sangwine-Yager.

[4] Jos B.T.M. Roerdink and Henk Bekker. Similarity measure computation of convex polyhedra revisited. *Digital and image geometry: advanced lectures*, pages 389–399, 2001.

[5] R. Schneider. *Convex Bodies. The Brunn-Minkowski Theory.* Cambridge University Press, Cambridge, 1993.

[6] A.V. Tuzikov, Jos B.T.M. Roerdink, and H.J.A.M. Heijmans. Similarity measures for convex polyhedra based on minkowski addition. *Pattern Recognition*, 6(33):979–995, 2000.

[7] R.C. Veltkamp. Shape matching: Similarity measures and algorithms. Technical report UU-CS-2001-03, Utrecht University, January 2001.