Arrangements on Parametric Surfaces II: Concretizations and Applications

Eric Berberich, Efi Fogel, Dan Halperin,
Michael Kerber and Ophir Setter

Abstract. We describe the algorithms and implementation details involved in the concretizations of a generic framework that enables exact construction, maintenance, and manipulation of arrangements embedded on certain two-dimensional orientable parametric surfaces in three-dimensional space. The fundamentals of the framework are described in a companion paper. Our work covers arrangements embedded on elliptic quadrics and cyclides induced by intersections with other algebraic surfaces, and a specialized case of arrangements induced by arcs of great circles embedded on the sphere. We also demonstrate how such arrangements can be used to accomplish various geometric tasks efficiently, such as computing the Minkowski sums of polytopes, the envelope of surfaces, and Voronoi diagrams embedded on parametric surfaces. We do not assume general position. Namely, we handle degenerate input, and produce exact results in all cases. Our implementation is realized using CGAL and, in particular, the package that provides the underlying framework. We have conducted experiments on various data sets, and documented the practical efficiency of our approach.

Mathematics Subject Classification (2010). Primary 68U05; Secondary 14Q10.

Keywords. computational geometry, arrangement of curves, parametric surface, CGAL, robust geometric computing, Voronoi diagram, lower envelope, Gaussian map, quadric, ring Dupin cyclide.

1. Introduction

Given a finite collection $\mathcal{C}$ of geometric objects (such as lines, planes, or spheres) the *arrangement* $\mathcal{A}(\mathcal{C})$ is the subdivision of the space where these objects reside

This work has been supported in part by the Israel Science Foundation (grant no. 236/06), by the German-Israeli Foundation (grant no. 969/07), and by the Hermann Minkowski–Minerva Center for Geometry at Tel Aviv University.

into cells as induced by the objects in $\mathcal{C}$. In this paper we concentrate on classes of arrangements embedded on two-dimensional orientable parametric surfaces. An arrangement of these classes consists of cells of dimension 0 (*vertices*), 1 (*edges*), and 2 (*faces*). We describe how such instances of arrangements can be produced, and how instances of such classes can be robustly constructed, maintained, and manipulated using exact computation. We divide the presented classes into two categories distinguished by the type of arithmetic required for their implementation, as the type of arithmetic plays a significant role in exact computation. The classes of arrangements we present in this paper are concretizations of a new generic framework [7, 8]. Relying on such a framework in general is advantageous for two reasons: (i) new concretizations are relatively easily produced, and (ii) all functionality available for the framework immediately becomes available for the new concretization. Our case is no exception. Generating a new concretization amounts to the provision of two compact components, called *geometry traits* and *topology traits*, that must satisfy two minimal sets of requirements, respectively. The geometry-traits class handles the specific shape of the surface and the family of curves that induce the embedded arrangement. The topology-traits class provides topological information related to the embedding surface. The framework itself is independent of the type of the inducing curves and of the type of the embedding surface. Typically, a single topology-traits class can be coupled with several different geometry-traits classes to produce new classes of arrangements. This framework setup and the separation between geometric and topological aspects of the two-dimensional subdivision enable the convenient production of new classes of arrangements.

CGAL, the Computational Geometry Algorithms Library,[1] provides a generic and robust, yet efficient, implementation of widely used geometric data-structures and algorithms. The library contains models of various geometric *Kernel* concepts [27, 42]. Each model defines several types of constant-size non-modifiable geometric primitive objects (such as points and lines) and predicates and operations on objects of these types. The library also contains many fundamental geometric data-structures and algorithms, the implementation of which is based on traits concepts, often already modelled by CGAL's geometry kernels. The `Arrangement_on_surface_2` package [60] of CGAL implements the aforementioned framework and various topology-traits and geometry-traits components. It is released as part of CGAL version 3.4 and higher. It supports the construction and maintenance of arrangements embedded on two-dimensional orientable parametric surfaces in three dimensions, and is the result of extending the former `Arrangement_2` package of CGAL. The implementation of the extended package maximizes code reuse by generalizing prevalent algorithms, such as the sweep-line and zone-traversal algorithms. The `Arrangement_on_surface_2` package comes with many useful operations on arrangements, such as point location, insertion of curves, removal of

[1] `http://www.cgal.org`

curves, and overlay computation, which enable its application for various geometric problems.

In this work, we report on two concretizations of the package. First, we consider arrangements induced by arcs of great circles, also known as geodesic arcs, embedded on the sphere [32, 33]. Such arrangements can be computed efficiently, since all calculations are performed with (exact) rational arithmetic. We also discuss numerous applications for geodesic arcs, that is, Minkowsi sum computation for three-dimensional polytopes, envelope computation, and Voronoi diagram construction.

As second class of concretizations, we look at arrangements embedded on elliptic quadrics [7] and on the Dupin cyclide [11]. This covers many "natural" surfaces, such as spheres, ellipsoids, and cylinders (quadrics), and tori (cyclides). In both cases, the curves on the surface are given by the intersections of the surface and other algebraic surfaces of arbitrary degree. Providing suitable models for the topology-traits and geometry-traits concepts is much more involved compared to geodesics on the sphere, because the topology of the surface is more complicated and algebraic arithmetic is necessary to deal with the curves in an exact manner. We therefore concentrate on the details of these models, and only sketch possible applications.

The software described in this paper rigorously adheres, as does CGAL in general, to the *generic programming* paradigm [4], making extensive use of C++ class-templates and function-templates. The generic-programming paradigm uses a formal hierarchy of abstract requirements on data types referred to as *concepts*, and a set of components that conform precisely to the specified requirements, referred to as *models*. Concepts and models correspond to expectations for template parameters and classes used to instantiate them, respectively. The main module of the framework is implemented as a class-template called `Arrangement_on_surface_2` parameterized by the geometry-traits and the topology-traits template parameters. A concretization of a specific class of arrangements is realized through the instantiation of the `Arrangement_on_surface_2` class template with geometry and topology-traits classes that model the corresponding concepts.

Our implementation is robust, as it is designed to handle all degeneracies,[2] and exact, as all underlying geometric operations follow the exact geometric computation paradigm [61].

Related Work. Arrangements in general are fundamental data-structures in computational geometry, and have been intensively studied for several decades. For related work on arrangements see Section 1 of the companion paper [8] and the references therein. Arrangements of linear objects in the plane in particular have many theoretical and practical applications (see, e. g., [2, 17, 31, 38]), and many of them have analogous applications, where the embedding plane and the inducing

[2]The implementation of the `Arrangement_on_surface_2` package currently lacks support for isolated points and curves on the boundary of the parameter space. Thus, some special input is not handled in software yet.

linear curves are substituted for the sphere and geodesic arcs on it. Arrangements of geodesic arcs on the sphere are useful in their own right. For example, they can be used to represent Gaussian maps of polytopes, which in turn can be used to compute Minkowski sums of polytopes; see Section 2.2.

Arrangements on quadrics, have previously been discussed [10]. However, the approach constructs two arrangements by considering a parallel projection of intersection curves induced by other quadrics onto the xy-plane, and the connection between the two is not implemented.

Outline. For the rest of the article we assume the reader to be familiar with the framework introduced in the companion paper [8]; especially with its notion and the *geometry-traits* and *topology-traits* concepts. We provide implementation details and application samples of our first non-planar concretizations: Section 2 covers geodesic arcs on the sphere and Section 3 presents arrangements on ring Dupin cyclides and elliptic quadrics. We conclude and present future-research directions in Section 4.

This paper unifies and summarizes our capabilities in concretizing of the `Arrangement_on_surface_2`. Preliminary results along these lines have been previously published in a series of conference and proceedings papers [7, 9, 11, 32, 33]. We here present their matured versions.

2. A Concretization with Rational Arithmetic and its Application

In this section we concentrate on the specific algorithms and implementation details involved in the exact construction and maintenance of arrangements induced by geodesic arcs embedded on the sphere (centered at the origin), and on applications of such arrangements, that is, the exact construction of Minkowski sums of polytopes, the computation of envelopes projected onto the sphere, and the exact construction of Voronoi diagrams on the sphere, the bisectors of which are geodesic arcs. The class of Voronoi diagrams includes the subclass of Voronoi diagrams of points and its generalization, power diagrams on the sphere, also known as Laguerre Voronoi diagrams.

There is an analogy between this class of arrangements and the class of planar arrangements induced by linear curves (i. e., segments, rays, and lines), as properties of linear curves in the plane often, but not always, hold for geodesic arcs on the sphere. For example, given any two non-antipodal points on the sphere there exists a unique great circle connecting the two points. When computing exact arrangements of non-linear objects (e. g., elliptic arcs on the plane) we usually have to employ number-types that can handle algebraic numbers in a certified manner (see, e. g., [46]), which typically incurs significant running-time penalties. The ability to robustly construct arrangements of geodesic arcs on the sphere, and robustly apply operations on them using only (exact) rational arithmetic (as is the case with planar arrangements induced by linear curves) is a key property that enables an efficient implementation.

2.1. Arrangements of Geodesic Arcs Embedded on the Sphere

We use the following parameterization of the unit sphere: $\Phi = [-\pi + \alpha, \pi + \alpha] \times [-\frac{\pi}{2}, \frac{\pi}{2}]$, $\phi_S(u, v) = (\cos u \cos v, \sin u \cos v, \sin v)$, where α must be substituted with an angle, the arctangent of which is rational, and defaults to 0 when the class is instantiated (at compile time).[3] The equator curve, for example, is given by $\gamma(t) = (\pi(2t-1)+\alpha, 0)$, for $t \in [0, 1]$. This parameterization induces two contraction points $p_s = (0, 0, -1) = \phi_S(u, -\frac{\pi}{2})$ and $p_n = (0, 0, 1) = \phi_S(u, \frac{\pi}{2})$, referred to as the south and north poles, respectively, and an identification curve $\{\phi_S(\pi+\alpha, v) \mid -\frac{\pi}{2} \leq v \leq \frac{\pi}{2}\}$, as $\phi_S(-\pi + \alpha, v) = \phi_S(+\pi + \alpha, v)$ for all v (which coincides with the opposite Prime (Greenwich) Meridian when $\alpha = 0$).

We developed the topology traits to support not only arrangements of geodesic arcs, but any type of curves embedded on the sphere parameterized as above, without compromising with the performance of the operations gathered in the traits class. We hope that this topology traits will come in handy in the future for constructing and maintaining arrangements induced by types other than geodesic arcs, such as general circular arcs, which appear in arrangements induced by intersections of spheres embedded on the sphere [15, 18, 39]. The topology traits initializes the eDcel to have a single bounded face, the embedding of which is the entire sphere. It is designed to retain the variant that this face always contains the point $p = (u_{\max}, v_{\max} - \varepsilon), \varepsilon > 0$, where ε is sufficiently small, such that the definition of the face does not depend on the choice of ε during modifications the arrangement may undergo; see [8, §3.2]. The topology traits maintains a search structure of vertices that coincide with the contraction points or lie on the identification arc; see [8, §A]. Flexible parameterization reflected in the ability to substitute α allows the user to choose a parameterization that induces as few as possible vertices on the boundary of the parameter space, reducing the time it takes to, insert them into, and find them in the search structure.

The geometry-traits class for geodesic arcs on the sphere is parameterized with a geometric kernel [42] that encapsulates the number type used to represent coordinates of geometric objects and to carry out algebraic operations on those objects. The implementation handles all degeneracies, and is exact as long as the underlying number type supports the arithmetic operations $+$, $-$, $*$, and $/$ in unlimited precision over the rationals, such as the one provided by Gmp, the Gnu Multi-Precision bignum library.[4] No other arithmetic operations are required even though the embedding surface is a sphere. We are able to use high-performance kernel models instantiated with exact rational, number-types for the implementation of this geometry-traits class, as exact rational arithmetic suffices to carry out all necessary algebraic operations.

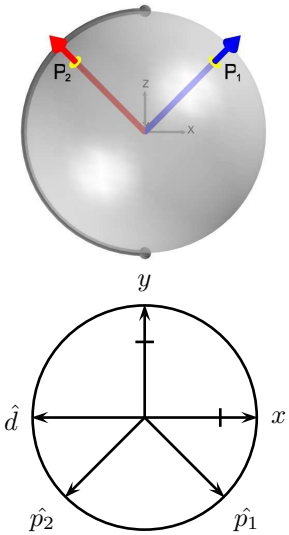
The geometry-traits class defines the point type to be an unnormalized vector in $\mathbb{R}^3$, representing the place where the ray emanating from the origin in the relevant direction pierces the sphere. An arc of a great circle is represented by

[3] The actual template parameters of the class are two integers specifying the arctangent of α.
[4] `http://gmplib.org`

its two endpoints, and by the plane that contains the endpoints (represented as directions) and the origin. The orientation of the plane and the distinction between the source and target points determine which one of the two arcs of the great circle is considered.

The point type is extended with an enumeration type that indicates whether the direction (i) pierces the south pole, (ii) pierces the north pole, (iii) intersects the identification arc, or (iv) is any other direction. An arc of a great circle is extended with three Boolean flags that indicate whether any one of the x, y, z coordinates of the normal to the plane that defines the arc vanishes.[5] Typically, points and curves that are input of geometry-traits operations are projected onto a target plane (one of the three major planes), before processed in a lower dimension. Naturally, planes orthogonal to the plane that defines an input arc must be avoided as target planes. The flags above are used to minimize the number of algebraic operations the geometry-traits operations perform, for example, eliminating invalid target planes. This reduction has a drastic effect on the performance of arrangement operations at the account of a slight increase in space consumption. This representation enables an exact, yet efficient, implementation of all geometric operations required by the geometry-traits concept using exact rational arithmetic, as normalizing directions and planes is completely avoided.

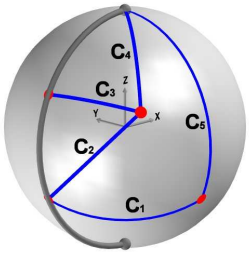
We describe in detail four predicates: **Compare** $\boldsymbol{u}$, **Compare** $\boldsymbol{uv}$, **Compare** $\boldsymbol{u}$ **near boundary**, and **Compare** $\boldsymbol{v}$ **near boundary**; see [8, §4.2] for the complete set of the concept requirements. The first among the four compares two points p_1 and p_2 by their u-coordinates. The concept admits the assumption that the input points do not coincide with the contraction points and do not lie on the identification arc. Recall that points are in fact unnormalized vectors in $\mathbb{R}^3$. We project p_1 and p_2 onto the xy-plane to obtain two-dimensional unnormalized vectors $\hat{p_1}$ and $\hat{p_2}$, respectively. We compute the intersection between the identification arc and the xy-plane to obtain a third two-dimensional unnormalized vector $\hat{d}$. Finally, we test whether $\hat{d}$ is reached strictly before $\hat{p_2}$ is reached, while rotating counterclockwise starting at $\hat{p_1}$. This geometric operation is supported by every geometric kernel of CGAL. In the figure to the right $\hat{d}$ is reached strictly before $\hat{p_2}$ is reached. Therefore, the u-coordinate of p_1 is larger than the u-coordinate of p_2.



The predicate **Compare** $\boldsymbol{uv}$ compares two points p_1 and p_2 lexicographically. It first applies **Compare** $\boldsymbol{u}$ to compare the u-coordinates of the two points. If the u-coordinates are equal, it applies a predicate that compares the v-coordinates of two points with identical u-coordinates, referred to as **Compare** $\boldsymbol{v}$. This predicate first

[5] We encode the three Boolean flags as three bits in order to save space.

compares the sign of the z-coordinates of the two unnormalized input vectors. If they are identical, it compares the squares of their normalized z-coordinates, essentially avoiding the square-root operation.

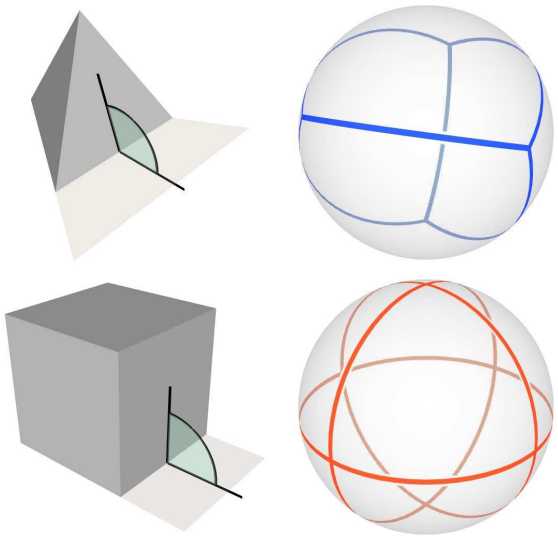


The predicates above accept points, the pre-images of which, lie in the interior of the parameter space. However, there is also a need to lexicographically compare the ends of arcs, the pre-images of which reach the boundary of the parameter space. The predicate **Compare u near boundary** accepts either (i) a point, the pre-image of which lies in the interior of the parameter space, and a curve-end, or (ii) two curve-ends. Such a curve-end is provided by an arc and an index that identifies one of the two ends of the arc, and must coincide with one of the contraction points. The first variant compares the u-coordinate of the input point and a point along the input arc near its given end, whereas the second variant compares the u-coordinates of two points along the input arcs near their respective given ends. Recall, that the u-coordinates of all points along a vertical arc are the same (C_4 and C_5 in the figure above). Thus, we can compare the u-coordinates of two respective arbitrary points on the two vertical arcs that lie inside the parameter space. Evidently, we compare the two vectors perpendicular to the normals to the planes that define the vertical arcs, respectively. For example, the u-coordinate of a point on the arc C_4 near its top end is smaller than the u-coordinate of a point on the arc C_5 near its top end, and in particular it is smaller than the u-coordinate of the bottom end of C_5. The **Compare v near boundary** predicate compares the v-coordinate of two arcs ends, the pre-images of which lie on the same (left or right) side of the boundary of the parameter space. We use the aforementioned **Compare v** predicate to compare the end points. If the points are equal, we compare the normals to the plane that define the arcs. In our example, the left end of C_1 is smaller than the left end of C_2, which is smaller than the left end of C_3.

All the required geometric operations listed in the geometry-traits concept are implemented using only rational arithmetic. Degeneracies, such as overlapping arcs that occur during intersection computations, are properly handled. The end result is a robust, yet efficient, implementation. Armed with the geometry-traits for geodesic arcs on the sphere, we can use all the arrangement machinery to solve a variety of problems involving such arrangements.

2.2. Minkowski Sum of Polytopes

An immediate motivation for constructing arrangements of arcs of great circles embedded on a sphere, is computing the *Gaussian map* of polytopes (bounded convex polyhedra). This representation enables, for example, the efficient computation of Minkowski sums of polytopes by overlaying their Gaussian maps, which in turn, for example, enables the quick detection of collision; see [29] and the references therein.



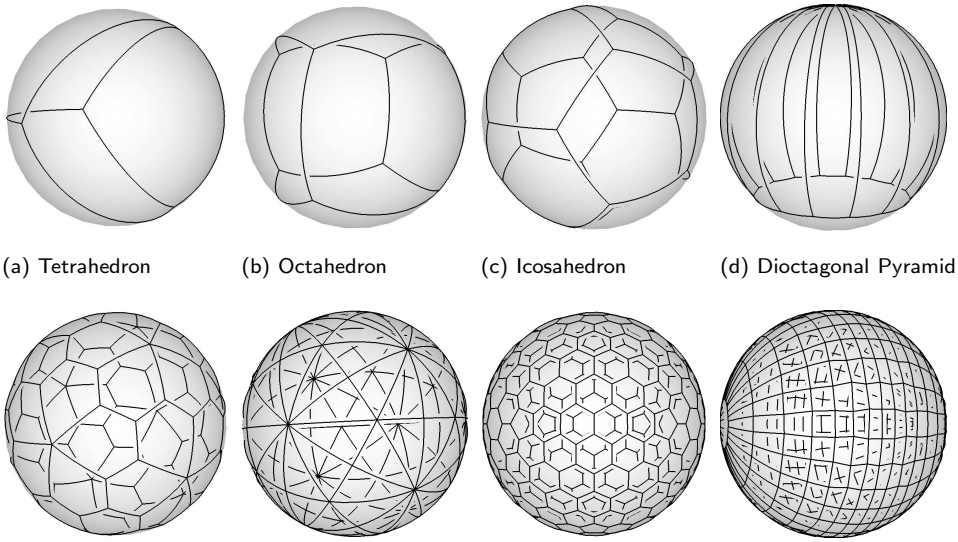
The *Gaussian map* $G = G(P)$ of a compact polytope P in Euclidean three-dimensional space $\mathbb{R}^3$ is a set-valued mapping from P to the unit sphere $\mathbb{S}^2$, which assigns to each point p on the boundary of P the set of outward unit normals to supporting planes to P at p. Thus, the whole of a facet f of P is mapped under G to a single point, representing the outward unit normal to f. An edge e of P is mapped to a (geodesic) segment $G(e)$ on $\mathbb{S}^2$, whose length is easily seen to be the exterior dihedral angle at e. A vertex v of P is mapped by G to a spherical polygon $G(v)$, whose sides are the image under G of edges incident to v, and whose angles are the angles supplementary to the planar angles of the facets incident to v; that is, $G(e_1)$ and $G(e_2)$ meet at angle $\pi - \alpha$ whenever e_1 and e_2 meet at angle α [44]. The above implies that $G(P)$ is an arrangement embedded on the unit sphere. Extending the mapping above, by marking each face $f = G(v)$ of the arrangement with its dual vertex v, enables a unique inverse Gaussian mapping, denoted by G^{-1}, which maps an extended arrangement embedded on the unit sphere back to a polytope boundary. The figure above shows (from left to right and then from top to bottom): a tetrahedron, the Gaussian map of the tetrahedron, a cube, and the Gaussian map of the cube.

We use an arrangement of arcs of great circles embedded on the sphere to maintain the Gaussian map $G = G(P)$ of a compact polytope P in $\mathbb{R}^3$.

2.2.1. Gaussian Map Construction. An input model of a polytope is typically provided as a polyhedral mesh. Constructing the Gaussian map of a model given in this representation is done indirectly. First, the CGAL `Polyhedron_3` [49] data-structure that represents the model is constructed. Then, the Gaussian map is constructed exploiting the accessible incidence relations between the polytope features stored in the `Polyhedron_3` data-structure. Once the construction of the Gaussian map is complete, the `Polyhedron_3` intermediate representation is discarded.

The `Polyhedron_3` data-structure, like the arrangement DCEL, consists of extendible vertices, halfedges, and facets and incidence relations on them. We traverse the `Polyhedron_3` features in a DFS fashion starting at an arbitrary vertex. For each vertex being processed, we visit its incident undiscovered edges and process them, and for each edge being processed, we visit its undiscovered vertex and

process it. When we process a polytope edge e we insert the arc that is the embedding of the dual edge $G(e)$ into the arrangement that represents the Gaussian map. Notice, that all arcs in the Gaussian map of a non-degenerate polytope are strictly less than π, each face of the arrangement is convex, and, naturally, all inserted edges are pairwise disjoint in their interior. These properties allow us to use one of the efficient insertion member-functions supported by the `Arrangement_on_surface_2` data-structure; see, e. g., [28]. These functions accept hints regarding the location and the incidence relations of the inserted arc, and return the newly created feature, which we store in the `Polyhedron_3`. We use this information in consequent insertions of arcs into the Gaussian map as hints. If the new arc to be inserted intersects the identification arc, it is first split at the intersection point into two u-monotone arcs, which are inserted instead. The first arc is inserted into an empty arrangement. For every other arc that is the embedding of a new edge, either one or both of the two incident vertices of the edge exist already when the arc is inserted. These vertices are passed as hints to the insertion function, resulting in an efficient overall process.



(a) Tetrahedron (b) Octahedron (c) Icosahedron (d) Dioctagonal Pyramid

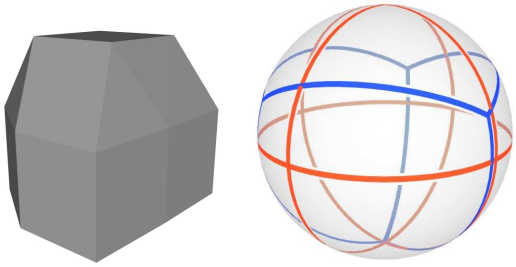
(e) Pentagonal Hexecontahedron (f) Truncated Icosidodecahedron (g) Geodesic Sphere level 4 (h) Ellipsoid like polyhedron

Figure 1: Gaussian maps of various polytopes.

We have created a database of various models of polytopes. Figure 1 depicts, for a small subset of our polytope collection, the Gaussian map of each polytope. Table 1 lists the number of features in the arrangement of geodesic arcs embedded on the sphere that represents the Gaussian map of each polytope. Recall that

the number of faces (**F**) of the Gaussian map is always equal to the number of vertices of the polytope. However, the number of halfedges (**HE**) and vertices (**V**) of the Gaussian map is either equal to twice the number of edges and the number of facets in the primal representation, respectively, or greater than the respective figures due to intersections between Gaussian-map edges and the identification arc. The table also lists the time in seconds (**t**) it takes to construct the arrangement once the intermediate polyhedron is in place, on a Pentium PC clocked at 1.7 GHz. In addition, the table provides information regarding an alternative representation of polytopes called Cubical Gaussian Map (CGM) [30] we have implemented in the past. The CGM employes six planar arrangements embedded on the six facets of the unit cube, respectively, stitched at the cube edges and vertices. Notice, that the number of features in the CGM representation is higher than in the corresponding Gaussian maps due to the excessive splitting caused by the stitching.

2.2.2. Minkowski-sum Construction. The overlay of two spherical subdivisions $\mathcal{S}_1$ and $\mathcal{S}_2$ is a subdivision $\mathcal{S}$, such that there is a face f in $\mathcal{S}$ if and only if there are faces f_1 and f_2 in $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively, such that f is a maximal connected subset of $f_1 \cap f_2$; this is a straightforward generalization of the planar case [17, Section 2.3].

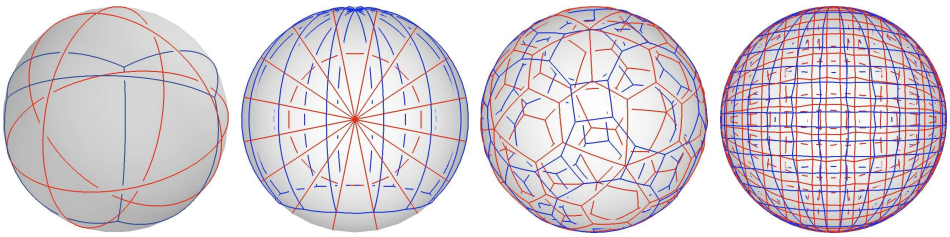


The overlay of the Gaussian maps of two polytopes P and Q identifies all the pairs of features of P and Q respectively that have parallel supporting planes, as they

Table 1: Complexities of the primal and dual representations. SGM — Spherical Gaussian Map, CGM — Cubical Gaussian Map, Tetra. — Tetrahedron, Octa. — Octahedron, Icosa. — Icosahedron, DP — Dioctagonal Pyramid, PH — Pentagonal Hexecontahedron, TI — Truncated Icosidodecahedron, GS4 — Geodesic Sphere level 4, El16 — Ellipsoid-like polytope made of 16 latitudes and 32 longitudes, t - time consumption in seconds.

Object	**Polytope**			**SGM**				**CGM**			
type	**V**	**E**	**F**	**V**	**HE**	**F**	**t**	**V**	**HE**	**F**	**t**
Tetra.	4	6	4	4	12	4	0.01	42	102	21	0.01
Octa.	6	12	8	10	28	6	0.01	24	48	12	0.01
Icosa.	12	30	20	21	62	12	0.01	72	192	36	0.01
DP	17	32	17	25	80	17	0.01	97	280	55	0.01
PH	60	150	92	101	318	60	0.03	200	600	112	0.02
TI	120	180	62	77	390	120	0.05	230	840	202	0.03
GS4	252	750	500	506	1512	252	0.08	708	2124	366	0.07
El16	482	992	512	528	2016	482	0.11	776	2752	612	0.06

occupy the same space on the unit sphere, thus, identifying all the pairwise features that contribute to the boundary of the Minkowski sum of P and Q. A facet of the Minkowski sum is either a facet f of Q translated by a vertex of P supported by a plane parallel to f, or vice versa, or a facet parallel to two parallel planes supporting an edge of P and an edge of Q, respectively. A vertex of the Minkowski sum is the sum of two vertices of P and Q, respectively, supported by parallel planes. The figure above shows the Minkowski sum of a tetrahedron and a cube (the left part of the figure) and the Gaussian map of the Minkowski sum.



(a) Tetra. ⊕ Cube (b) DP ⊕ ODP (c) PH ⊕ TI (d) El16 ⊕ OEl16

Figure 2: Gaussian maps of Minkowski sums. Refer to Tables 1 and 2 for the abbreviations.

When the overlay operation progresses, new vertices, edges, and faces of the resulting arrangement are created based on features of the two operands. When a new feature is created its attributes are updated. For example, a new face f is induced by the overlap of two faces f_1 and f_2 of the Gaussian maps of the two summands, respectively. In that case, the primal vertex associated with f is set to be the sum of the primal vertices associated with f_1 and f_2, respectively.

Table 2: Complexities of primal and dual Minkowski-sum representations. ODP — Dioctagonal Pyramid orthogonal to DP, RGS4 — Rotated Geodesic Sphere level 4, OEl16 — Ellipsoid-like polytope made of 16 latitudes and 32 longitudes orthogonal to El16. Refer to Table 1 for the remaining abbreviations.

Smd 1	**Smd 2**	**Minkowski Sum**								
		Primal			**SGM**			**CGM**		
		V	**E**	**F**	**V**	**HE**	**F**	**V**	**HE**	**F**
Icosa.	Icosa.	12	30	20	21	62	12	72	192	36
DP	ODP	131	261	132	141	540	131	242	832	186
PH	TI	248	586	340	429	1712	429	514	1670	333
GS4	RGS4	1048	2582	1536	1564	5220	1048	1906	6288	1250
El16	OEl16	2260	4580	2322	2354	9224	2260	2826	10648	2510

Smd 1	Smd 2	SGM	CGM	NGM	Fuk	CH
Icosa.	Icosa.	0.01	0.01	0.12	0.01	0.01
DP	ODP	0.04	0.02	0.33	0.35	0.05
PH	TI	0.13	0.03	0.84	1.55	0.20
GS4	RGS4	0.71	0.12	6.81	5.80	1.89
El16	OEl16	1.01	0.14	7.06	13.04	6.91

Table 2 lists the number of features (**V**, **HE**, **F**) in the arrangement that represents the Gaussian map of the respective Minkowski sums. The table to the right shows the time in seconds (**t**) it takes to construct the arrangement once the Gaussian maps of the summands are in place (**SGM**). It also shows the time it takes to compute exact Minkowski sums using a second method based on the CGM representation (**CGM**), a third method implemented by Hachenberger based on Nef polyhedra embedded on the sphere [37] (**NGM**), a fourth method implemented by Weibel[6]based on an output-sensitive algorithm designed by Fukuda [35] (**Fuk**), and a non output-sensitive method that computes the convex hull of the pairwise sum of the vertices of the two summands (**CH**).[7] Note that Fukuda's algorithm is more general, as it can be used to compute the Minkowski sum of polytopes in an arbitrary dimension d, and as far as we know, it has not been optimized specifically for $d = 3$. While our (spherical) Gaussian map method exhibits better performance than the **NGM**, **Fok**, and **CH** methods, it is still inferior to the **CGM** method. As the implementation of the framework, the traits classes, and the application is rather new and haven't been fully optimized yet, we believe that once all optimizations are in place the gap between the time consumption of these two methods will decrease or even revert. Figure 2 depicts Gaussian maps of some of the resulting Minkowski sums listed in the tables above.

2.3. Voronoi Diagrams on the Sphere

Voronoi diagrams were thoroughly investigated and were used to solve many geometric problems [3, 55]. The Voronoi diagram of a set of objects, referred to as Voronoi sites, is the decomposition of the embedding space into maximal relatively-open connected cells, where each cell consists of points that are closer to one particular site (or a set of sites) than to any other site. Voronoi diagrams are strongly connected to arrangements [20] — a property that yields a very general approach for computing Voronoi diagrams.

This concept of space decomposition was extended to various kinds of geometric Voronoi sites, ambient spaces, and distance functions, such as power diagrams of circles in the plane, multiplicatively weighted Voronoi diagrams, additively weighted Voronoi diagrams, and more (e. g., [3, 13, 25, 26, 55]). An immediate extension is the creation of various Voronoi diagrams, embedded on two-dimensional orientable parametric surfaces in general [50, 56], and on the sphere in particular [53, 54, 58].

[6] `http://roso.epfl.ch/cw/poly/public.php`

[7] The convex hull is computed using the `CGAL::convex_hull_3` function, which implements the *quickhull* algorithm [43].

In this section we describe another application of arrangements of arcs of great circles embedded on a sphere. The new ability to construct this class of arrangements provides the means to efficiently construct *envelopes* of functions defined over the sphere, thus enabling the construction of Voronoi diagrams, the bisectors of which are composed of geodesic arcs. We show how we use a general framework for constructing Voronoi diagrams to construct two types of Voronoi diagrams on the sphere.

The technique to compute Voronoi diagrams on two-dimensional orientable parametric surfaces in an exact manner described in this section can be applied to other surfaces as well, conditioned on the ability to handle bisectors of sites embedded on these surfaces.

2.3.1. Envelopes on Surfaces. We define lower envelopes of functions on the sphere in a way similar to the standard definition of lower envelopes of bivariate functions in the three-space [38]:

Definition 2.1 (Lower Envelope over the Sphere). Given a set of bivariate functions (possibly partially defined) $\mathcal{F} = \{f_1, \ldots, f_n\}$, where $f_i : \mathbb{S}^2 \to \mathbb{R}$, their *lower envelope* $\Psi(u, v)$ is defined to be their pointwise minimum:

$$\Psi(u, v) = \min_{1 \le i \le n} f_i(u, v).$$

The *minimization diagram* $\mathcal{M}_{\mathcal{F}}$ of the set $\mathcal{F}$ is the subdivision of $\mathbb{S}^2$ into maximal relatively-open connected cells, such that the function (or the set of functions) that attains the lower envelope over all points of a specific cell of the subdivision is the same. Alternatively, the minimization diagram can be seen as the two-dimensional central projection of the lower envelope onto $\mathbb{S}^2$. The *upper envelope* and the *maximization diagram* are defined similarly.

Agarwal *et al.* [1] presented a simple and efficient divide-and-conquer algorithm for the construction of lower and upper envelopes of bivariate functions defined over $\mathbb{R}^2$. Assuming that all functions are "well-behaved," they showed that the theoretical worst-case time complexity of the algorithm is[8] $O(n^{2+\varepsilon})$, for any $\varepsilon > 0$. Randomization admits an expected running time of $O(F(n) \log n)$, for envelopes of complexity $F(n) = \Omega(n^{1+\varepsilon})$, for some $\varepsilon > 0$, and of $O(n \log^2 n)$ for envelopes of complexity $O(n)$ [57].

The algorithm partitions $\mathcal{F}$ into two disjoint subsets $\mathcal{F}_1$ and $\mathcal{F}_2$ of roughly equal size, and recursively constructs their respective minimization diagrams $\mathcal{M}_{\mathcal{F}_1}$ and $\mathcal{M}_{\mathcal{F}_2}$. Each feature – vertex, edge, or face – of $\mathcal{M}_{\mathcal{F}_1}$ and $\mathcal{M}_{\mathcal{F}_2}$ is labeled with the set of functions that attain the lower envelope over it. The merging of $\mathcal{M}_{\mathcal{F}_1}$ and $\mathcal{M}_{\mathcal{F}_2}$ into the final minimization diagram $\mathcal{M}_{\mathcal{F}}$ starts with overlaying the two minimization diagrams to obtain a new refined arrangement whose features are labeled with the functions attaining the lower envelopes over both diagrams.

[8] A bound of the form $O(f(n) \cdot n^{\varepsilon})$ means that the actual upper bound is $C_{\varepsilon} f(n) \cdot n^{\varepsilon}$, for any $\varepsilon > 0$, where C_{ε} is a constant that depends on ε, and generally tends to infinity as ε goes to 0.

Next, the minimization diagram over each feature is constructed, splitting some of the cells (edges or faces). The splitting of features is a non-trivial task and requires a careful handling. Finally, redundant features are removed, and faces labeled with identical sets of functions are stitched together, to yield the combined final minimization diagram.

Meyerovitch presented an implementation of this algorithm in the form of a CGAL package, named `Envelope_3` [51, 52]. The implementation computes the lower (or the upper) envelope of a set of general surfaces in three dimensions, and deals with all inputs, including all degenerate situations. The efficient implementation is the by-product of minimizing the number of exact (and slow) arithmetic operations through a clever propagation of pre-computed geometric and topological information. The resulting minimization diagram is represented with an arrangement data-structure. The implementation mainly makes use of two operations supported by the `Arrangement_2` package: (i) sweep-based overlay operation, which is used to overlay two minimization diagrams, and (ii) zone computation-based insertion operation, which is used to insert bisector curves that partition cells of the refined arrangement.

The new `Arrangement_on_surface_2` package extends the aforementioned operations, that is, the sweep-line and zone-computation, to support two-dimensional parametric surfaces. Thus, we utilize the `Envelope_3` code to handle minimization diagrams that are embedded on two-dimensional parametric surfaces with little effort.

While computing lower envelopes of functions defined over two-dimensional orientable parametric surfaces has its own significance, we concentrate, in Section 2.3.2 below, on describing how this ability is exploited to compute Voronoi diagrams on the sphere.

2.3.2. Exact Construction of Voronoi Diagrams on the Sphere. Let $O = \{o_1, \ldots, o_n\}$ be a set of n objects in $\mathbb{S}^2$ (also referred to as *Voronoi sites*), and let $\rho : O \times \mathbb{S}^2 \to \mathbb{R}$ be a distance function between Voronoi sites and points on the sphere.

Definition 2.2 (Voronoi Diagram on the Sphere). The *Voronoi diagram* of O over $\mathbb{S}^2$ with respect to ρ is defined to be the partition of $\mathbb{S}^2$ into maximal relatively-open connected cells, where each cell consists of points that are closer to one particular site (or a set of sites) than to any other site. Formally, every point $p \in \mathbb{S}^2$ lies in a cell corresponding to a set of sites $P \subseteq O$ if, and only if, $\rho(p, p_i) < \rho(p, p_j)$ for every $p_i \in P, p_j \notin P$, and $\rho(p, p_i) = \rho(p, p_j)$ for every $p_i, p_j \in P$.

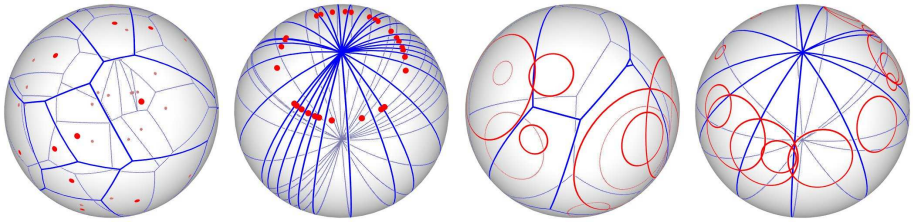
The *bisector* of two sites is the locus of points that have an equal distance to both sites.

Definition 2.3 (Geodesic Distance). Given two points $p, q \in \mathbb{S}^2$, the *geodesic distance* between them $\rho(p, q)$ is defined to be the shortest distance measured along a path on the surface of the sphere. The geodesic distance $\rho(p, q)$ is equal to the length of a geodesic arc that connects p and q.

We define the Voronoi diagram for a set of points on the sphere (or *spherical Voronoi diagram*) to be the Voronoi diagram of the set as induced by the geodesic distance function. The bisector of two point sites on the sphere is a great circle, which is the intersection of the sphere and the bisector plane of the points in $\mathbb{R}^3$, as imposed by the Euclidean metric [54, 55].

Another type of Voronoi diagrams whose bisectors are great circles is the power diagram of circles on the sphere [58], which generalizes the Voronoi diagram of points; see Figures 3c and 3d. Power diagrams on the sphere have several applications similar to the applications of power diagrams in the plane. For example, determining whether a point is included in the union of circles on the sphere, and finding the boundary of the union of circles on the sphere [45, 58].

Given two circles on the sphere c_1 and c_2, let π_1 and π_2 be the planes containing c_1 and c_2, respectively. The bisector of c_1 and c_2 is the intersection of the sphere and the plane that contains the intersection line of π_1 and π_2 and the origin. If π_1 and π_2 are parallel planes, then the bisector is the intersection of the sphere and the plane that contains the origin and is parallel to both π_1 and π_2.



(a) (b) (c) (d)

Figure 3: Voronoi diagrams on the sphere. Sites are drawn in red and Voronoi edges are drawn in blue. (a) The Voronoi diagram of 32 random points. (b) A highly degenerate case of Voronoi diagram of 30 point sites on the sphere. (c) The power diagram of 10 random circles. (d) A degenerate power diagram of 14 sites on the sphere.

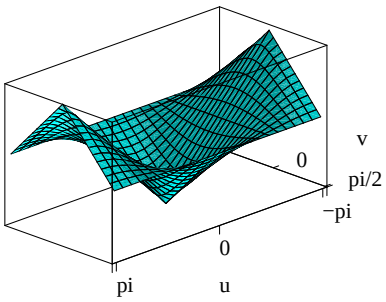
Edelsbrunner and Seidel observed the connection between Voronoi diagrams in $\mathbb{R}^d$ and lower envelopes of the corresponding distance functions to the sites in $\mathbb{R}^{d+1}$ [20]. This observation also holds for the spherical case. For example, given a set $P = \{p_1, \ldots, p_n\}$ of points on the sphere if we take $f_i(x) = \rho(x, p_i)$, for $1 \leq i \leq n$, then the minimization diagram of $\{f_1, \ldots, f_n\}$ over $\mathbb{S}^2$ corresponds to the Voronoi diagram of P over $\mathbb{S}^2$.

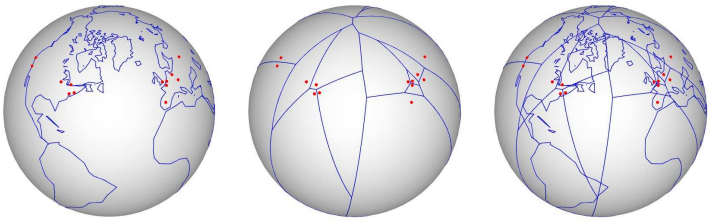
A new framework was developed to compute different types of Voronoi diagrams based on the envelope algorithm of CGAL [57]. The implementation is exact and handles all kinds of degenerate input. The framework provides a reduced and convenient interface between the construction of Voronoi diagrams and the construction of envelopes.

Obtaining a new type of Voronoi diagrams requires the provision of a traits class. This traits class models the *EnvelopeVoronoiTraits_2* concept that refines one

of the traits concepts for the geometry-traits classes, which handles bisector curves of the new diagram type. Essentially, every type of Voronoi diagram, the bisectors of which can be handled by an arrangement traits class, can be computed using this framework, provided that the user supplies a small additional set of procedures, e. g., a procedure for comparing distances to two sites from a given point; see [57].

The newly created traits class (described in Section 2.1) enables the computation of Voronoi diagrams on the sphere, the bisectors of which are great circles (or *piecewise curves* composed of geodesic arcs, though currently there are no such instances implemented). As mentioned above, the bisectors of Voronoi diagrams of points and power diagrams on the sphere are great circles; see Figure 3. We implicitly construct envelopes of distance functions defined over the sphere to compute Voronoi diagrams. The image to the right illustrates the distance function from $(0,0) \in [-\pi,\pi] \times [-\frac{\pi}{2},\frac{\pi}{2}]$ in the parameter space. Projecting the intersection of two such functions onto the unit sphere using a central projection results in a great circle. Notice, however, that we use only rational arithmetic, and so, only Voronoi diagrams of *rational* points on the sphere and power diagrams of circles contained in *rational* planes are supported.





Voronoi diagrams are represented as arrangements, and can be passed as input to consecutive operations supported by the `Arrangement_on_surface_2` package and its derivatives. The left-most figure shows an arrangement on the sphere induced by (i) the continents and some of the islands on earth, and (ii) 20 major cities in the world, which appear as isolated vertices. The arrangement consists of 1065 vertices, 1081 edges, and 117 faces. The data was taken from gnuplot[9] and google maps.[10] The middle sphere embeds an arrangement that represents the Voronoi diagram of the 20 cities above. The right figure shows the *overlay* of the two aforementioned arrangements (computed with the generic overlay function from the `Arrangement_on_surface_2` package).

[9] `http://www.gnuplot.info`
[10] `http://maps.google.com`

3. Concretizations with Algebraic Arithmetic

Beyond geodesic arcs on a sphere, we also demonstrate concretizations for more complicated surfaces. For that purpose, we look at the following scenario: given some *reference surface* S, and a set $\{S_1, \ldots, S_n\}$ of other surfaces not overlaping with S, the intersection curves $S \cap S_i$ induce an arrangement on S. Our goal is to compute such arrangements for certain choices of an algebraic reference surface S, and for arbitrary algebraic surfaces S_i as in the following definition.

Definition 3.1 (Algebraic surface). Let $g \in \mathbb{Q}[x, y, z]$. The *real algebraic surface* induced by g is the point set $V_{\mathbb{R}}(g) := \{(x, y, z) \in \mathbb{R}^3 \mid g(x, y, z) = 0\}$.

We assume that the input surfaces $S_1, \ldots, S_n$ are given by their defining equations $g_1, \ldots, g_n$. We report on two different approaches for two respective types of reference surfaces.

In our first example we consider a ring Dupin cyclide, which generalizes a torus, as the reference surface. We directly exploit a rational parameterization ϕ_S of the cyclide, that is, the surface is attained by the image of the unbounded plane $\mathbb{R}^2$ under ϕ_S. This allows to represent the intersection curves of the cyclide with algebraic surfaces in the two-dimensional parameter space, namely as arrangement of unbounded algebraic curves. We show how to enhance a proper geometry-traits class, having four open sides, for this use case. The underlying planar model utilizes the interplay of the two (yet) prototypical CGAL packages `Algebraic_kernel_d` and `Curved_kernel_via_analysis_2`, which fundamentally rely on the provided analyses of singles and pairs of algebraic plane curves.[11] The analyses' efficiency is guaranteed by a clever combination of approximative though certified methods (like real root isolation with the bitstream Descartes method [23]) with unavoidable symbolic computations; see [21], [22], and [48] for in-depth presentation of the task. Beyond the geometry traits, we also describe details that our model of the *ArrTopologyTraits_2* concept for a cyclide respects in order to support the specialties of this reference surface of genus one.

We then turn to elliptic quadrics, that means ellipsoids, elliptic paraboloids, and elliptic cylinders (the sphere is thus also included as a special case). While rational parameterizations for those reference surface also exist, we abstain from exploiting such. Instead, we follow a different approach, which is nicely supported by the `Arrangement_on_surface_2` package as well: We construct a parameterization that is more suitable for a projection approach. Its parameter space can be decomposed into two parts: the image of one is exactly representing the lower subsurface of the quadric, the image of the other constitute the upper subsurface. Then, we project all intersection curves embedded in a subsurface into the xy-plane. Again, we enhance the geometry traits for unbounded algebraic plane curves by level numbers to support the special geometry of this parameterization. The topology-traits classes required for ellipic quadrics as reference surfaces are simpler than the one for cyclides, but all share common ideas.

[11] We also remark the possibility to replace these layers by filtered variants [47].

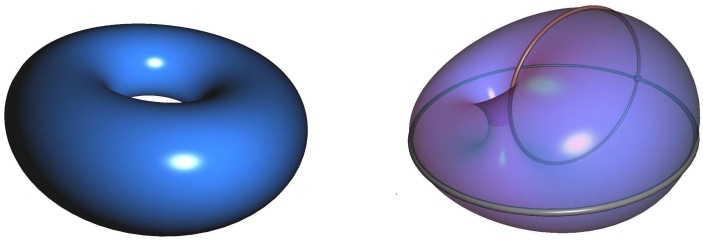
For both kinds of reference surfaces we show how to utilize the exact and efficient planar kernel, highlight details of the implementations, and report experimental results. We refrain from an extensive discussion of conceivable applications in order to concentrate the more elaborate traits classes for these surfaces. We previously presented our work in [7, §4 and §5], and [11]; a detailed discussion is given in [6, §4].

3.1. On a (ring) Dupin cyclide

Our first reference surface considered is a parameterized ring Dupin cyclide S. Dupin cyclides have been introduced by Dupin as surfaces whose lines of curvature are all circular [19]; a quite intuitive construction of a (Dupin) cyclide is due to Maxwell (cited from Boehm [12]):

> Let a sufficiently long string be fastened at one end to one focus of an ellipse, let the string be kept always tight while sliding smoothly over the ellipse, then the other end sweeps out the whole surface of a cyclide S.

Observe that a torus is yielded if the ellipse is actually a regular circle. For simplicity of presentation, we assume that a cyclide is in *standard position and orientation*, that is, the chosen base ellipse is defined by $(x/a)^2 + (y/b)^2 = 1$, $a \geq b > 0$.



(a) $a = 2$, $b = 2$, $\mu = 1$

(b) $a = 13$, $b = 12$, $\mu = 11$. Pole, outer circle, and tube circle are drawn.

Figure 4: Two examples of ring Dupin cyclides. All cyclide pictures are produced with `xsurface` that is based on CGAL's planar curve renderer [24].

The cyclide is defined uniquely by a, b, and a parameter μ that is the length of the string minus a. We define $c = \sqrt{a^2 - b^2}$, which represents the distance between the focus and the center of the ellipse. We concentrate on *ring cyclides* where $c < \mu < a$.[12] Such a surface looks like a squashed torus and is free of pinch points; see Figure 4 for two examples. We refer the reader to [16] for a complete classification of Dupin cyclides.

[12] Non-ring cyclides might contain self-intersections, which are not (yet) handled by CGAL' `Arrangement_on_surface_2` framework.

Crucial for our approach is the fact that ring Dupin cyclides possess a rational parameterization. We start with the following (trigonometric) parameterization [34]:

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} \mapsto \begin{pmatrix} \frac{\mu(c-a\cos\alpha\cos\beta)+b^2\cos\alpha}{a-c\cos\alpha\cos\beta} \\ \frac{b(a-\mu\cos\beta)\sin\alpha}{a-c\cos\alpha\cos\beta} \\ \frac{b(c\cos\alpha-\mu)\sin\beta}{a-c\cos\alpha\cos\beta} \end{pmatrix}$$

with $\alpha, \beta \in [-\pi, \pi]$.

If $\alpha = \pi$ or $(\alpha = -\pi)$ is fixed, the parameterization above yields the *tube circle* $(x+a)^2 + z^2 = (\mu+c)^2$. If $\beta = \pi$ (or $\beta = -\pi$) is fixed, it yields the *outer circle* $(x+c)^2 + y^2 = (a+\mu)^2$. The intersection $p := (-\mu - c - a, 0, 0)$ of tube and outer circle is called the *pole* of the cyclide.

To arrive at a rational parameterization, we use the following identities:

$$\cos\theta = \frac{1-\tan^2\frac{\theta}{2}}{1+\tan^2\frac{\theta}{2}} \quad \sin\theta = \frac{2\tan\frac{\theta}{2}}{1+\tan^2\frac{\theta}{2}}$$

By setting $u := \tan\frac{\alpha}{2}$ and $v := \tan\frac{\beta}{2}$, we get rid of the trigonometric functions. We write the resulting parameterization in homogenous coordinates, that is, the common (non-zero) denominator is treated as a separate fourth variable.

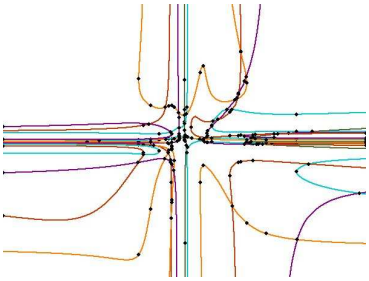
$$\begin{array}{rcl} \mathring{P} : \mathbb{R}^2 & \to & \mathbb{R}^4, \\ \begin{pmatrix} u \\ v \end{pmatrix} & \mapsto & \begin{pmatrix} \mu(c(1+u^2)(1+v^2) - a(1-v^2)(1-u^2)) + b^2(1-u^2)(1+v^2) \\ 2u(a(1+v^2) - \mu(1-v^2))b \\ 2v(c(1-u^2) - \mu(1+u^2))b \\ a(1+u^2)(1+v^2) - c(1-u^2)(1-v^2) \end{pmatrix} \end{array}$$

Observe, that the image of $\mathring{P}$ is the cyclide without the tube circle and the outer circle. As a geometric intuition, we can think of cutting the cyclide along the outer circle and tube circle and "roll out" the surface to cover the plane. Thus, we also refer to the outer circle and the tube circle of a cyclide as its *cut circles*.

Rational parameterizations of the cut circles are obtained by setting $\alpha = \pi$ or $\beta = \pi$, respectively, and applying the same identities as above to get rid of trigonometric functions. By interpreting the tube circle parameterization as the closure of $\mathring{P}$ for $x = \pm\infty$, and the parameterization of the outer circle for $y = \pm\infty$, one obtains a (continuous) parameterization of the whole cyclide as a map $P : (\overline{\mathbb{R}})^2 \to \mathbb{R}^3$. Clearly, while P is bijective in its interior, points on the cut circles have two pre-images (and even four for the pole), because of the identification of opposite sides.

Our approach. We aim for a direct representation of the intersection curves in the parameter space of the cyclide. The idea behind this is fairly simple: consider a surface S_i of arbitrary degree with implicit (homogeneous) equation $g_i = 0$, then the intersection of S_i and S is given by the equation

$g_i(\mathring{P}(u,v)) = 0$. Doing this for all input surfaces, induced by $g_1, \ldots, g_m$, yields an arrangement of m algebraic curves in the parameter space of S; this can be computed utilizing the planar geometry traits provided by the packages `Algebraic_kernel_d` and `Curved_kernel_via_analysis_2`, which we have mentioned at the beginning of this section. An example of such an arrangement is depicted to the right. It is induced by 5 intersecting surfaces of degree 3 on a torus and consists of 208 vertices, 314 edges, and 107 edges.



For our purposes, we "only" have to interpret the result as an arrangement on the cyclide during its construction, that means, we have to use a non-trivial topology-traits class that realizes identifications of opposite sides, and to enhance the planar geometry with respect to the identifications. For brevity, we do not go into much more details, but some points of our implementation should be discussed. We start with the geometry traits:

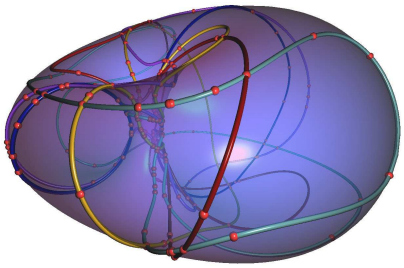
- If the degree of each g_i is bounded by n, the intersection curves have bidegree up to $(2n, 2n)$, and total degree up to $4n$. Although this does not pose a principal problem, the computation becomes practically infeasible for too high degrees. Our approach is practicable for surfaces of moderate degree.
- We have to deal with identifications on both pais of opposite sides of the boundary, that is, we have to provide comparisons of curve-ends near the boundaries, to check whether a point or curve lies on an identification, and to compare points on identified sides; see [8] for more details. Observe that the `Curved_kernel_via_analysis_2` is a model that deals with four open (unbounded in parameter space) sides and that the comparisons near the boundaries still perfectly fit.
- There exists certain components that live exclusively on the cut circles, and are thus are not observeable when only considering the interior of the parameter space: isolated points at infinity (happens when a surface touches the cyclide at a cut circle), or lines at infinity (happens when a surface completely contains a cut circle). Both cases, however, can be detected quite easily by plugging in the rational parameterizations of both cut circles into g_i. The degree of the resulting (univariate) polynomial determines the presence of a line on the boundary, and its real roots give all intersections of S_i and S and the cut circles. This allows to determine whether a point or a (sub)curve on an identification curve exists.

On the topology-traits class we want to make the following remarks. We concentrate on the realization and the outcome of topological tasks. Further details can be found in [8, §3 and §5].

- The initial face is bounded and covers the whole cyclide.

- For each identification, that is, for each cut circle, we maintain a sorted list of EDCEL-vertices. Their order is determined by the comparisons of points along the cut circles, as provided by the geometry-traits class for cyclides. To locate the correct position of such a special curve-end in the circular list of incident curves around a vertex on a cut-circle, we make use of the `Arrangement_on_surface_2`'s internal functor `Is_between_cw_2` that returns `true` if a curve is between two curves meeting at the same point while rotating counterclockwise.
- The design of our topology-traits class ensures that the root of the face-component graph is either a single face (torus-like) or that there are non-contractible components at the top level. We remark one specialty on our surface of genus one. Our topology-traits class minds the case that the first non-contractible closed curve does not result in a face split, but only convert the torus-like initial face into an cylinder-like face. For more details on this issue we refer the reader to [8].
- We remark, that curves intersecting or touching the pole of the cyclide require special handling, for instance, applying symbolic pertubations. That is, for consistency reasons we symbolically let the intersection take place on exactly one identification.

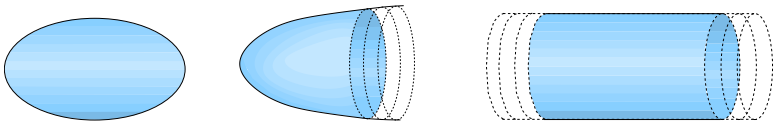
The arrangement on a cyclide shown on the right side is computed using our traits classes. It is induced by 5 algebraic surfaces of degree 3 intersecting the reference surface. It consists of 240 vertices, 314 edges, and 74 faces. Experimental results are presented in Section 3.3 below.



3.2. Arrangements of Intersection Curves on a Quadric

We come to our second type of reference surfaces, an x-elliptic quadric, which is an algebraic surface defined by $q \in \mathbb{Q}[x, y, z]$ having total degree 2. Its intersections with any plane $x = x_0$ constitutes an ellipse (or a single point, or the empty set). The set of x-elliptic quadrics comprises all ellipsoids, elliptic cylinders that are unbounded in the x-direction, and paraboloids that are either unbounded towards $x = -\infty$ or $x = +\infty$. Figure 5 illustrates examples of such surfaces. These quadrics have nice properties: first, they are composed of a single connected component and second, they allow a "level"-parameterization, which is explained below.

An xy-functional surface is given as the graph of a bivariate function $z = f(x, y)$, which is not true for an x-elliptic quadric S. But S can be subdivided into two xy-functional surfaces by cutting along the *silhouette curve* $silhouette(S) := V_{\mathbb{R}}(q) \cap V_{\mathbb{R}}(\frac{\partial q}{\partial z})$. It induces the *lower* and *upper* part of S. For example, the equator of a sphere splits it into the southern and into the northern hemisphere. Both hemispheres are xy-functional. The *projected silhouette* of S onto the xy-plane is



(a) ellipsoid (b) elliptic paraboloid (c) elliptic cylinder

Figure 5: Elliptic quadrics

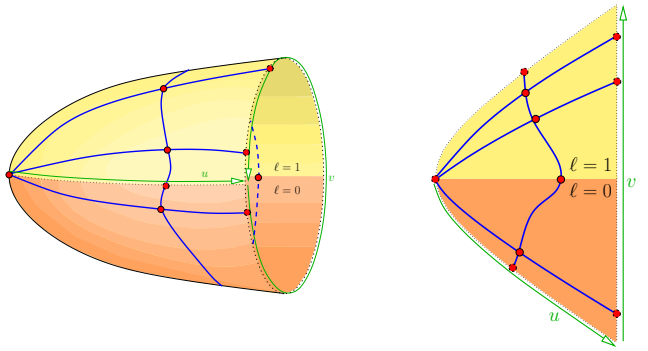
algebraically defined by the resultant polynomial[13] $\mathrm{res}_z(q, \frac{\partial q}{\partial z})$. It has degree 2 for quadrics.

Consider next the spatial intersection curve of S with another surface $V_{\mathbb{R}}(g_i)$, that is, $V_{\mathbb{R}}(q) \cap V_{\mathbb{R}}(g_i)$. The projection of this set onto the xy-plane is contained in a real algebraic plane curve of total degree $2 \cdot \deg_{\text{total}}(g_i)$, defined by $\mathrm{res}_z(q, g_i)$. The projected curve can be split at its critical points and intersection points with the projected silhouette of S, resulting in (weakly) x-monotone curves and isolated points. Each such object can be assigned to the lower or upper part of S (and in some cases to both parts). That is, we compute the decomposition of the space curve respecting the silhouette of S; see [10] for "lifting" the projected intersection curve induced by an arbitrary quadric, for the intersection with an algebraic surfaces of arbitrary degree we refer the reader to [6, §5.5.3]. This assignment enables to compute two individual *planar* arrangements that correspond to the subdivisions on the two parts of S. The merging of both parts has not been done so far.

Our approach. Our technique to stitch the two parts is to parameterize the x-elliptic reference quadric S over a rectangular domain $\Phi = U \times [0, 2\pi]$, with $U \subseteq \overline{\mathbb{R}}$ with the continiuous function $\phi_S(u, v) = (u, y(u, v), r(u, y(u, v), -\sin v))$. We define $y(u, v) = y_{u,\min} + (\sin \frac{v}{2})(y_{u,\max} - y_{u,\min})$. The interval $[y_{u,\min}, y_{u,\max}]$ denotes the y-range of the ellipse that S induces on the plane $x = u$. The function $r(x, y, s)$ returns the minimal ($s \leq 0$) or maximal ($s > 0$) element of $\mathcal{R}_{S,x,y} := \{z \,|\, q(x, y, z) = 0\}$, $|\mathcal{R}_{S,x,y}| \leq 2$.

However, we avoid to exploit this non-rational parameterization. Looking closer at Φ and ϕ_S, we observe that the sin-function horizontically decomposes Φ into two rectangular areas, namely $\Phi_0 := [l, r] \times [0, \pi]$ and $\Phi_1 := [l, r] \times (\pi, 2\pi)$. It holds that $\phi(\Phi_0)$ forms the (closed, i. e., with silhouette) lower part of S and $\phi(\Phi_1)$ models the (open, i. e., without silhouette) upper part of S. As $C_I := \phi_S(u, 0) = \phi_S(u, 2\pi)$, we observe an identification curve for this parameterization. Obviously, C_I is a connected subset of S's silhouette. In addition, depending on the type of S, that is, if $u_{\min}$ is finite or not, we observe a contracted (ellipsoid, bounded end of paraboloid) or an open (unbounded end of paraboloid, cylinder) left side of Φ; similar for $u_{\max}$ and the right side. In Figure 6 we illustrate this decomposition (i. e., parameterization) on an example of a paraboloid that is intersected by surfaces.

[13] The resultant is the determinant of the Silvester matrix of two polynomials [5, Chapter 4].



(a) On the paraboloid

(b) Simulation in the plane by *inversion* of upper part

Figure 6: Illustration of a paraboloid's parameterization: the dark-shaded (orange) area represents Φ_0, the bright-shaded (yellow) area corresponds to Φ_1.

The decomposition into two subdomains enables to derive a special geometry traits for curves embedded on the reference quadric from a planar geometry traits as basic ingredient: Given a point $w_0 = (u_0, v_0)$, with $p_0 := \phi_S(u_0, v_0) = (x_0, y_0, z_0)$ being its counterpart on S, the *level* of p_0 is $\ell \in \{0, 1\}$ if $w_0 \in \Phi_\ell$. We represent a point $p_i = (x_i, y_i, z_i)$ on S as the combination of a planar point $\overline{p}_i(x_i, y_i)$ and its level $\ell_i \in \{0, 1\}$. Given two points p_1, p_2, the uv-lexicographic order of their counterparts w_1, w_2 in the parameter space is first reflected by the order of $x_1 = u_1$ and $x_2 = u_2$. In the case that $u_1 = u_2$ we infer the v-order from (y_1, ℓ_1) and (y_2, ℓ_2). We distinguish between 3 cases: (a) if $0 = \ell_1 < \ell_2 = 1$ then $w_1 <_v w_2$, (b) if $\ell_1 = \ell_2 = 0$ then w_1 and w_2's v-order is identical to the y-order of $\overline{p}_1$ and $\overline{p}_2$, and finally, (c) if $\ell_1 = \ell_2 = 1$ then w_1 and w_2's v-order is the opposite of $\overline{p}_1$ and $\overline{p}_2$'s y-order.

Similarly, we represent an arc cv on S with a u-monotone pre-image by a projected arc $\overline{cv}$ that is enhanced by three levels: $\ell_{\min}$ at the minimal end of $\overline{cv}$, $\ell_{\max}$ at the maximal end of $\overline{cv}$, and ℓ representing the level in the interior of $\overline{cv}$, which must be constant for all interior points. As Φ_0 is closed, an arc with $\ell = 1$ (lying on the upper part of S) can have ends that lie on S's silhouette, that is, having $\ell_{\min} = 0$, or $\ell_{\max} = 0$, or both. This holds in particular if an end meets the identification curve.

We provide a geometry traits respecting this parameterization based on the planar `Curved_kernel_via_analysis_2` which is instantiated with CGAL's `Algebraic_curve_kernel_2` that supports arbitrary-degree algebraic curves in the projection plane. Our modifications and extensions are simple recombinations of the existing planar counterparts.

- We assign levels to planar objects and replace all predicates that involve planar y-comparisons with versions that respect the partitioning of Φ, that is,

implementing the new v-comparison. Those predicates first compare levels, and in case both input objects lie on the upper part of S, return the inversed result of the original predicate. Geometric constructions, such as the intersection operation, are also replaced by implementations that rely on planar constructions, but whose output is now "levelled" onto S.

- Following [8] we provide comparisons of curve-ends (and points) near the boundaries of Φ, that is, close to contracted or open boundaries (in u-direction) or close to the identification (in v-direction). We only discuss the comparison near a left contraction point; the other predicates are implemented similarly. Consider two curves on S approaching the left contraction point. If one of their pre-images belongs to Φ_0 and the other to Φ_1, the order is determined by the assigned levels. Otherwise, consider the corresponding planar situation: both projected curves emanate from the projected contraction point. Consequently, their order in parameter space slightly to the right of the contracted side is given by the y-order of the planar curves slightly to the right of their minimal intersection (which coincides with the projected contraction point). In the case that both pre-images belong to Φ_1, we invert the result of the planar y-comparison to the right of the projected contraction point.
- Using similar combinations of planar operations, we can easily decide whether a given curve or point lies on the identification or belongs to the left or right side of the boundary.
- We are also required to compare points on the identification curve. Knowing that $x(\phi_S(u, v)) = u$, we can simply reuse the planar comparison of points' x-coordinates.

The additional comparisons of levels are negligible compared to the operations on planar curves of degree up to 4 that determine the overall performance of this geometry traits. This bound is also tight for rational parameterizations of elliptic quadrics, as done with the cyclide. There exist smaller-degree parameterizations, which would lead to smaller-degree curves. But those curves are then defined over a non-trivial extension field, namely with square-roots. Although they can be handled by CGAL's algebraic kernel, its efficiency suffers.

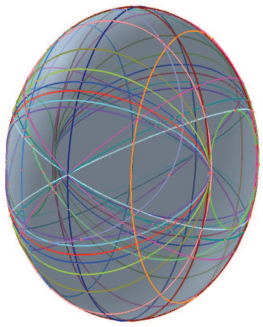
The topology of the reference quadric S requires special handling. We next discuss details of our topology-traits class that combines the various cases (i. e., ellipsoid, paraboloid, and cylinder). Remember that the topology-traits class mainly helps to consistently construct a EDCEL respecting the surface's topology.

- The initial record of the EDCEL is a single face, which is bounded if S is an ellipsoid, and unbounded if S is a paraboloid or a cylinder.
- It maintains the vertices related to the boundary of the parameter space. Vertices V_{left} and V_{right} are designated to represent a contraction point, or an open end, depending on whether S is an ellipsoid (both sides contracted), a cylinder (both sides unbounded), or a paraboloid (one side unbounded, one side contracted). Vertices on S's top-bottom identification curve are maintained in a sorted sequence (`std::map`). The order of stored vertices is defined

by the order of attached points using the corresponding geometry-traits functor. The location of a curve-end in the circular list of curves incident to a vertex related to the boundary is implemented by clever combinations of planar comparisons of curves in the neighborhood of the planar point.

- Following [8], the topology traits also helps to decide whether the insertion of a curve results in face split and the nesting of faces. Depending on the type of S, we decided to follow the different strategies: for an ellipsoid or paraboloid, we ensure exactly one outermost face. In case of an ellipsoid, this face always contains the image of the parameter space's upper right corner, that is, the "west pole". For a paraboloid, the selected outermost is always unbounded, either the image of the upper right corner of Φ (if the paraboloid opens to the right), or the image of the lower left corner of Φ (if the paraboloid opens to the left). No such face is maintained, if S is an unbounded cylinder. In particular, faces that contain a non-contractible closed curve are "outermost faces". We refer the reader to [8] for more details on surfaces with an identification curve.

We instantiated the `Arrangement_on_surface_2` class-template with the two described traits classes, which resulted in a robust algorithm to compute the arrangement on an x-elliptic quadric. Our software successfully constructs even highly degenerate arrangements. The figure to the right shows such an example on an ellipsoid induced by 23 other ellipsoids intersecting it. Further experimental results are given in Section 3.3.



3.3. Results

We measured the performance when computing the arrangement on given base surfaces intersecting a set of given surfaces using our new geometry-traits and topology-traits classes. Since both geometry-traits classes reduce their computations to a planar one, we also compare the obtained results with corresponding planar arrangements: for quadrics, we compute the lower and upper subdivision as in [10]. Note that, due to the "level"-parameterization, the task of decomposing into sweepable curves is similar for planar and embedded arrangements. For cyclides, we compute the arrangement of curves in the cyclide's uv-parameter space using CGAL's topology-traits class for the unbounded plane.

All experiments were executed on an AMD Dual-Core Opteron(tm) 8218 multi-processor Debian Etch platform, each core equipped with 1 MB internal cache and clocked at 1 GHz. The total memory consists of 32 GB. As compiler we used `g++` in version 4.1.2 with flags `-O2 -DNDEBUG`. We rely on exact arithmetic number types provided by CORE and utilize CGAL's prototypical `Algebraic_curve_kernel_2` for analyses of algebraic curves in the plane.

Cyclide. In our tests, we used two different reference cyclides. First, the *standard torus* S_T with $a = 2$, $b = 2$, $\mu = 1$, centered at the origin with no applied rotation.

Table 3: Running times (in seconds) to construct arrangements on S_T or S_C induced by algebraic surfaces.

Reference: **torus** S_T										
		Split	**sweep planar arrangement**				**sweep toroid arrangement**			
Instance	#S	**t (s)**	#V	#E	#F	**t (s)**	#V	#E	#F	**t (s)**
ipl-1	10	**0.05**	146	190	93	**0.09**	119	190	71	**0.09**
ipl-1	20	**0.10**	440	682	337	**0.48**	384	682	298	**0.48**
ipl-1	50	**0.27**	1960	3363	1642	**1.73**	1837	3363	1526	**1.87**
ipl-2	10	**0.22**	392	575	252	**1.03**	358	575	217	**0.85**
ipl-2	20	**0.59**	1301	2147	1014	**2.46**	1211	2147	937	**2.55**
ipl-3	10	**1.02**	593	847	355	**3.60**	542	847	305	**3.82**
ipl-3-6points	10	**1.39**	787	1092	466	**29.78**	680	1092	412	**31.04**
ipl-3-2sing	10	**1.15**	794	1062	429	**4.42**	694	1062	368	**4.67**
ipl-4	10	**6.40**	858	1204	483	**43.57**	785	1204	419	**44.02**
ipl-4-6points	10	**9.83**	1100	1529	598	**440.71**	989	1529	540	**451.91**
ipl-4-2sing	10	**8.50**	1034	1471	602	**44.28**	933	1471	538	**45.51**

Reference: **ring Dupin cyclide** S_C										
		Split	**sweep planar arrangement**				**sweep cyclidean arrangement**			
Instance	#S	**t (s)**	#V	#E	#F	**t (s)**	#V	#E	#F	**t (s)**
ipl-1	10	**0.11**	204	280	147	**0.35**	169	280	111	**0.42**
ipl-1	20	**0.09**	539	808	420	**0.45**	456	808	352	**0.77**
ipl-1	50	**0.22**	3425	6084	3052	**3.11**	3228	6084	2856	**3.56**
ipl-2	10	**0.20**	506	710	313	**1.01**	450	710	260	**1.02**
ipl-2	20	**0.71**	1448	2247	1050	**2.86**	1323	2247	924	**2.71**
ipl-3	10	**1.15**	534	682	269	**4.21**	474	682	208	**4.09**
ipl-4	10	**8.69**	1064	1406	495	**43.74**	988	1406	418	**42.24**

Second, a non-torical cyclide S_C with $a = 13$, $b = 12$ and $\mu = 11$. We remark that our implementation allows to transform a cyclide in standard position and orientation, that is, to translate it by a vector and to rotate it with respect to a rotational matrix with rational entries. Thus, we translated S_C's center to $(1, 1, 1)$ and applied a rotation defined by the matrix

$$\frac{1}{3}\begin{pmatrix} 2 & -2 & 1 \\ 2 & 1 & -2 \\ 1 & 2 & 2 \end{pmatrix}$$

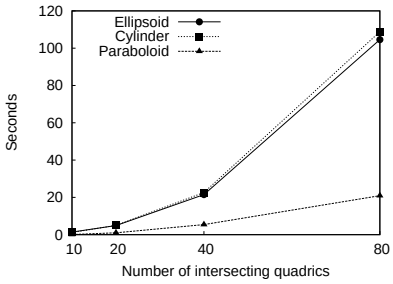
We intersect both with algebraic surfaces of fixed degree that are interpolated by randomly chosen points on a three-dimensional grid, having no or some degeneracies with respect to S_T: the surfaces in "6points" instances share at least 6 common points on S_T, one of them is the pole of S_T. The surfaces in the "2sing" instances induce (at least) two singular intersections on S_T.

Our obtained running times are listed in Table 3. For such random examples, our algorithm shows a good general behavior, even for higher degree surfaces. Degeneracies with respect to the reference surface result in higher running times as the instance "6points" shows.

Unlike the planar approach, our direct method yields the aspired EDCEL representation of the induced arrangement on the surface. This can be observed by the reduced number of vertices and faces in the output, due to identifications. As

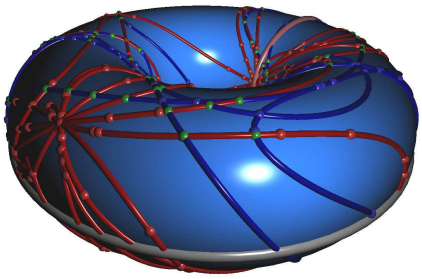
previously presented, our traits classes provide the few boundary-specific tasks to achieve this goal. The performance penalty is marginal, so the efficiency of the implementation is not harmed by the topology-traits class. This is not surprising, as the computations in a topology-traits hardly trigger any costly geometric operation. Thus, we infer that the chosen approach strongly hinges on the efficiency of the underlying (bivariate) algebraic kernel.

Quadrics. For these experiments we intersect instances from [40] containing 10, 20, 40, and 80 quadrics with three random reference quadrics (ellipsoid, cylinder, and paraboloid). All surfaces have 10-bit coefficients. Table 4 lists the observed running times with structural data, that is, number of EDCEL-records. The figure to the right illustrates the running times with respect to the number of intersecting quadrics and the type of reference quadric. As one expects for a sweep-line approach, growth of running time is super linear in the number of quadrics. Clearly, the more complex the arrangement, the more time is required to compute it. To give a better feeling for the relative time consumption, we indicate the time spent for each pair of half-edges in the EDCEL of the computed arrangement. This time varies in the range between 2.0 ms and 4.5 ms. Other parameters have significant effect on the running time as well, for example the bit-size of the coefficients of the intersection curves.



4. Conclusions

We addressed the problem of practically computing arrangements on non-linear surfaces, such as spheres, ellipsoids, cylinders, and tori. Using the framework of the companion paper [8], this task becomes nearly as simple as the construction planar arrangements. Moreover, our implementations benefit from other software tools available in the `Arrangement_on_surface_2` package of CGAL and its derivatives [59]. As an example, the picture on the right shows an overlay of two arrangements (displayed in red and blue color) on a cyclide. Other immediate applications are Boolean set operations on surfaces and point location queries in an arrangement structure.



Another application has been considered by Hemmer et al. [41]. They consider an arrangement of quadrics in $\mathbb{R}^3$, and describe the local neighborhood of each vertex in a so-called *environment map*. For an exact representation of this environment map, they simply compute the arrangement on a sufficiently small

Table 4: Comparing planar and quadrical topologies. We report performance measures (in seconds) for random quadrics intersecting three reference quadrics and distinguish the computation of two planar arrangements from the computation of one quadrical arrangement.

Reference: **Ellipsoid**									
	Split	**sweep two planar arrangements**				**sweep ellipsoidal arrangement**			
#	**t (s)**	#V	#E	#F	**t (s)**	#V	#E	#F	**t (s)**
10	**2.36**	213+217	295+289	84+84	**1.29**	396	584	190	**1.36**
20	**4.18**	544+540	844+838	302+300	**4.53**	1038	1682	646	**4.90**
40	**7.62**	1831+1837	3192+3210	1363+1375	**20.57**	3568	6402	2836	**21.50**
80	**15.47**	7187+7191	13363+13379	6178+6190	**97.66**	14144	26742	12600	**104.56**

Reference: **Cylinder**									
	Split	**sweep two planar arrangements**				**sweep cylindrical arrangement**			
#	**t (s)**	#V	#E	#F	**t (s)**	#V	#E	#F	**t (s)**
10	**1.65**	191+179	260+240	71+64	**1.17**	344	500	158	**1.23**
20	**3.38**	551+509	852+780	303+273	**4.74**	1012	1632	622	**5.00**
40	**6.76**	1821+1755	3168+3040	1349+1287	**21.28**	3474	6208	2736	**22.57**
80	**14.28**	7086+6914	13179+12831	6095+5919	**100.91**	13768	26010	12244	**108.76**

Reference: **Paraboloid**									
	Split	**sweep two planar arrangements**				**sweep paraboloidal arrangement**			
#	**t (s)**	#V	#E	#F	**t (s)**	#V	#E	#F	**t (s)**
10	**1.02**	28+16	37+13	11+2	**0.14**	36	50	17	**0.14**
20	**1.86**	124+96	181+129	60+35	**0.93**	196	310	116	**0.96**
40	**4.83**	469+337	787+533	321+198	**5.21**	756	1320	566	**5.38**
80	**9.87**	1303+1267	2309+2272	1008+1006	**20.25**	2472	4580	2110	**20.90**

ball containing the vertex, induced by the intersection of the ball with all quadrics in the arrangement. As a result, they compute all edge cycles that bound faces of the arrangement, an important step towards the complete representation of the three-dimensional arrangement.

Computational biology frequently employs spherical arrangements in molecular modeling: each sphere represents an atom of a molecule and the arrangement on the sphere represents the intersection pattern with neighboring atoms. Geodesics are insufficient to support this application, but there is an on-going effort to provide a specialized geometric-traits class for small circles on the sphere based on the results in [15, 18]. Our spherical topology-traits class remains suitable for this case as well. Beyond the spherical modelling of atoms, their anisotropic interactions can be represented using ellipsoids as primitive objects. In all cases `Arrangement_on_surface_2` is instantly usable for algorithmic problems within this context.

Future work. Natural directions for further work consist of the provision of geometry-traits classes for other families of curves embedded on the available surfaces and the augmentation of the set of supported parametric surfaces. Conceivable, for instance, are specialized geometry-traits classes that efficiently deal with geodesic arcs on quadrics or cyclides. For other surfaces with rational parameterization, it is mainly the algebraic degree of the curves in parameter space that limits practical feasibility.

Another famous example for a surface with rational parameterization is the Möbius strip. It has two bounded sides and two opposite *twisted-identified* sides. The latter turns the surface non-orientable, which constitutes another problem: while we would be able to provide all required geometric operations, we are unable to represent an induced arrangement using a DCEL-like representation. Guibas and Stolfi's quad-edge data structure is a sufficient alternative [36]. As a partial achievement, it may be interesting to extract all intersections, with the help of a sweep line invocation, or to maintain a "virtual cut", that is, not to merge faces (and their boundary cycles) incident to the "twisted-identification".

We also see possible applications in rounding. Piecewise linear curves in the plane, also referred to as polylines, are of particular interest, as they can be used to approximate more complex curves. At the same time they are easier to deal with in comparison to higher-degree algebraic curves, as rational arithmetic is sufficient to carry out exact computations on them. A similar argument holds for piecewise geodesic curves. Our software package already supports planar polylines. As our implementation is generic, supporting piecewise (geodesic) curves on surfaces as well, is just a matter of proper instantiation of existing components and enables, for instance, approximations of curves on the sphere with poly-geodesics.

The topology-traits classes for spheres and cyclides might come in handy in other situations, not directly related to 3D-applications on a reference surface. One concrete example is robot motion planning, where the configuration space of a certain one-link or two-link robot can be modelled by a periodic space. More generally, periodic spaces and computations therein are of great importance in various fields — the authors of [14] mention the need of (three-dimensional) periodic spaces in astronomy, biomedical computing, solid-state chemistry, physics of condensed matter and fluid dynamics.

Acknowledgments

We thank Ron Wein for fruitful collaboration on the research and development of the generic framework the work described in this paper is based on. We also thank Ron for many useful advises and rich discussions directly related to the material described in this paper.

We thank Pavel Emeliyanenko for his work on visualizing arrangements embedded into algebraic surfaces in three-dimensional space, which is utilized in Section 3.

References

[1] Pankaj K. Agarwal, Otfried Schwarzkopf, and Micha Sharir. The overlay of lower envelopes and its applications. *Discrete & Computational Geometry*, 15:1–13, 1996.

[2] Pankaj K. Agarwal and Micha Sharir. Arrangements and their applications. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*,

chapter 2, pages 49–119. Elsevier Science Publishers, B.V. North-Holland, Amsterdam, North-Holland, 2000.

[3] Franz Aurenhammer and Rolf Klein. Voronoi diagrams. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, chapter 5, pages 201–290. Elsevier Science Publishers, B.V. North-Holland, Amsterdam, North-Holland, 2000.

[4] Matthew H. Austern. *Generic Programming and the STL*. Addison-Wesley, 1999.

[5] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in Real Algebraic Geometry*, volume 10 of *Algorithms and Computation in Mathematics*. Springer, 2nd edition, 2006.

[6] Eric Berberich. *Robust and Efficient Software for Problems in 2.5-Dimensional Non-Linear Geometry (Algorithms and Implementations)*. Ph.D. thesis, Universität des Saarlandes, Germany, 2008.

[7] Eric Berberich, Efi Fogel, Dan Halperin, Kurt Mehlhorn, and Ron Wein. Sweeping and maintaining two-dimensional arrangements on surfaces: A first step. In *Proceedings of 15th Annual European Symposium on Algorithms (ESA)*, volume 4698 of *LNCS*, pages 645–656. Springer-Verlag, 2007.

[8] Eric Berberich, Efi Fogel, Dan Halperin, Kurt Mehlhorn, and Ron Wein. Arrangements on parametric surfaces I: General framework and infrastructure, 2010. Accepted to Mathematics in Computer Science.

[9] Eric Berberich, Efi Fogel, Dan Halperin, and Ron Wein. Sweeping over curves and maintaining two-dimensional arrangements on surfaces. In *Abstracts of 23rd European Workshop on Computational Geometry*, pages 223–226, 2007.

[10] Eric Berberich, Michael Hemmer, Lutz Kettner, Elmar Schömer, and Nicola Wolpert. An exact, complete and efficient implementation for computing planar maps of quadric intersection curves. In *Proceedings of 21st Annual ACM Symposium on Computational Geometry (SoCG)*, pages 99–106. Association for Computing Machinery (ACM) Press, 2005.

[11] Eric Berberich and Michael Kerber. Exact arrangements on tori and Dupin cyclides. In *Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling (SPM)*, pages 59–66. Association for Computing Machinery (ACM) Press, 2008.

[12] Wolfgang Boehm. On cyclides in geometric modeling. *Computer Aided Geometric Design*, 7:243–255, 1990.

[13] Jean-Daniel Boissonnat, Camille Wormser, and Mariette Yvinec. Curved Voronoi diagrams. In J.-D. Boissonnat and M. Teillaud, editors, *Effective Computational Geometry for Curves and Surfaces*, pages 67–116. Springer-Verlag, 2007.

[14] Manuel Caroli and Monique Teillaud. Compute 3D periodic triangulations. Technical Report 6823, Inria Sophia-Antipolis, 2009.

[15] Frédéric Cazals and Sébastien Loriot. Computing the arrangement of circles on a sphere, with applications in structural biology. *Computational Geometry: Theory and Applications*, 42(6-7):551–565, 2009.

[16] Vijaya Chandru, Debasish Dutta, and Christoph M. Hoffmann. On the geometry of Dupin cyclides. *The Visual Computer*, 5(5):277–290, 1989.

[17] Mark de Berg, Mark van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, Germany, 2nd edition, 2000.

[18] Pedro M.M. de Castro, Frédéric Cazals, Sébastien Loriot, and Monique Teillaud. Design of the CGAL 3D Spherical Kernel and application to arrangements of circles on a sphere. *Computational Geometry: Theory and Applications*, 42(6-7):536–550, 2009.

[19] Charles Dupin. *Applications de Géométrie et de Méchanique*. Bachelier, Paris, 1822.

[20] Herbert Edelsbrunner and Raimund Seidel. Voronoi diagrams and arrangements. *Discrete & Computational Geometry*, 1:25–44, 1986.

[21] Arno Eigenwillig and Michael Kerber. Exact and efficient 2D-arrangements of arbitrary algebraic curves. In *Proceedings of 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 122–131, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics (SIAM).

[22] Arno Eigenwillig, Michael Kerber, and Nicola Wolpert. Fast and exact geometric analysis of real algebraic plane curves. In *Proceedings of the 2007 International Symposium on Symbolic and Algebraic Computation*, pages 151–158, New York, NY, USA, 2007. Association for Computing Machinery (ACM) Press.

[23] Arno Eigenwillig, Lutz Kettner, Werner Krandick, Kurt Mehlhorn, Susanne Schmitt, and Nicola Wolpert. A Descartes algorithm for polynomials with bit-stream coefficients. In *8th International Workshop on Computer Algebra in Scientific Computing*, volume 3718 of *LNCS*, pages 138–149, 2005.

[24] Pavel Emeliyanenko. Visualization of points and segments of real algebraic plane curves. M.Sc. thesis, Universität des Saarlandes, Germany, 2007.

[25] Ioannis Zacharias Emiris and Menelaos Ioannis Karavelas. The predicates of the Apollonius diagram: Algorithmic analysis and implementation. *Computational Geometry: Theory and Applications*, 33(1-2):18–57, 2006.

[26] Ioannis Zacharias Emiris, Elias P. Tsigaridas, and George Tzoumas. Voronoi diagram of ellipses in CGAL. In *Abstracts of 24th European Workshop on Computational Geometry*, pages 87–90, 2008.

[27] Andreas Fabri, Geert-Jan Giezeman, Lutz Kettner, Stefan Schirra, and Sven Schönherr. On the design of CGAL a computational geometry algorithms library. *Software — Practice and Experience*, 30(11):1167–1202, 2000.

[28] Efi Fogel. *Minkowski Sum Construction and other Applications of Arrangements of Geodesic Arcs on the Sphere*. Ph.D. thesis, The Blavatnik School of Computer Science, Tel-Aviv University, 2009.

[29] Efi Fogel and Dan Halperin. Exact and efficient construction of Minkowski sums of convex polyhedra with applications. In *Proceedings of 8th Workshop on Algorithm Engineering and Experiments*, 2006.

[30] Efi Fogel and Dan Halperin. Exact and efficient construction of Minkowski sums of convex polyhedra with applications. *Computer-Aided Design*, 39(11):929–940, 2007.

[31] Efi Fogel, Dan Halperin, Lutz Kettner, Monique Teillaud, Ron Wein, and Nicola Wolpert. Arrangements. In J.-D. Boissonnat and M. Teillaud, editors, *Effective Computational Geometry for Curves and Surfaces*, chapter 1, pages 1–66. Springer-Verlag, 2007.

[32] Efi Fogel, Ophir Setter, and Dan Halperin. Exact implementation of arrangements of geodesic arcs on the sphere with applications. In *Abstracts of 24th European Workshop on Computational Geometry*, pages 83–86, 2008.

[33] Efi Fogel, Ophir Setter, and Dan Halperin. Movie: Arrangements of geodesic arcs on the sphere. In *Proceedings of 24th Annual ACM Symposium on Computational Geometry (SoCG)*, pages 218–219. Association for Computing Machinery (ACM) Press, 2008.

[34] Andrew Russel Forsyth. *Lectures on the Differential Geometry of Curves and Surfaces*. Cambridge University Press, 1912.

[35] Komei Fukuda. From the zonotope construction to the Minkowski addition of convex polytopes. *Journal of Symbolic Computation*, 38(4):1261–1272, 2004.

[36] Leonidas J. Guibas and Jorge Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics*, 4(2):74–123, 1985.

[37] Peter Hachenberger, Lutz Kettner, and Kurt Mehlhorn. Boolean operations on 3D selective Nef complexes: Data structure, algorithms, optimized implementation and experiments. *Computational Geometry: Theory and Applications*, 38(1-2):64–99, 2007. Special issue on CGAL.

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

[39] Dan Halperin and Christian R. Shelton. A perturbation scheme for spherical arrangements with application to molecular modeling. *Computational Geometry: Theory and Applications*, 10:273–287, 1998.

[40] Michael Hemmer. *Exact Computation of the Adjacency Graph of an Arrangement of Quadrics*. Ph.D. thesis, Johannes-Gutenberg-Universität, Mainz, Germany, 2008.

[41] Michael Hemmer, Sebastian Limbach, and Elmar Schömer. Continued work on the computation of an exact arrangement of quadrics. In *Collections of Abstracts of 25th European Workshop on Computational Geometry*, pages 313–316, 2009.

[42] Susan Hert, Michael Hoffmann, Lutz Kettner, Sylvain Pion, and Michael Seel. An adaptable and extensible geometry kernel. In *Proceedings of 5th International Workshop on Algorithm Engineering (WAE)*, volume 2141 of *LNCS*, pages 79–90. Springer-Verlag, 2001.

[43] Susan Hert and Stefan Schirra. 3D convex hulls. In CGAL Editorial Board, editor, CGAL *User and Reference Manual*. 3.4 edition, 2008.

[44] Craig D. Hodgson, Igor Rivin, and Warren D. Smith. A characterization of convex hyperbolic polyhedra and of convex polyhedra inscribed in the sphere. *Bull. (New Series) of the AMS*, 27:246–251, 1992.

[45] Hiroshi Imai, Masao Iri, and Kazuo Murota. Voronoi diagram in the Laguerre geometry and its applications. *SIAM Journal on Computing*, 14(1):93–105, 1985.

[46] Vijay Karamcheti, Chen Li, Igor Pechtchanski, and Chee K. Yap. A core library for robust numeric and geometric computation. In *Proceedings of 15th Annual ACM Symposium on Computational Geometry (SoCG)*, pages 351–359. Association for Computing Machinery (ACM) Press, 1999.

[47] Michael Kerber. On filter methods in CGAL's 2D curved kernel. Technical Report ACS-TR-243404-03, Algorithms for Complex Shapes, 2008.

[48] Michael Kerber. *Geometric Algorithms for Algebraic Curves and Surfaces*. Ph.D. thesis, Universität des Saarlandes, Germany, 2009.

[49] Lutz Kettner. Using generic programming for designing a data structure for polyhedral surfaces. *Computational Geometry: Theory and Applications*, 13(1):65–90, 1999.

[50] Richard Kunze, Franz-Erich Wolter, and Thomas Rausch. Geodesic Voronoi diagrams on parametric surfaces. In *Computer Graphics International Conference*, page 230, Washington, DC, USA, 1997. IEEE Computer Society Press.

[51] Michal Meyerovitch. Robust, generic and efficient construction of envelopes of surfaces in three-dimensional space. In *Proceedings of 14th Annual European Symposium on Algorithms (ESA)*, volume 4168 of *LNCS*, pages 792–803. Springer-Verlag, 2006.

[52] Michal Meyerovitch, Ron Wein, and Baruch Zukerman. 3D envelopes. In CGAL Editorial Board, editor, CGAL *User and Reference Manual*. 3.4 edition, 2008.

[53] R. E. Miles. Random points, sets and tessellations on the surface of a sphere. *The Indian Journal of Statistics*, 33:145–174, 1971.

[54] Hyeon-Suk Na, Chung-Nim Lee, and Otfried Cheong. Voronoi diagrams on the sphere. *Computational Geometry: Theory and Applications*, 23(2):183–194, 2002.

[55] Atsuyuki Okabe, Barry Boots, Kokichi Sugihara, and Sung Nok Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons, New York, NY, 2nd edition, 2000.

[56] Guilherme Albuquerque Pinto and Pedro Jussieu de Rezende. Additively weighted Voronoi diagram on the oriented projective plane. In *The Canadian Conference on Computational Geometry*, 2000.

[57] Ophir Setter, Micha Sharir, and Dan Halperin. Constructing two-dimensional Voronoi diagrams via divide-and-conquer of envelopes in space. In *Proceedings of 6th Annual International Symposium on Voronoi Diagrams in Science and Engineering (ISVD)*, pages 43–52, 2009.

[58] Kokichi Sugihara. Laguerre Voronoi diagram on the sphere. *Journal for Geometry and Graphics*, 6(1):69–81, 2002.

[59] Ron Wein, Efi Fogel, Baruch Zukerman, and Dan Halperin. Advanced programming techniques applied to CGAL's arrangement package. *Computational Geometry: Theory and Applications*, 38(1–2):37–63, 2007. Special issue on CGAL.

[60] Ron Wein, Efi Fogel, Baruch Zukerman, and Dan Halperin. 2D arrangements. In CGAL Editorial Board, editor, CGAL *User and Reference Manual*. 3.4 edition, 2008.

[61] Chee K. Yap. Robust geometric computation. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 41, pages 927–952. Chapman & Hall/CRC, 2nd edition, 2004.

Eric Berberich
School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel
e-mail: ericb@post.tau.ac.il

Efi Fogel
School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel
e-mail: efif@post.tau.ac.il

Dan Halperin
School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel
e-mail: danha@post.tau.ac.il

Michael Kerber
Max-Planck-Institut für Informatik, D1: Algorithms and Complexity D-66123 Saarbrücken, Germany
e-mail: mkerber@mpi-inf.mpg.de

Ophir Setter
School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel
e-mail: ophir.setter@cs.tau.ac.il