Continued Work on the Computation of an Exact Arrangement of Quadrics

Michael Hemmer* Sebastian Limbach* Elmar Schömer†

Abstract

We present an exact and complete algorithm for the computation of all edge cycles bounding the faces of a three-dimensional arrangement of algebraic surfaces of degree two (quadrics). The algorithm is based on the implementation by Hemmer et al. [9] which computes the adjacency graph of the arrangement. However, this graph is just the set of vertices and their connectivity along the edges. We enhance each vertex of the graph by a description of its local neighborhood in a so-called environment map, which finally enables the initialization of all edge cycles in the arrangement. This is a major step towards the computation of the full arrangement. The implementation is complete up to a few special cases, and covers several variants in order to compute the environment map.

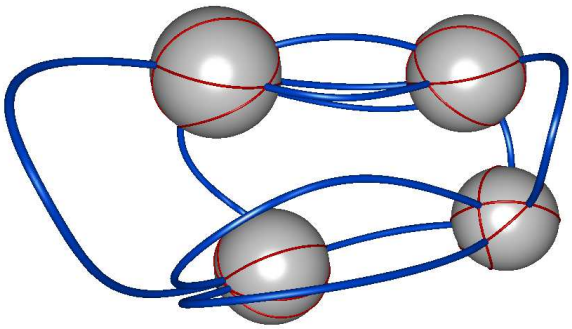

Figure 1: Four environment sphere maps and the corresponding edges of the 3-dimensional arrangement.

1 Introduction

We aim for the computation of the three-dimensional arrangement $\mathcal{A}(\mathcal{Q})$ induced by a given set $\mathcal{Q}$ of quadric surfaces, or quadrics for short. A *quadric* Q_f is given by a trivariate polynomial $f \in \mathbb{Q}[x, y, z]$ of degree 2. Quadrics cover a couple of common surfaces such as spheres, ellipsoids, cones, cylinders, hyperboloids, planes, and double planes. The set of input surfaces may also cover simple rational planes. The *arrangement* $\mathcal{A}(\mathcal{Q})$ is the decomposition of $\mathbb{R}^3$ by the surfaces into cells of dimension 0 (vertices), 1 (edges), 2 (faces) and 3 (volumes) [8]. Arrangements are ubiquitous in computational geometry and can be applied to many problems, for instance, they can serve as a fundamental first step for CSG operations. However, existing inexact, floating point based implementations suffer from rounding errors which can lead to ruinous effects. In contrast, existing exact and complete methods are often not efficient enough for actual application (e.g. [3]). Our goal is to close this gap by following an approach that is exact and complete by design, on the one hand, and efficient enough to be used in practice, on the other hand.

On this way a major goal was recently reached: Hemmer et al. [4, 9] presented an implementation for the computation of the adjacency graph $\mathcal{G}(\mathcal{Q})$. The approach is based on Dupont et al. [5, 10], which provides an exact parameterization of the appearing intersection curves. With these parameterizations, the approach represents the intersection points of quadrics by the exact parameter values with respect to the intersection curves they lie on. Therefore, it is possible to sort the points along the curve, which enables the initialization of the adjacency graph.

Obviously, the adjacency graph $\mathcal{G}(\mathcal{Q})$ does not contain enough information in order to represent the full arrangement as it just stores all vertices and edges of the arrangement $\mathcal{A}(\mathcal{Q})$. The missing parts are the two and three dimensional cells, that is, the faces and volumes of the arrangement and, in particular, their incidences at each vertex. Therefore, we investigate the local neighborhood of each vertex. It is represented by a two dimensional arrangement which is conceptually embedded on a sphere. We call this the *environment map*[1] of a vertex.

A vertex on the sphere corresponds to an incoming edge of $\mathcal{G}(\mathcal{Q})$ and a halfedge on the sphere corresponds to an incident halfface of $\mathcal{A}(\mathcal{Q})$. That is, each face of $\mathcal{A}(\mathcal{Q})$ is bounded by a sequence of halfedges whereas each of these halfedges can be identified with a halfedge of an environment map. This is similar to the existing approach for the arrangement of planes in [6, 7].

*Max-Planck-Institut für Informatik, 66123 Saarbrücken, {hemmer,slimbach}@mpi-inf.mpg.de

†Institut für Informatik, Johannes Gutenberg-Universität Mainz, schoemer@uni-mainz.de

[1] Here, the term "environment map" is not related to its usage in the context of computer graphics.

A more detailed discussion of the environment map is given in Section 2. With these details, Section 3 shows how the edge-cycles in the three dimensional arrangement are initialized. Section 4 presents the available implementations, which are compared in Section 5.

The algorithm is implemented within the context of the EXACUS[2] project.

2 The environment map

Let v be a vertex of $\mathcal{A}(\mathcal{Q})$ and $\mathcal{Q}_v \subseteq \mathcal{Q}$ be the set of quadrics v lies on. Obviously, the quadrics in $\mathcal{Q}_v$ are the only quadrics that influence the local neighborhood. In order to obtain a representation of this local neighborhood the idea is to intersect $\mathcal{Q}_v$ with a sufficiently small sphere $\mathcal{S}$ centered at v. This results in a two-dimensional arrangement $\mathcal{A}(\mathcal{C})$ induced by the intersection curves $\mathcal{C} = \{\mathcal{C}_i = \mathcal{Q}_i \cap \mathcal{S} \mid \mathcal{Q}_i \in \mathcal{Q}_v\}$. The arrangement $\mathcal{A}(\mathcal{C})$ consists of cells of dimension 0, 1 and 2 on the surface of $\mathcal{S}$.

Obviously, there exists an r_0 such that every arrangement on a sphere $\mathcal{S}_r$ with radius $r \leq r_0$ has the same topology as the arrangement on $\mathcal{S}_{r_0}$. The fundamental question is how to detect that the sphere is actually small enough. Obviously, the arrangement does not change once the sphere is small enough with respect to every single quadric, every pair of quadrics and every triple of quadrics out of $\mathcal{Q}_v$. The following three constrains ensure this.

1. If v is the apex of a cone, or v lies in the common line of intersecting planes, the topology of the intersection of this surface with $\mathcal{S}$ will not change at all. For any other quadric, the intersection with the interior of $\mathcal{S}$ must be homeomorphic to a disk, that is, the quadric induces exactly one intersection curve homeomorphic to a circle on the sphere.

2. The intersection of every pair of quadrics (intersection curves) of $\mathcal{Q}_v$ intersected with the interior of $\mathcal{S}$ must be homeomorphic to a line segment (or two intersecting line segments, in case of a nodal quartic with v at its nodal point).

3. The intersection of triples of quadrics (vertices) of $\mathcal{Q}_v$ intersected with the interior of $\mathcal{S}$ of correct size must be v itself.

Note that the bounding object does not need to be a sphere, for instance, it could be replaced by a box or an even simpler bounding object. Therefore, we decided to follow the generic programming paradigm [1] using the C++ template mechanism and kept the code generic in the way the local neighborhood is constructed and represented. We can instantiate the code by any type that fulfills a certain set of requirements. Such a set of of requirements is called a *concept*. A type that fulfills all these requirements is called a *model* of the concept. More precisely, we introduced the concept `EnvironmentMap_2`, which we briefly discuss next.

Given a vertex v a model of `EnvironmentMap_2` is supposed to represent the local neighborhood of v by a two dimension arrangement that is conceptually embedded on a sphere. The arrangement is stored in a DCEL data structure consisting of *halfedges*, *vertices* and *faces*. Since a vertex on the sphere corresponds to an incoming edge of $\mathcal{G}(\mathcal{Q})$, the model must identify each vertex with the corresponding outgoing edge of $\mathcal{G}(\mathcal{Q})$. Similarly, every halfedge must be identified with the corresponding quadric. This is complemented by the so-called *alignment flag*, which is used to associate each halfedge with either the inside or the outside of the corresponding quadric. These informations are needed later on, since they allow the identification of the next edge in an 3D-edge cycle. For more details, which are in particular relevant in case of degenerated quadrics (e.g. two intersection planes), we refer to [11].

Note that we actually do not enforce that the local neighborhood is constructed via the intersection with a concrete bounding object. We will use this fact in Section 4, which provides a model that computes the local neighborhood based on the tangent planes of all quadrics in $\mathcal{Q}_v$ at v. This will turn out to be very efficient but it is not applicable in all cases. Therefore, we defined the additional concept `EnvironmentMapFilter_2`. It has the same requirements as `EnvironmentMap_2` and in addition it requires a static function that takes the vertex as input and returns a boolean flag indicating whether the filter is applicable or not.

3 Initializing the edge cycles

The environment maps provide us with all information the algorithm needs for traversing an edge cycle bounding a halfface[3].

We start at an arbitrary halfedge of the environment map of any vertex. The associated surface and the indication whether the halfedge belongs to the inside or the outside of the surface (the alignment flag) correspond uniquely to one of the halffaces adjacent to the halfedge. From the endpoint of our halfedge we travel along the associated edge of $\mathcal{G}(\mathcal{Q})$, reaching a new vertex on the next environment map. We then pick the next outgoing halfedge by comparing the associated surface and the alignment flag and continue

[2] "Efficient and exact algorithms for curves and surfaces", `http://www.mpi-inf.mpg.de/projects/EXACUS/`.

[3] In analogy to the DCEL of our environment maps, we divide each face into two halffaces.

until we reach our starting edge again. The creation of all edge cycles is implemented accordingly.

4 Environment map models

Our implementation consists of two different approaches for the computation of an environment map:

Sphere map model: The first intuitive idea for constructing the environment map of a vertex v is the actual computation of an arrangement on a spherical surface. This results in an arrangement $\mathcal{A}(\mathcal{C})$ of intersection curves of degree up to four. Berberich et al. [2] developed a general framework for sweeping a set of curves embedded on a 2D-parametric surface. An implementation is available in the `Arrangement_on_surface_2` package of CGAL[4]. The sphere map model uses the `Arrangement_on_surface_with_history_2` (AOS) class of this `Arrangement_on_surface_2` package for the computation of the two-dimensional arrangement $\mathcal{A}(\mathcal{C})$ together with a construction history. It is a model of the `EnvironmentMap_2` concept.

The AOS class is capable of constructing the arrangement of intersection curves induced by quadric surfaces on a given base surface. So it remains to construct and to initialize a sphere of the correct size as a base surface, to convert the arrangement obtained from the AOS class into our own DCEL representation, to connect all vertices and halfedges of $\mathcal{A}(\mathcal{C})$ with the corresponding edges and surfaces from the adjacency graph and finally to initialize the alignment flag.

For the matching of the vertices of $\mathcal{A}(\mathcal{C})$ with the outgoing edges at v, we intersect the edges with $\mathcal{S}$ to obtain a set of intersection points on each edge. We then use a matching algorithm based on so-called bigfloat interval (BFI) arithmetic for matching approximations of the positions of the vertices with approximations of the positions of the intersection points. A bigfloat is a multi precision floating point number, that is, the bitsize of its representation is only limited by the available memory. A BFI is an interval number type which uses bigfloats as interval borders. See [9, 11] for more details about the application of BFIs in our algorithms.

For the matching of the halfedges with the corresponding quadrics we only have to look at the construction history of the AOS class. It remains to initialize the alignment flag on each halfedge. For this flag we usually take the midpoint of the successor of the halfedge and check if the point is inside or outside the quadric. Alternatively, if the next halfedge belongs to the quadric we want to check, we take the midpoint of the successor of the twin halfedge and negate the result. If we are in the rare situation that both successors are part of the quadric, we go the other way around: We take any point on $\mathcal{S}$ that is not part of the quadric, compute whether it is at the inside or the outside of it, and check if the point is inside the face bounded by the halfedge, or in any face which is a hole inside the face bounded by the halfedge.

Nef filter model: The computation of an arrangement on a concrete sphere can be a very costly and complex task. At the same time, such arrangements can get simpler if we consider an infinitesimally small sphere. In such a case, the surfaces of all general quadrics are equivalent to their tangential planes at the position of v. To avoid the high algebraic degrees occurring in an exact representation of the normal vectors of these tangential planes, we approximate all vectors using sufficiently precise BFI approximations. The arrangement $\mathcal{A}(\mathcal{C})$ exists and is unambiguous as long as all tangential planes exist and all triples of normal vectors of the involved tangential planes are linearly independent. Therefore, the drawback of this approach is that it can not be applied in general. But the overall performance benefits from the application of such a model as a filter.

The Nef filter model is implemented using the `Nef_polyhedron_S2` class from the `Nef_S2` package of CGAL. This class is capable of representing two-dimensional Nef polyhedrons (arrangements of halfspaces bounded by lines in the plane), embedded on a sphere [6, 7]. Our implementation is a model of the `EnvironmentMapFilter_2` concept.

We first construct the sufficiently precise BFI approximations of all normal vectors of the tangential planes. Afterwards, we can computed the two-dimensional arrangement $\mathcal{A}(\mathcal{C})$ on $\mathcal{S}$ using the `Nef_polyhedron_S2` class. Next, we convert the arrangement into our own DCEL representation. For each new halfedge, we determine the corresponding surface, as well as the value of the alignment flag. We obtain the corresponding surface the halfedge is embedded in by matching the normal vector of the plane induced by the origin and the halfedge with the normal vectors of all tangential planes of the surfaces. Whether the normal vectors have the same or opposite direction indicates the belonging of the halfedge to the inside or the outside of the corresponding surface.

For each vertex of $\mathcal{A}(\mathcal{C})$, we identify the intersection curve the vertex is associated with, and we determine the corresponding edge of $\mathcal{G}(\mathcal{Q})$. We obtain the supporting curve by looking at the two associated surfaces of the halfedges adjacent to the vertex of $\mathcal{A}(\mathcal{C})$. Finally, we determine the corresponding edge we associate with each vertex by matching the direction from v to the vertex with the tangent vector of the supporting curve at v.

[4] "Computational Geometry Algorithms Library", `http://www.cgal.org`.

5 Benchmarks

We compared the runtime of our models by a series of benchmarks. First, we directly compared the performance of our sphere map implementation to the implementation of the Nef filter model on a set of quadrics with random coefficients. The results of this benchmark are presented in table 1.

$\|\mathcal{Q}\|$	sphere map $[s]$	filter $[s]$	adja. graph $[s]$
3	1.3986	0.204885	0.099786
4	18.2372	0.786321	0.244173
5	86.3439	1.68017	0.512816
6	237.234	1.74667	0.337523
7	352.006	2.91448	0.535007
8	580.153	4.79391	0.797845
9	853.958	7.06226	1.14679
10	1168.73	8.6739	1.50417

Table 1: Comparison of the sphere map model and the Nef filter model on random quadrics.

In such a random situation, the probability of having degenerated environments at vertices is practically zero. Therefore, we can always apply the faster Nef filter implementation. One observes that the gain in runtime for the Nef filter potentiates with respect to the increasing number of vertices of the three dimensional arrangement.

$\|\mathcal{Q}\|$	sphere map $[s]$	filter $[s]$	adja. graph $[s]$
3	60.2265	5.62893	1.3789
4	165.422	19.6249	5.02124
5	404.945	38.7777	12.8201
6	971.758	73.1718	25.1551

Table 2: Comparison of the sphere map model and the filter model on an arrangement with one degenerated vertex environment.

For the second benchmark, we compared the performance of the general sphere map model implementation with a filtered model implementation, that decides whether to apply the sphere map or the Nef filtered implementation for each vertex. We tested the implementation on an increasing number of quadrics that all intersect tangentially in one vertex. At this vertex, the filter cannot apply the Nef filter and has to fall back to the slower general model. Table 2 shows the results of the second benchmark. As in the previous benchmark, the sphere map implementation performs worse than the filtered version. The time to take the decision whether or not to apply the filter and the overhead of initializing the edge cycles is negligible compared to the runtimes of the adjacency graph and the environment map computation.

6 Conclusion and outlook

We presented an algorithm, a concept and two different implementations for the computation of all edge cycles bounding the faces of the three-dimensional arrangement of a set of quadrics. For the complete initialization of the faces, however, we still have to detect the holes of each face, that is, information about the nesting of the faces. For that, the idea is to construct intersection curves on the quadrics and to intersect them with the other quadrics of the arrangement. The order of the intersections provides the required information about the nesting of the faces.

References

[1] M. H. Austern. *Generic Programming and the STL.* Addison-Wesley, 1998.

[2] E. Berberich, E. Fogel, D. Halperin, K. Mehlhorn, and R. Wein. Sweeping and maintaining two-dimensional arrangements on surfaces: A first step. In *Proc. ESA 2007*, LNCS, pages 645–656, Eilat, Israel, 2007. Springer.

[3] G. E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Second GI Conference on Automata Theory and Formal Languages*, volume 33 of *LNCS*, pages 134–183, 1975.

[4] L. Dupont, M. Hemmer, S. Petitjean, and E. Schömer. Complete, exact and efficient implementation for computing the adjacency graph of an arrangment of quadrics. In *Proc. ESA 2007*, LNCS, pages 633–644, Eilat, Israel, 2007. Springer.

[5] L. Dupont, D. Lazard, S. Lazard, and S. Petitjean. Near-optimal parameterization of the intersection of quadrics: I+II+III. *Journal of Symbolic Computation*, 2008.

[6] M. Granados, P. Hachenberger, S. Hert, L. Kettner, K. Mehlhorn, and M. Seel. Boolean operations on 3D selective Nef complexes. In *Proc. ESA 2003*, LNCS, pages 654–666, Budapest, Hungary, 2003. Springer.

[7] P. Hachenberger. *Boolean Operations on 3D Selective Nef Complexes.* PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, 2006.

[8] D. Halperin. Arrangements. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 24, pages 529–562. Chapman & Hall/CRC, 2nd edition, 2004.

[9] M. Hemmer. *Exact Computation of the Adjacency Graph of an Arrangement of Quadrics.* PhD thesis, Johannes Gutenberg-Universität, Mainz, Germany, 2008.

[10] S. Lazard, L. M. Peñaranda, and S. Petitjean. Intersecting quadrics : An efficient and exact implementation. In *Proc. 20th ACM Symposium on Computational Geometry*, Brooklyn, NY, 2004.

[11] S. Limbach. Continued Work on the Computation of an Exact Arrangement of Quadrics. Master's thesis, Universität des Saarlandes, Saarbrücken, Germany, 2008.