Overview of project proposal

We present here a proposal for an INRIA research team on effective computational geometry with applications to computer graphics and solid modeling. The main scientific focus of our proposal is the design and implementation of technology-independent, robust and efficient geometric algorithms for 3D visibility and low-degree algebraic surfaces.

By technology-independent we mean usable solutions to geometric problems that will be relevant long after current hardware is obsolete. By robust we mean algorithms that do not crash on degenerate inputs and always output topologically consistent data. By efficient we mean algorithms that run reasonably quickly on realistic data where performance is ascertained both experimentally and theoretically.

Meeting our computational objectives requires mathematical tools that are both geometric and algebraic. In particular, we need further knowledge of the basic geometry of lines and surfaces in a variety of spaces and dimensions as well as to adapt sophisticated algebraic methods, often computationally prohibitive in the most general setting, for use in solving seemingly simple geometric problems.

The applications we currently target are computer graphics and solid modeling. In computer graphics, our goal is to improve the performance of visibility computations for realistic rendering of complex scenes. In solid modeling, our goal is to introduce efficient solutions for robust computation with curved surfaces.

The team

Permanent members
- Hazel Everett [professor, Univ. Nancy 2]
- Xavier Goaoc [researcher, INRIA] (starting May 1st, 2005)
- Sylvain Lazard [researcher, INRIA] (scientific leader)
- Sylvain Petitjean [researcher, CNRS] (associate scientific leader)

PhD students
- Marc Glisse [AMN grant holder, ENS, started in October 2003]
- Geoffroy Lauvaux [CIFRE grant holder, CIRTES, expected to defend in Spring 2005]
- Maria Pentcheva [MENESR–ACI grant holder, started in October 2004]
- Linqiao Zhang [joint with McGill University, started in January 2004]

Post-doctoral fellow
- Laurent Dupont [ATER, since October 2004]

Administrative staff
- Isabelle Herlich [TR INRIA]
## Contents

1 **Context and historical perspective** 1
   1.1 From computational geometry to effective geometric computing . . . . . . . . . . . . . 1
   1.2 From ISA to VEGAS . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2

2 **Scientific objectives and methodology** 3
   2.1 Robustness . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
   2.2 Efficiency . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
   2.3 Versatility . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

3 **Research directions** 9
   3.1 Effective 3D global visibility, theory and applications . . . . . . . . . . . . . . . . . . 9
      3.1.1 Panorama . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
      3.1.2 Theory of lines in space . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
      3.1.3 Global 3D visibility data structures: algorithmics and implementation . . . . . . 12
      3.1.4 Visibility and applications . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
   3.2 Exact geometric computing for low-degree surfaces . . . . . . . . . . . . . . . . . . . 14
      3.2.1 Intersection of surfaces . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
      3.2.2 Arrangements of surfaces . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16

4 **Application domains** 17
   4.1 Computer graphics . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
   4.2 Solid modeling . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
   4.3 Other potential applications . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18

5 **Software aspects** 18
   5.1 General policy . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
   5.2 Software developed by team members . . . . . . . . . . . . . . . . . . . . . . . . . . 19
   5.3 Other software used . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19

6 **Scientific collaborations** 20
   6.1 Collaborations within INRIA . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
   6.2 Other collaborations in France . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
   6.3 International collaborations . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20

7 **Industrial partners** 21

8 **Positioning** 22
   8.1 Our originality . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
   8.2 Positioning in the scientific community . . . . . . . . . . . . . . . . . . . . . . . . . 23
   8.3 Positioning with respect to other INRIA projects . . . . . . . . . . . . . . . . . . . . . 23

9 **Scientific animation** 25

10 **Publications** 26

**Appendix: summaries of representative publications** 29
   A The Expected Number of 3D Visibility Events is Linear . . . . . . . . . . . . . . . . . . 29
   B Geometric Permutations of Disjoint Unit Spheres . . . . . . . . . . . . . . . . . . . . . 30
   C Common Tangents to Spheres in \( \mathbb{R}^3 \) . . . . . . . . . . . . . . . . . . . . . . . . 31
   D Near-Optimal Parameterization of the Intersection of Quadrics . . . . . . . . . . . . . 32
   E Intersecting Quadrics: An Efficient and Exact Implementation . . . . . . . . . . . . . . 33
1 Context and historical perspective

The physical world in which we live is essentially geometric. Computations dealing with models of real-life objects must deal with geometric data. Computational applications that manipulate geometric models are thus ubiquitous in science and technology. Geometric computing is a central building block in many fields, such as computer-aided design, manufacturing, computer graphics, robotics, molecular biology, geographic information systems, astrophysics, computer vision, metrology and many others. The rapid advances in computer hardware, network facilities and visualization systems make geometric computation even more dominant.

For over two decades, computational geometry has been dedicated to supplying a solid foundation for the study of geometric algorithms which are relevant to all these applications.

1.1 From computational geometry to effective geometric computing

Computational geometry (CG) is the field of computer science that studies geometric problems from an algorithmic or complexity-theoretic point of view. It started in the 1980s with the stated ambition to be both a receptacle of important problems emanating from practical situations and a source of algorithmic tools and mathematically sound analyzes for the practitioners.

Arguably, CG has been one of the most active and productive areas of algorithmic design. It has met with considerable success in analytically and computationally solving problems involving basic geometric structures, mostly in the world of linear objects. To name just one success story, CG has provided a solid theoretical and practical understanding of Delaunay triangulations, which are at the core of essentially all three-dimensional piecewise-linear mesh generators. Meshes are invariably required to numerically solve the partial differential equations arising in many areas of computational physics and engineering like fluid flow, heat transfer, electromagnetic modeling and stress analysis.

But while CG succeeded in laying rigorous mathematical foundations and growing to a mature field of research, it has largely distanced itself from applications and failed on one count: building a two-way pipeline connecting theory to practice. In the middle of the last decade, recognition of that fact provoked a community-wide debate, which culminated in 1999 with the publication of the report of the Computational Geometry Impact Task Force \cite{C+99}. The sobering reality is, the report notes, that “[CG] has failed to address the problems that people in practice want to see solved”. Now, the experts of the Task Force argue, there are two options: “CG can use its successes as justification for keeping the pursuit of theoretical investigations as the centerpiece of its agenda. Or it can move towards building an effective pipeline with geometric computing.” The second option is seen by many as the most profitable in the long run. And while theoretical work is of tantamount importance, “CG has much to gain from close interaction with the real-world practice of geometric computing”. So, the report concludes, “CG’ers must start thinking not only in terms of asymptotic complexity but also in terms of code robustness, precision, CPU times, standards, benchmarks, distribution, etc.”

The community as a whole has been quite receptive to the proposed changes, and notably to experimental work and software building. INRIA researchers in particular have been at the forefront of efforts to move research in this direction. The PRISME research team at INRIA Sophia-Antipolis (now GEOMETRICA) has been a major actor of an European consortium of eight sites created to promote the translation of CG results into robust programs for industrial applications. This led to the development of a common library of algorithms, CGAL (Computational Geometry Algorithms Library), which is now a de facto standard.

1.2 From ISA to VEGAS

Geometry and computer graphics have traditionally been closely tied. Geometric research topics in graphics include hidden surface removal algorithms, finite element methods, object simplification, levels of detail, ray tracing, morphing, and partitioning of complex surfaces into simpler pieces.

Since its onset in 1995, the ISA research team at INRIA Lorraine has been a customer of geometry in its many different forms. Over the years, geometry has played a significant part in several achievements of the team. One example where the collaboration within ISA between the geometers and the researchers on image synthesis has been especially fruitful is the Virtual Mesh method for efficiently computing radiosity in scenes consisting of polygons and curved patches [1] (see Figure 1).

Along with these successes came the understanding that, to get the proposed algorithms outside the perimeter of research, it would take more than a simple use of known geometric tools and that the same problems could be and often had to be attacked from different perspectives.

Nothing illustrates this better than the work on the Virtual Mesh. The main feature of this method is its ability to handle curved surfaces in a global illumination loop directly, and not by piecewise-linear approximations as had been done before. Our goal was to illustrate this novelty on models made of patches of quadratic surfaces (better known as quadrics). The chess set of Figure 1 is such a model. But while radiosity algorithms can only work with surfacic models, the only known entirely quadratic models were volumetric and obtained by Boolean set operations on basic primitives. A preliminary step (or at least what looked so at the time) was thus to extract the “skin” of a given volumetric model, a process known as boundary evaluation. Solving numerical and robustness issues turned out to be a nightmare. While we succeeded in computing the boundary representation of a few models with a good deal of manual intervention, the automatic robust conversion of even simple objects was completely out of reach, and by far. On that seemingly simple problem, a lot of theoretical work had to be done.

Thus, at the same time that the CG community realized it had to close the gap with applications, a move in the direction from applications toward geometry took place within ISA. The work on the Virtual Mesh, and other instances, convinced us that there was room for two somewhat orthogonal but complementary agendas: one fast-paced, well adapted to the quickly changing field of computer graphics, with rapidly-integrable research goals, and making good use of optimization methods and GPUs (Graphical Processing Units) for numerical geometry; the other slow-paced, in phase with the computational geometry and geometric computing communities, with long-term research goals focused on revisiting the mathematical and algorithmic foundations of basic problems in photorealistic rendering and solid modeling.

1“Models, algorithms and geometry for computer graphics and vision”, http://www.loria.fr/isa.
The first agenda is embodied in the ALICE research team proposal, headed by Bruno Lévy; the second agenda is the subject of this proposal. It is the outcome of a continuing process of maturation over the past few years and grew out of two young-researcher grants, the ATIP Jeune équipe of CNRS (2002-2003) and the ACI Jeunes chercheurs of the French ministry of research (2003-2006), entitled “Effective geometry for the realistic rendering of complex scenes”.

2 Scientific objectives and methodology

Computational geometry has traditionally treated linear objects like line segments and polygons in the plane, and point sets and polytopes in three-dimensional space, occasionally (and more recently) venturing into the world of non-linear curves like circles and ellipses. The methodological experience and the accumulated know-how have been enormous.

For many applications, particularly in the fields of computer graphics and solid modeling, it is necessary to manipulate more general objects such as complex curves and surfaces given in either implicit or parametric form. Typically such objects are handled by approximating them by simple objects such as triangles. This approach is extremely important and it has been used in almost all of the usable software existing in industry today. It does, however, have some disadvantages. Using a tessellated form in place of its exact geometry may introduce spurious numerical errors (the famous gap between the wing and the body of the aircraft), not to mention that thousands if not hundreds of thousands of triangles could be needed to adequately represent the object. Moreover, the curved objects that we consider are not necessarily everyday three-dimensional objects, but also abstract mathematical objects that are not linear, that may live in high-dimensional space, and whose geometry we do not control. For example, the set of lines in 3D (at the core of visibility issues) that are tangent to three polyhedra span a ruled quadratic surface and the lines tangent to a sphere correspond, in projective five-dimensional space, to the intersection of two quadratic hypersurfaces.

The main scientific objective of the VEGAS research team is to contribute to the development of an effective geometric computing dedicated to non-trivial geometric objects. Included among its main tasks are the study and development of new algorithms for the manipulation of geometric objects, the experimentation of algorithms, the production of reliable, quality software, and the application of such algorithms and implementations to research domains that deal with a large amount of geometric data, notably solid modeling and computer graphics.

Effectiveness is a key word of our research proposal. By requiring our algorithms to be effective, we imply that the algorithms should be robust, efficient, and versatile. By robust we mean algorithms that do not crash on degenerate inputs and always output topologically consistent data. By efficient we mean algorithms that run reasonably quickly on realistic data where performance is ascertained both experimentally and theoretically. Finally, by versatile we mean algorithms that work for classes of objects that are general enough to cover realistic situations and that account for the exact geometry of the objects, in particular when they are curved.

We give in the following subsections a brief description and panorama of the robustness, efficiency, and versatility objectives.

2.1 Robustness

In geometric applications, degeneracies, which occur when the objects considered lie in special positions are, in practice, often the norm, not the exception. This is especially true in computer graphics and shape modeling. The example of Figure 1 can serve to illustrate this point. While the generic intersection between two quadrics is a skew curve (i.e., it does not lie in a single plane) of degree four, it is fairly clear that most if not all of the arcs outlining the boundary representation of the knight piece are line segments and conic arcs, and thus are planar. In other words, these curves are degenerate instances of surface intersections.
Completely and correctly accounting for degeneracies is a major challenge of most geometric software. The overall utility of an implementation may critically depend upon the correct treatment of special cases. Allowing for degenerate data vastly increases the number of special cases and thus the number of exceptional branchings and dedicated procedures in the software. One solution to this problem is to symbolically perturb the input, thereby symbolically moving the geometric problem away from a singular case and resolving degeneracies automatically (see the survey paper by Sugihara [Sug00]).

While handy, the symbolic perturbation approach is often inappropriate, for several reasons. First, its applicability is limited and it has been worked out in detail for only a small class of problems. A perturbation scheme has to move the input into generic position for every degenerate problem instance. Finding such a scheme can be extremely difficult and in any case requires recognizing that the input is degenerate to begin with, which is a substantial part of the work needed to fully treat degeneracies. Second, a symbolic perturbation may unnecessarily slow down computation. Indeed, solving a generic problem instance can involve manipulating arithmetic expressions of longer bit length, and thus be computationally more costly, than when the instance is singular. Our work on the intersection of quadrics has shown precisely that [23]. Since geometric inputs are often degenerate by design (aligned primitives, touching objects, etc), this can have a dramatic impact on the overall performance. Finally, when the input is degenerate, a geometric software applying symbolic perturbation does not necessarily solve the given problem instance. Indeed the output is obtained by a limit-taking process applied to the solution of the perturbed problem; this limit may be structurally different than the solution to the given instance. For example, when intersecting two tangent objects, if the perturbation scheme moves them apart, the limit of the intersection of the perturbed instance will be the empty set instead of, say, a point. For applications where exactness is an issue, this might well be unacceptable.

By contrast, our work falls within the paradigm of exact geometric computing. Recall that a geometric object is really two things: a combinatorial structure (which, for instance, encodes the incidences between the constituting elements of the object) and a set of numerical quantities describing the embedding of the object in space, which one can usually consider to be algebraic. Since there are consistency constraints governing the relation between combinatorial information and numerical quantities, the numerical instability of geometric algorithms is intimately linked to this double nature of geometric objects. Exact geometric computing means performing computations in which numerical quantities are evaluated to sufficient precision in order for the underlying combinatorial structure to be mathematically exact.

The dependence of combinatorial decisions on numerical computation is encapsulated in the notion of predicates. Evaluating a geometric predicate usually consists in determining the sign of some polynomial expression in the input numerical quantities; a simple example is “does a given point lie to the left, to the right or on a given line?”. The paradigm of exact geometric computing requires predicates to be evaluated correctly, which ensures that the branchings made by the algorithm are correct and thus that the topological structure of the output is exact. This paradigm does not necessarily require, however, that the output numerical quantities are exact.

In full generality, solving a geometric problem in this paradigm amounts to

- characterize (mathematically and geometrically) the degeneracies, which can be intrinsic or algorithmic;
- translate each geometric decision (and branching of the algorithm) into the sign-evaluation of some algebraic expression;
- exactly and efficiently evaluate the sign of these algebraic expressions.

Characterizing degenerate situations is crucial because undiscovered degeneracies usually result in fatal runtime error or combinatorially invalid output. Degeneracies are essentially of two different species: intrinsic degeneracies, which are inherent to the problem at hand and have to be handled by any algorithm that intends to robustly solve the problem, and algorithmic degeneracies, which are induced by algorithmic

choices. A careful study of intrinsic degeneracies can lead to very efficient dedicated algorithms, as shown by our work on intersection of quadrics \([21, 23, 26]\). Perhaps less intuitively, a fine analysis of non-intrinsic, algorithmic degeneracies can suggest structural modifications to the algorithm and induce dramatic improvements to its efficiency, as exemplified by the work of Angelier on 2D visibility \([\text{Ang02}]\).

Once a geometric predicate has been identified, it cannot in general be resolved using (fixed-precision) floating-point arithmetic: if a problem instance is nearly degenerate, then the value of the corresponding expression can be very small, possibly less than the rounding error in the floating-point evaluation of the expression. Hence the sign-evaluation may be incorrect, likely resulting, later, in a fatal error. Exact and efficient evaluation of predicates is usually performed with the use of exact (arbitrary-precision) arithmetic and filters. When the value of a polynomial expression is sufficiently far away from zero, filters can compute the exact sign of the expression without computing its value exactly (filters typically use interval arithmetic); otherwise the value of the expression is computed exactly.

The degrees of the polynomials expressing geometric predicates is a direct measure of required arithmetic bit length and computational efficiency: the higher the degree of a predicate is, the more often the filters will fail and the more costly exact evaluation will be. Translating geometric decisions into low-degree predicates is critical to strictly limit the arithmetic and computational demands of a robust implementation. However, deciding whether a predicate realizing a given decision has minimal degree or can be broken down into predicates of smaller degree seems to be, in general, a very difficult algebraic problem\(^2\).

While simple enough for basic geometric primitives, the characterization of degenerate instances of a problem and its translation into low-degree predicates can be mathematically involved even when the primitives are simple. Consider for instance an environment made of spheres of arbitrary radii. From the point of view of 3D visibility, characterizing degeneracies requires detecting when quadruples of spheres admit infinitely many common tangents. While one may be content with such a low-level description, the “infinitely many common tangents” condition cannot be easily turned into (a set of) geometric predicates and detection will be both expensive and prone to numerical instability. Recently, we proved (using tools from complex projective geometry) that degenerate instances of quadruples of spheres are those with aligned centers and admitting at least one common tangent \([3]\). Now, this condition has a much more geometric flavor and can be efficiently verified by low-degree geometric predicates.

Our approach to geometric problems includes a systematic characterization of degeneracies and the design of low-degree predicates for making the corresponding geometric decisions. We however do not work on the design of arithmetic filters which is almost an independent field of research; we only use existing implementations such as the one developed in the kernel of the CGAL library.

## 2.2 Efficiency

In computational geometry an algorithm’s efficiency has been historically measured in terms of worst-case asymptotic complexity in the real RAM model of computation. In this model, time (and space) optimal algorithms are often the ultimate goal. Although worst-case optimal algorithms are clearly important, they are not always the most efficient in practice; quicksort is recognized as the fastest sorting algorithm in practice though only if it is implemented using the \(\Theta(n^{2})\) worst-case algorithm rather than the optimal \(\Theta(n \log n)\) worst-case algorithm. Moreover, they can be hopelessly complicated; Chazelle’s linear-time algorithm for triangulating simple polygons \([\text{Cha91}]\) is unanimously considered to be unimplementable \([\text{Ski97}, \text{pp. 355-357}]\). Nevertheless, we will always be interested in the worst-case complexity of an

---

\(^{2}\)Also, the intrinsic complexity of the problem at hand can well imply that the bit length required for exactly evaluating a predicate is necessarily high. This raises the issue of giving a realistic measure of the complexity of predicate-based algorithms, since some operations can no longer be assumed to take constant time as in the traditional real RAM model of computation.

---


algorithm because it provides a starting point for further analysis.

If the size of the output of an algorithm can vary substantially, then the goal is to design algorithms that optimize the worst-case complexity expressed in terms of two parameters, input and output sizes, the so-called output-sensitive algorithms. An example of this is the problem of computing the convex hull of a set of points in 2D. Although the convex hull may contain all the input points it may also contain as few as three. An optimal worst-case algorithm computes the convex hull of \( n \) points in \( \Theta(n \log n) \) time whereas there is a simple \( \Theta(n \log h) \)-time algorithm where \( h \) is the output size [CSY97]. Clearly, the advantage of the output-sensitive approach is even greater when the output can be as large as \( \Theta(n^4) \) which is the case when computing lines tangent to 4 amongst \( n \) objects in three dimensions.

Computational geometry problems, particularly in three and higher dimensions, are plagued by contorted worst-case configurations that simply never arise in practice. Data sets required to induce an exponential-time performance of the simplex method for linear programming is a well-known example. When worst-case asymptotic complexity is not a relevant measure, another option is to do a probabilistic analysis of the algorithm in which one assumes a distribution on the inputs and derives the expected time complexity. This is often very difficult even for some simple cases such as uniform distributions (see for instance [7]). Although such simple distributions are arguably irrelevant since real-world data is not uniformly random, a probabilistic analysis can give some insight beyond that given by a study of the complexity in the worst-case; worst-case examples don’t necessarily represent real-world data either!

A randomized algorithm is one which makes random choices. An analysis of a randomized algorithm yields an expected-time bound that holds for all inputs; no assumption on the distribution of inputs is made. Randomized quicksort is the classic example. Randomized algorithms have known a tremendous success in computational geometry because they are generally incredibly simple to implement and are fast [Sei91]. Though randomized algorithms should perhaps not be used in critical real-time applications (such as landing a space shuttle), randomization is the way to go in graphics and solid-modeling applications.

Efficiency of algorithms can often be improved by first applying some simple algorithm if the data exhibits some special characteristics. One example is in the quadric intersection algorithm; degenerate cases are recognized and handled by specialized faster algorithms [23]. We advocate this approach. We do not, however, advocate the use of algorithms that solve the problem only in some special cases and crashes or outputs nonsense otherwise.

A judicious choice of data structures can have a huge impact on the efficiency of an algorithm. Hierarchical decompositions of space such as octrees is a classic example from graphics. When the problem involves answering many queries, this can be facilitated by first preprocessing the input into some data structure. Obviously, space becomes an important issue.

When the theoretical analysis of an algorithm shows that it is promising or when we fail to obtain an analysis because we simply don’t have the right tools, the next step would be to implement the algorithm and analyze its running-time (and space) complexity on random and real data. This step has posed enormous problems to computational geometers partly because of the robustness issues discussed in the previous section. Many, possibly most, published geometric algorithms work only on non-degenerate input. Since typical graphics scenes contain degeneracies, it is critical to write algorithms that work on all inputs so that they may be tested.

To summarize, our objective is to design and implement algorithms, correct on all inputs, that require a reasonable amount of extra storage and that run quickly on realistic data. We will ascertain the performance of our algorithms both theoretically and through experiments.


2.3 Versatility

When confronted with curved objects, most applications in realistic rendering and solid modeling use polyhedral approximations to these surfaces and apply existing algorithms to manipulate these approximations. This approach leads invariably to very large data which has a negative impact on the algorithm’s efficiency. In addition, the resulting computations are inaccurate and topological consistency cannot be guaranteed.

While efficient algorithms are essential to allow increases in the size and complexity of triangulated meshes, effective geometric computing must also move towards more realism and therefore enter the realm of non-linear objects. Unfortunately, computational geometry provides efficient methods only when the models can be described as large collections of linear objects. State-of-the-art algorithmic knowledge is largely helpless when it comes to robustly and efficiently manipulating even simple curved objects.

Our objective is to contribute to giving solid foundations to effective geometric computing for simple non-linear objects and to validate theoretical advances with robust and efficient implementations. Mastering this challenge implies developing multidisciplinary research, involving computational geometry, computer algebra, mathematics of curves and surfaces, and real and complex algebraic geometry. Indeed, the difficulties inherent in extending the computational geometry repertoire to curved objects are of several natures:

- **mathematical and theoretical**: compared to linear objects, the effective handling of even the simplest of curved objects is not a baby step but a giant step and the intrusion of mathematics is often massive (see, for example, our work on intersecting quadrics [21, 26] and on counting geometric permutations of unit spheres [5]);
- **algorithmic**: basic algorithms have to be revisited and sometimes entirely rethought to suit the needs of curved primitives (see, for example, our work on circular ray-shooting in the plane [4] and [MS97] for ray shooting amidst spheres);
- **computational**: robustness and degeneracy issues are even harder to handle in the world of curved objects than in the linear world (see, for example, our work on characterizing “visual” degeneracies among spheres [3] and again on intersecting quadrics [21, 26]). Also, purely algebraic computational tools often lead to extremely slow running time.

Most objects of interest in computer graphics and solid modeling are (semi-)algebraic, i.e., they are defined by polynomial equations (and inequations). Thus algebra plays a fundamental role in non-linear geometric computing. Many operations at the heart of algorithms dealing with curved primitives boil down to evaluating, manipulating and solving systems of polynomial equations.

Computer algebra and symbolic computation provide a possible approach to the realization of these operations. Well-known tools such as resultants, Gröbner bases and Sturm sequences have proved to be reliable means of attacking and solving geometric problems. Reliance on these tools should however enter only at the end of the pipeline, once the problem has been completely straightened out from a geometric standpoint. Each problem can be formulated (algebraically) in many different ways and computer algebraists know from experience that the formulation can have a dramatic impact on the computational cost of the solving process. Also, the algebraic translation of a problem tends to blur the information of geometrical nature. Finding formulations that are both geometrically meaningful and that lend themselves well to symbolic manipulation is always a difficult problem.

That symbolic computation should be driven by geometric information can be illustrated as follows. Consider again the problem of deciding whether four given spheres admit infinitely many common tangents and parameterizing such tangents. This problem can be directly translated into a polynomial system which can be solved, or, using our recent geometric characterization of degenerate instances of the problem [3], one can check whether the four spheres have aligned centers, if so check whether they admit one
tangent (with a low-degree predicate), and only then, if necessary, parameterize all the common tangents using their geometric characterization. That the latter procedure will be computationally much more efficient than the former should be quite clear.

Intuition dictates that the smaller is the degree of the (class of) surfaces considered, the more the algorithms can take advantage of the geometry. To support this intuition, we can call on the following two facts. One is well-known, the other is mathematically more involved:

- Quadrics (algebraic surfaces of degree 2) and cubic surfaces (degree 3) are instances of rational surfaces, i.e., surfaces that can be parameterized by polynomial functions, while, generically, quartic surfaces (degree 4) are not. Thus, quadrics and cubics have a dual parametric/implicit nature which is very important for algorithm design. It is for instance crucial in our work on parameterizing intersections of quadrics [21, 23].

- The Abel-Ruffini Theorem states that the roots of a univariate polynomial \( f \) of degree \( n \geq 5 \) cannot in general be expressed in closed form, that is, in terms of its coefficients using a finite number of additions, multiplications, divisions, and root extractions. Another way to put it is that the Galois group of \( f \) is not solvable. Now, consider a problem of a geometric nature, like “count the number of lines tangent to four spheres”. Associated to the problem is also a Galois group acting on the lines tangent to the spheres. The amount of interplay between the algebra and the geometry of the problem is largely governed by the type of this Galois group [Baj86]: if the group is not solvable, a purely geometric method will not be able to handle the problem. But some seemingly simple problems are already known to have non-solvable Galois groups. For instance, a general plane quartic curve \( C \) has finitely many bitangents; can we give formulas for these bitangents in terms of the defining equation of \( C \)? The answer is no: the Galois group acting on the bitangents is not solvable [Har79].

Accordingly, we shall focus on problems involving low-degree algebraic objects, in the hope that geometry will play a large part. Obtaining a clear and precise view of the problems involved in the extension of classical computational geometry data structures and algorithms to curved objects is another key objective. Results have recently been obtained in that direction with the effective realization of the well-known Bentley-Ottmann sweep-line algorithm for arcs of circles [DFMT02], conics [BEH+02], and cubics [EKSW04]. Extending those results to 3-dimensional objects is currently an area of very active research.

The conceptual complexity of geometric algorithms involving curved objects is high, because of the proliferation of special cases. All cases should be considered and correctly handled to lift reliability concerns. Also, the dependence of combinatorial decisions on numerical computation is even more consequential in the world of curved objects and keeping topology and geometry coherent is harder to achieve. In other words, the failure rate of a floating-point implementation is likely to be higher and robustness is a critical issue. Geometric predicates are usually highly non-linear and since their exact evaluation is very costly the need for filtering techniques and approximate methods with certified answer is considerable. Efficiently coupling symbolic and numerical computing is an important problem.

---


3 Research directions

We now present our two main directions of research: effective 3D global visibility and exact geometric computing for low-degree surfaces.

It should be stressed that these two themes are strongly related, as the following examples illustrate. First, the line transversals to three lines in space span a ruled surface which is quadratic. Hence computing the lines through four lines, a basic operation in 3D visibility, is akin, in some sense, to intersecting two quadratic surfaces. Second, it is useful when studying lines in 3D to represent them as the points of some space. Unfortunately, the parameterization of the lines of projective 3-space cannot be as simple as for points and planes: it is known that there is no “algebraic model” for the space of lines that is itself a projective space [PW01, p. 143]. The best we can hope for is a parameterization of lines by the points of a quadratic hypersurface $\Psi$ in projective 5-space (the so-called Plücker embedding). Now, each tangency condition in 3-dimensional space has an equivalent in 5-dimensional space: the lines meeting a given line correspond to the intersection of a hyperplane of $\mathbb{P}^5(\mathbb{R})$ with $\Psi$ and the lines tangent to a sphere correspond to the intersection of a hypersurface of degree 2 with $\Psi$. This way, visibility problems in $\mathbb{P}^3(\mathbb{R})$ (or $\mathbb{R}^3$) can be formulated as problems on arrangements of low-degree surfaces in $\mathbb{P}^5(\mathbb{R})$. This shows that 3D visibility problems are inherently non-linear, even when dealing initially with polyhedral objects. Knowing how difficult problems on arrangements of low-degree surfaces already are in 3-dimensional space should give an idea as to why robust visibility computations present an enormous research challenge.

3.1 Effective 3D global visibility, theory and applications

We first present a panorama of 3D global visibility and then describe our research objectives.

3.1.1 Panorama

Visibility computations are central in computer graphics applications. Computing the limit of the umbra and penumbra cast by an area light source, identifying the set of blockers between any two polygons and determining the view from a given point are examples of visibility queries that are essential for the realistic rendering of 3D scenes. In global illumination algorithms, where the flow of light in a scene is simulated according to the laws of geometrical optics, visibility computations are excessively costly. In fact more than half of the overall computation time can routinely be spent on visibility queries in radiosity simulations. Our ultimate goal in studying 3D visibility is to speed up these queries and improve the quality of the rendering.

It should be stressed that, for solving most visibility problems, the graphics community has moved decades ago to hardware solutions such as dedicated z-buffer hardware or, more recently, GPUs (Graphics Processing Units). However, these solutions are mainly used for computing 2D images though very rarely for computing light-flow simulations in 3D scenes. The reason for this is that current hardware solutions are suited for dealing with two-dimensional sets of rays (for instance, a z-buffer can be used for efficiently computing the view from a point) but are not suited for general visibility queries which deal by essence with four-dimensional sets of rays (lines in 3D have 4 degrees of freedom). The panorama we present here does not recall the long body of work on hardware-oriented visibility solutions but focuses on algorithmic solutions for realistic rendering using accurate light-flow simulation in three-dimensional scenes.

In a given scene, two points are visible if the line segment joining them is free of any obstacles in the scene. The study of visibility is thus essentially the study of free line segments. The set of free line segments is bounded by segments that are tangent to the obstacles; in 3D, a line is tangent to up to 4 objects in general position. To put it another way, consider a moving viewpoint in a 3D scene. The view will change when a new object appears (or disappears) from behind an obstacle. When this happens there is a line passing through the viewpoint, tangent to the obstacle and arriving at the new object (and actually

tangent to it as well). It is thus imperative to fully understand tangencies between lines and obstacles in order to answer visibility queries.

Visibility information is commonly computed online when needed. Computations are local, restricted to the pairs of points for which visibility needs to be determined. Most such approaches fail to exploit the inherently global nature of visibility and the intrinsic consistency between visibility queries involving neighboring points.

In the global approach to visibility the idea is to compute, in a preprocessing step, most of the visibility information of the scene. This information is then stored in a data structure supporting queries of the type: are two elements of surfaces mutually visible? Our aim is to compute exact surface-to-surface visibility. Classical data structures based on spatial decomposition, such as octrees or BSP trees, are of little help in that respect. These structures are, indeed, designed to solve ray-shooting or point-to-point visibility queries, i.e., problems that are intrinsically of dimension two (the dimension of the set of lines going through a point), whereas surface-to-surface visibility involves the set of lines intersecting a surface, which is four-dimensional.

One data structure that has been proposed for global visibility is the visibility complex, which is a roughly partition of the space of maximal non-occluded line segments into connected cells of segments that are tangent to exactly the same objects. The structure was introduced by Pocchiola and Vegter in 2D [PV96a] and by Durand, Drettakis, and Puech in 3D [Dur99,DDP02]. In 2D, the visibility complex has been extensively studied [PV96a,Riv97,HH02,AP03] as well as its application for rendering [ORDP96,CF99]. In 3D, this structure has cells of dimension up to four, Durand et al. introduced, for practical reasons, the visibility skeleton, the structure that consists of the cells of dimension zero and one of the visibility complex [DDP97]. They proposed a proof-of-concept implementation showing that the skeleton leads to higher quality images in light simulation together with improved computation time compared to previous algorithms [DDP99].

Despite these positive results their pioneering approach suffered two major impediments: the poor performance of their algorithm, based on a systematic enumeration of worst-case complexity $\Theta(n^5)$ and observed complexity $\Theta(n^{2.5})$ [Dur99], and the lack of robustness of their implementation which required massive human intervention to remove degeneracies from the scenes. As a result, the largest scene they were able to handle had no more than 1,500 triangles [DDP99].

A few years later, Duguet and Drettakis proposed a practically robust implementation for computing a two-dimensional section of the visibility skeleton in which the lines supporting the maximal non-occluded line segments are restricted to be all concurrent (possibly at infinity) [DD02]. Their implementation was
successfully tested on scenes with over 100,000 polygons showing that, in the restricted context of punctual lights or lights at infinity, the approach leads to higher quality images in light simulation together with improved computation time compared to previous algorithms.

There are two reasons why little research has been done on the topic of 3D global visibility. One is that the problems are formidable because, as hinted earlier, the problems are intrinsically non-linear, even when dealing with polyhedral data, and are fairly high-dimensional since the space of lines in 3D is of dimension four. The other reason is that the space requirements of global visibility data structures have always been considered huge due to theoretical complexity bounds: the size of the visibility skeleton is \( \Theta(n^4) \) in the worst case for \( n \) triangles (as there are \( \Theta(n^4) \) lines tangent to 4 amongst \( n \) triangles) and \( O(n^{8/3}) \) in average for \( n \) uniformly distributed objects of roughly the same size [DDP02]. Durand [Dur99] observed a much smaller empirical size of \( \Theta(n^{2.4}) \), which was unfortunately still much too large for real-world graphics applications. However the small size of the scenes they were able to handle seriously diminishes the significance of this experimental asymptotic complexity since it is not clear that it corresponds to the asymptotic behavior.

We believe that these bounds are overly pessimistic for practical applications. In the last few years, we improved the known theoretical bounds on the visibility complex/skeleton to \( \Theta(n) \) if the scene consists of \( n \) uniformly distributed objects of roughly the same size [7], and to \( \Theta(n^2k^2) \) if the scene consists of \( n \) triangles that are organized into \( k \) convex polyhedra in arbitrary position and possibly intersecting [15]. In other words, taking into account the underlying structure of a scene provides new insight on the size of these data structures and gives evidence that they may be useful in real-world graphics applications.

We also made progress on the two major bottlenecks of the existing algorithms. On the algorithmic front we presented an algorithm for computing the visibility skeleton of \( k \) disjoint convex polyhedra in three dimensions of near-optimal complexity \( \Theta(n^2k^2 \log n) \) in the worst case [13]. We believe that this algorithm should have a much better complexity in practice and we just started its implementation. On the front of characterizing degeneracies surprisingly little was known. As an example, we only wrote last year a paper describing the possible degeneracies arising in lines tangent to polyhedra; whereas 4 lines in generic position admit up to 2 line transversals (a classical result), 4 segments in arbitrary position may admit up to 4 connected components of line transversals [25]. The situation for curved objects is much more challenging although we have recently characterized the sets of spheres that admit infinitely many tangents [3].

Our research objectives include pursuing the work of Durand, Drettakis and Puech to produce a full robust implementation of the visibility skeleton in 3D for polyhedra and apply it to computer graphics, in particular to discontinuity meshing for radiosity algorithms. More generally, we have identified three main directions of research. First, we need a better understanding of the properties of lines in space to design algorithmic solutions for 3D visibility. To that purpose, we plan to work on predicates, degeneracies, and combinatorics for lines. It should be noted that the theoretical study of the properties of lines in three-dimensional space is a fundamental line of research that is of interest independent of any direct application. Second, we want to improve the algorithmic tools for global 3D visibility data structures. In the polyhedral setting, where we have already developed an algorithm for computing the visibility skeleton of disjoint convex polyhedra, we will proceed on the implementation phase and develop effective algorithms for general polyhedral objects. In the realm of curved objects, we will build on our expertise and focus on the case of spheres, for which algorithmic solutions are now within sight. Finally, our goal is to apply our results, in particular to the graphics problems that motivated our studies.

We now detail, for each of these three main objectives, several problems we intend to address in the coming years.


3.1.2 Theory of lines in space

Participants: M. Glisse, X. Goaoc, H. Everett, S. Lazard, S. Petitjean.

On the topic of understanding the properties of lines in three-dimensional space, we are interested in degeneracies and predicates, combinatorial bounds, and topology of sets of lines.

Degeneracies and predicates. Understanding degeneracies with respect to visibility in three-dimensional scenes and studying the corresponding predicates are one of our topics of interest. For instance we study the predicates of seemingly simple and fundamental questions like: “how to test robustly whether four segments admit a line transversal?” and “does a line tangent to four triangles intersect a fifth triangle?”. These questions are crucial for global visibility data structures because their robust implementation requires testing whether a line segment tangent to four polyhedra is occluded by another polyhedron (if so the line segment does not correspond to a visibility event).

Combinatorial bounds. Very few results are known on combinatorial bounds on lines in three-space in the context of visibility. As an example, we recently asked how many tangents can four triangles have. Intuitively, one would think it should not be too large. We realized that the answer is unknown and that the problem is surprisingly hard. Though we proved a partial result that four triangles in generic position can have at least 88 and no more than 162 tangents [16], the problem remains open.

More fundamental questions concern the asymptotic combinatorial bounds of sets of lines or line segments in three-space under constraints. For example we work on showing that the worst-case complexity of the visibility complex/skeleton of a set of $n$ unit balls in 3D is $\Theta(n^4)$. Though negative, this result should be important in the sense that it would show that even for translated copies of a fat object the complexity of the visibility complex/skeleton is prohibitive in the worst case.

Another classic open problem we consider is to prove that a polyhedron with $n$ vertices that approximates a given surface has silhouettes of asymptotic complexity $O(\sqrt{n})$ under reasonable assumptions. Proving this conjecture, which tends to be considered as a fact in computer graphics, would typically provide some good theoretical ground to the study of algorithms complexity.

Topology. Extremely few results are known on the topology of sets of lines in the context of visibility. On that front we attack very elementary problems. For instance we ask whether the boundary of the four-dimensional cells of the visibility complex is connected. Another basic problem is to prove that the set of lines intersecting two disjoint spheres is homeomorphic to the Euclidean product of two disks. This problem which looks trivial, though it is not, is crucial for parameterizing such sets of lines and shows again how little is known on the topic.

3.1.3 Global 3D visibility data structures: algorithmics and implementation


We present here several problems, all on algorithmic and implementation issues of global 3D visibility data structures. The first four concern the 3D visibility skeleton for polyhedra. The last deals with the visibility graph for spheres, a first step in the realm of curved objects. We finally mention long-term objectives.

Implementation. As mentioned above, we recently developed an algorithm for computing the visibility skeleton of disjoint convex polyhedra in three dimensions [13]. Briefly, this algorithm consists in sweeping a plane around a reference edge (for each edge in turn) and maintaining in that plane the non-occluded tangents to two polygons; this can also be viewed as maintaining the two-dimensional visibility skeleton of the intersection of the scene with the sweep plane. Its complexity is $\Theta(n^2 k^2 \log n)$ in the worst case. We believe this algorithm to be effective, that is, to be implementable robustly and running reasonably fast, and we just started its implementation using CGAL and Angelier’s implementation of the 2D visibility complex [Ang02]. This implementation is however a major task which raises both theoretical

and practical problems.

Output-sensitive algorithms. We plan to work on improving the effective complexity of our algorithm. Our goal is to avoid as much as possible computing events that do not correspond to actual changes in the visibility, namely those that correspond to two bitangents merging, in the sweep plane, into a tritangent invisible from the reference edge. To that purpose, we plan to first turn our algorithm into a topological sweep, that is to process the events in an order consistent with their intrinsic dependencies rather than a geometric ordering. This approach has been considered with success by O. Hall-Holt [HH02] to maintain the view of a moving point in two dimensions, and S. Hornus (ARTIS research team) is currently trying to generalize it for the same problem in three dimensions.

Versatile algorithms. We also work on making the visibility skeleton algorithm, and ultimately its implementation, more versatile in the sense that we want to treat arbitrary polyhedra instead of only disjoint convex ones.

Queries. So far, the visibility complex and skeleton have been used for computing shadow boundaries. In this case the information is retrieved directly from the data structure. Another query we are interested in is computing the list of surfaces that are visible from a given surface. These queries cannot be answered directly; we need to develop additional dedicated data structures and algorithms.

Visibility graphs of spheres. Research on robust visibility computations amongst low-degree surfaces is really in its infancy. A first step, but nonetheless a challenge, is to compute the visibility graph of a set of spheres, that is to determine all the pairs that are mutually visible. The only algorithm known for this basic problem is brute-force: use an algebraic formulation to resort to tools from computer algebra. Practically, this approach fails even for small scenes consisting in no more than 50 spheres. We propose two new approaches. The first is to develop a set of tests that will rapidly find pairs of balls that are “clearly” visible or invisible and then resort to algebra for only a small number of difficult cases. These tests will be done by bucketing line-space. The second approach is to use a randomized algorithm with good expected performance. This algorithm is likely to be theoretically fast but not so fast in practice.

Long-term objectives. A longer-term line of research we are interested in concerns the scalability of 3D visibility global data structures. Ideally we would like to define a hierarchy of levels of detail for the visibility, in which a degree of myopia would be associated to each level. A relevant approach for these problems has been considered in two dimensions by Eggert et al. [EBD93] who used the concept of scale space [Wit86] for aspect graphs. They take into account both distance from viewer to object and relative sizes of features in order to develop a much smaller description, though still realistic, of the viewpoint space parcellation.

More long-term research directions include algorithmics for computing the 3D visibility complex (vs. skeleton), three-dimensional visibility among curved objects, introducing temporality (kinematic data structures), and online addition and deletion of objects (dynamic data structures).

3.1.4 Visibility and applications


We finally present two directions of research on the applications of 3D visibility, both related to photorealistic rendering.

Discontinuity meshing. One of the difficulties with the radiosity method in photorealistic rendering is the representation of shadow boundaries. In the radiosity approach, surface elements are subdivided and re-subdivided until the lighting across the entire element is more-or-less uniform. When there are sharp shadow boundaries it is a good idea to subdivide along those shadow boundaries; this produces

better quality images and reduces the number of subdivisions, thus improving the efficiency. The process is called discontinuity meshing and the visibility skeleton is a good structure for its solution. Indeed, as observed by Durand et al. [DDP97, DDP99], the 1D cells of the visibility skeleton correspond exactly to the shadow boundaries. As they point out, in order to be usable in practice, a robust and efficient implementation is required and this is what we plan to do.

Note that even if such an implementation turns out to be too cumbersome for use on large scenes, it would nevertheless be extremely useful as a tool for assessing the quality of other rendering solutions (hierarchical radiosity, Monte Carlo path tracing, photon mapping, …). Indeed, it is a problem today that the accuracy of shadow computation cannot be verified because there is no existing robust, exact solution of any kind. This remark also applies to other visibility computations such as hidden surface removal.

On-demand computing. The visibility skeleton is a data structure that contains visibility information for computing the discontinuity mesh. However, this structure is large and contains information about shadow boundaries that are not visible by a human being. Another approach that avoids precomputing the visibility skeleton consists in computing the relevant shadow boundaries on the fly in a hierarchical radiosity algorithm. Our algorithm for computing the visibility skeleton can be simply adapted to compute very efficiently a significant part of the boundary of the shadow cast by an area light source. This approach has the advantage of avoiding the storage of the data structure but has the drawback that some possibly visible shadow boundaries will be missed, in particular, those corresponding to shadows that are not induced by the boundary of the light source. Another advantage is that the on-demand computation that will be performed will always remain a subset of all the computation performed by our sweep-plane algorithm for computing the visibility skeleton.

3.2 Exact geometric computing for low-degree surfaces

Low-degree real algebraic surfaces (quadrics, cubics and quartics) allow for a good compromise between simplicity, flexibility and modeling power. They play a leading role in the construction of accurate computer models of physical environments for simulation and prototyping purposes, especially in the fields of industrial design, architecture and manufacturing. It is estimated that 95% of all mechanical parts are well-modeled by patches of natural quadrics (planes, spheres, cones and cylinders) and tori, which are degree four surfaces [RV82]. Also cubic surface patches are sufficient to describe the boundary of objects with arbitrary topology in a C1 piecewise-smooth manner [Baj93].

Despite this ubiquity and decades of research in the CAD/CAGD community, low-degree surfaces are far from being understood well enough to be robustly and efficiently manipulated by geometric algorithms. Our recent work on intersections of quadrics has proved that dramatic improvements can be accomplished when the right mathematics (linear and non-linear algebra, number theory, projective geometry) are put into motion [21, 23].

In this context, we plan to pursue two different lines of research: extend our work on intersections of surfaces to more primitives, other primitives and other dimensions, and find the right data structures, algorithms and predicates for efficiently computing arrangements of low-degree surfaces and related structures. We also aim at exploring the applications of our results.

We present below these two directions of research.

3.2.1 Intersection of surfaces

Participants: L. Dupont, S. Lazard, S. Petitjean.

Computing an explicit representation of the intersection of two general quadrics is a fundamental problem in solid modeling. The quadratic nature of the equations defining quadrics permits an explicit representation of their intersection curves. This problem has a long and rich history, going back to the work of J. Levin [Lev76]. But until recently no complete, general and usable solution had surfaced, despite many refinements to the original approach (see in particular [FNO89, WGT03]).

In [21], we presented the first exact, robust and general algorithm for parameterizing the intersection of two quadrics. This algorithm is optimal from the point of view of the functions involved and near-optimal from the point of view of the size of the field on which the coefficients of the parameterization are defined. Later, we proposed an efficient C++ implementation of this algorithm which uses arbitrary-precision integer arithmetic (see [23] and Section 5).

Many questions and problems remain open in the direct continuation of this work. We detail below some of these problems that we intend to address in the near future.

Filtered arithmetic implementation. Even though the implementation is already very efficient, it uses costly arbitrary-precision integer arithmetic for its computations. It is thus desirable to move to fixed-precision arithmetic and filters, and to rely on exact computation only when needed. We also aim at porting our code to the CGAL library.

Bit-length arithmetic optimality. The asymptotic bit length of arithmetic expressions involved in the parameterization of the intersection of quadrics can potentially be very large. When, for instance, the intersection curve is a smooth quartic, our algorithm outputs a parameterization that involves the square root of a polynomial function whose coefficients are polynomial expressions of degree 38 in the coefficients of the input quadrics [23]. It is currently unknown whether this degree is optimal. To minimize the arithmetic demands of a robust implementation using filtered arithmetic, it is crucial to identify predicates of minimal complexity at each branching of the algorithm.

Algebraic optimality. In its current state, the parameterizations output by our algorithm are, in some cases, only near-optimal, i.e., they involve a possibly unnecessary square root in the coefficients. Deciding whether this square root can be avoided (and removing it) amounts to deciding whether a particular curve or surface has a rational point (and computing such a point). Most of the near-optimal cases can be resolved by computing a rational point on a rational conic, a problem for which efficient solutions have recently appeared [CR03]. Integrating these solutions in our code would have a significant impact in practical situations.

Predicates. Geometric predicates of major interest in practice are those governing the question: “Do two given quadrics intersect?” This set of predicates is far from being well understood at this stage. A characterization of emptiness of intersection is known, but it does not lend itself well to an implementation with predicates.

Long-term objectives. In the longer term, we want to introduce dynamic aspects in the parameterization problem. In other words, given two quadrics whose coefficients are indexed by a parameter \( t \) (time, say), we would like a parameterization of the intersection of the two quadrics that is indexed by \( t \). This involves in particular identifying the time stamps at which the morphology of the intersection curve changes. Preliminary experiments indicate that this approach might be feasible at least in a restricted context.


An appealing generalization of the intersection problem would be to quadratic complexes. These piecewise-smooth surfaces can be seen as generalizations of polyhedra with faces embedded on quadratic surfaces and bounded by quadratic curves. Traditional representations of molecules are good examples of such surfaces. Also, it is important to be able to handle a more general class of surfaces. Tori are especially important in that respect, since they appear in all kinds of practical applications. More generally, we will investigate parameterized surfaces of bidegree \((1, 2)\) \cite{EGL04} and quadratically parameterizable surfaces \cite{CSS96}, which encompass quadrics and tori.

Another line of research we are interested in is the extension of our work on intersections of quadrics to higher dimensions. As already hinted at, quadrics appear naturally in visibility problems and common tangency of lines to a set of simple primitives can be formulated as the intersection of quadrics in \(\mathbb{P}^5(\mathbb{R})\). Additionally, statisticians are interested in determining whether \(d\)-dimensional ellipsoids intersect.

### 3.2.2 Arrangements of surfaces

Participants: L. Dupont, S. Lazard, S. Petitjean, M. Pentcheva.

In solid modeling, the two most widely used types of object representation are constructive solid geometry (CSG) and boundary representation (BRep). Both representations having their own respective advantages, solid modeling kernels often need an efficient and reliable way to switch from the CSG to the BRep representation. CSG-to-BRep conversion, also known as boundary evaluation, is a well-understood problem. However, past approaches have often put more emphasis on efficiency than on robustness and accuracy. Most current modelers use only finite-precision arithmetic for CSG-to-BRep conversion. The topological consistency of the computed BRep can easily be jeopardized by small amounts of error in the data introduced by finite-precision computations. For many applications in design and automated manufacturing, where topological consistency and accuracy are critical, this may be unacceptable.

A number of approaches have been proposed for the robust and accurate boundary evaluation of polyhedral models \cite{BMP94,For95}. Most rely heavily on numerical computation, with varying dependence on exact and floating-point arithmetic. Computing the topological structure of a BRep involves accurate evaluation of signs of arithmetic expressions. Assuming the input data has a bounded precision and allowing whatever bit length is necessary for number representation, these signs can be computed exactly.

By contrast, there has been much less work on robust CSG-to-BRep conversion algorithms for curved primitives. One notable exception is the work of Keyser, Krishnan and Manocha on the boundary evaluation of low-degree CSG solids specified with rational parametric surfaces \cite{KKM99}. The authors use exact arithmetic, present compact data structures for representing the boundary curves as algebraic curves and the boundary vertices as algebraic numbers and use efficient algorithms for computing the intersection curves of parametric surfaces. However, they assume that the input surfaces are in general position, i.e., degeneracies are not handled.

We now present the problems we plan to work on relative to arrangements of curved surfaces.

**CSG-to-BRep conversion.** In practical situations, objects are often in degenerate positions not by accident but by design. Accordingly, we want to focus on the robust boundary evaluation of CSG models built from volumes bounded by low-degree surfaces, typically quadrics, in arbitrary position. This actually

\[\text{EGL04} \quad \text{M. Elkadi, A. Galligo, and T. Le. Parametrized surfaces in } \mathbb{P}^3 \text{ of bidegree } (1, 2). \quad \text{In } \text{Proc. of International Symposium on Symbolic and Algebraic Computation}, \text{ pages 141–148, 2004.}\]


\[\text{For95} \quad \text{S. Fortune. Polyhedral modelling with exact arithmetic. In } \text{Proc. of ACM Symposium on Solid Modeling and Applications}, \text{ pages 225–234, 1995.}\]

was the original motivation of our work on the intersection of quadrics. Indeed, BReps are made of quadric surface patches bounded by curved segments which are portions of the intersection of the underlying quadrics.

While CSG-to-BRep conversion of entire models has a more combinatorial and algorithmic flavor than our work on the intersection of two quadrics, there are many mathematical and theoretical issues that are critical for its effective realization. Let us mention a few.

**Intersection of three quadrics in the parametric domain.** Vertices of the BRep model are intersections of triples of quadrics of the CSG model. Such vertices can be computed in the parametric domain of one of the quadrics from which it comes. Indeed, given three quadrics, their intersection corresponds to the intersection of the curve of intersection of two of them with the third. Once a parameterization of the intersection of the first two has been computed, and plugged in the equation of the third, finding the vertices of the triple amounts to solving a univariate polynomial equation whose coefficients are possibly algebraic. Solving equations of that kind is of interest in the computer algebra community and we have already experimented some strategies with researchers of the SPACES research team (now CACAO and SALSA).

**Predicates.** Since the expressions involved have very high degree (in a related context, that of sweeping an arrangement of quadrics, the appearance of a predicate of degree 256 was recently noted [MTT]), finding optimal sets of predicates is of the utmost importance. Related to the above, one such predicate (or set of predicates) should be able to decide the question: “Do three given quadrics have a common intersection?” Next to nothing is known about this question.

**Arrangements of quadrics.** Beyond computing BReps, another line of research is the computation of arrangements of quadrics, which has recently become a popular topic of research [MTT, SW]. This problem is intimately related to CSG-to-BRep conversion since the computed BRep is a sub-arrangement of the full arrangement determined by the surfaces of the scene. Computing BReps by first computing the whole arrangement is certainly not the way to go, but it is expected that research in one area will benefit from research in the other area.

**Embedding.** In the longer term, we will also have to consider the problem of “embedding” the geometrical objects computed. Indeed, knowing the exact geometric representation of an object (the BRep extracted from a CSG tree for instance) is not enough to know an embedding at fixed precision of this object, i.e. a correct machine realization. Negative complexity results show that computing a fixed-precision embedding that is topologically consistent with the combinatorial structure is a difficult problem. For instance, rounding the coefficients of the equations of the planes supporting the faces of a polyhedron without altering the combinatorial information is known to be NP-complete [MN90].

4 Application domains

4.1 Computer graphics

Our main application domain is photorealistic rendering in computer graphics. We are especially interested in the application of our work to virtual prototyping, which refers to the many steps required for the creation of a realistic virtual representation from a CAD/CAM model.

When designing an automobile, detailed physical mockups of the interior are built to study the design and evaluate human factors and ergonomic issues. These hand-made prototypes are costly, time-consuming, and take a long time to produce. With virtual prototyping, this process can be significantly accelerated and made more efficient.

The application of our work to virtual prototyping involves computing the BRep of the interior of the vehicle. This requires a detailed and accurate model of the interior surfaces, including those that are not visible from the exterior.

**References**

consuming, and difficult to modify. To shorten the design cycle and improve interactivity and reliability, realistic rendering and immersive virtual reality provide an effective alternative. A virtual prototype can replace a physical mockup for the analysis of such design aspects as visibility of instruments and mirrors, reachability and accessibility, and aesthetics and appeal.

Virtual prototyping encompasses most of the work on effective geometric computing we envision. In particular, our work on 3D visibility should have fruitful applications in this domain. As already explained, meshing objects of the scene along the main discontinuities of the visibility function can have a dramatic impact on the realism of the simulations.

4.2 Solid modeling

Solid modeling, i.e., the computer representation and manipulation of 3D shapes, has historically developed somewhat in parallel to computational geometry. Both communities are concerned with geometric algorithms and deal with many of the same issues. But while the computational geometry community has been mathematically inclined and essentially concerned with linear objects, solid modeling has traditionally had closer ties to industry and has been more concerned with curved surfaces.

Clearly, there is a considerable potential for interaction between the two fields. Standing somewhere in the middle, our project has a lot to offer. Among the geometric questions related to solid modeling that are of interest to us, let us mention: the description of geometric shapes, the representation of solids, the conversion between different representations, data structures for graphical rendering of models and robustness of geometric computations.

4.3 Other potential applications

The VEGAS project proposal emanates from a research team (ISA) which has been at the forefront of efforts in realistic rendering to use radiosity methods in real-world applications. This is the historical motivation for the application domains outlined above we currently target. However, it should be clear that progress on the understanding of lines in space and on the efficient handling of low-degree curved objects would also impact other domains. Computer vision, robotics and computational biology are some that come to mind.

5 Software aspects

5.1 General policy

Recall that our main scientific objective is to contribute to the unfolding of an effective geometric computing dedicated to non-trivial geometric objects, and that, included among our objectives are the study and development of new algorithms for the manipulation of geometric objects, the experimentation of algorithms, and the production of quality software.

Our policy on software development goes in three phases. First we aim at developing proof-of-concept, research implementations of our new algorithms. These implementations are internal prototypes whose purpose is not to be openly distributed. Second, when the prototypes are promising, we endeavor to develop quality software that is reliable and efficient enough to be distributed. Third, we intend to assist the transfer of our reliable software to global, integrated, production-quality solutions when this is appropriate. Since we do not have the manpower for making these transfers ourselves, this will have to rely on third-party personnel.

In terms of licenses, our policy is to distribute our software freely for non-commercial use. Typically, we plan to use the INRIA and/or the QPL licenses. This can however change depending on the context.

The development of our code for intersecting quadrics is a good example of our policy. We first developed a prototype of our quadric intersection software in MuPAD [MuP] before developing a C++
reliable, efficient and complete implementation, QI, which is now distributed [QI]. Entering the third phase, we now intend to integrate our software into more global libraries. Recently, L. Dupont made that transfer into the Exacus library [EXA] developed at the Max-Planck-Institut für Informatik during a long visit in Saarbrücken. We also plan to assist the transfer of our code into a CGAL package [CGA]. QI is currently distributed under the free for non-commercial use INRIA license and will be distributed very shortly under the QPL license.

5.2 Software developed by team members

**QI.** QI stands for “Quadrics Intersection” [QI]. QI is the first exact, robust, efficient and usable implementation of an algorithm for parameterizing the intersection of two arbitrary quadrics, given in implicit form, with integer coefficients. This implementation is based on the parameterization method described in [21, 23] and represents the first complete and robust solution to what is perhaps the most basic problem of solid modeling by implicit curved surfaces.

QI computes an exact parameterization of the intersection of two quadrics with integer coefficients of arbitrary size. It correctly identifies, separates and parameterizes all the connected components of the intersection and gives all the relevant topological information. The parameterizations computed are optimal in terms of their defining functions and near-optimal in terms of the size of the extension on which their coefficients are defined. QI can routinely compute parameterizations of quadrics having coefficients with up to 50 digits in less than 100 milliseconds on an average PC.

QI is written in C++ and builds upon the LiDIA computational number theory library [LiD] bundled with the GMP multi-precision integer arithmetic [GMP]. Our implementation consists of roughly 18,000 lines of source code. QI has being registered at the Agence pour la Protection des Programmes (APP). It is distributed under the free for non-commercial use INRIA license and will be distributed very shortly under the QPL license. QI was first officially released in June 2004.

**Visibility Skeleton.** We are in the very early stages of developing a robust and efficient implementation for the construction of the visibility skeleton of a polyhedral scene. This development will serve as a focal point for a large part of our work on practical 3D visibility.

5.3 Other software used

**CGAL.** CGAL is the outcome of a collaborative effort of several sites in Europe and Israel [CGA]. It makes the most important of the solutions and methods developed in computational geometry available to users in industry and academia in a C++ library. CGAL provides easy access to useful and reliable geometric algorithms. Starting from version 3.0, CGAL is available under an Open Source License.

We use CGAL as a platform for development of prototypes when the context is appropriate. For instance we are developing our Visibility Skeleton software in CGAL, using the kernel, the basic library and the 2D visibility complex package developed by P. Angelier (ENS Ulm, GéCoaL team). But we based the development of our quadric intersection software (QI) on the LiDIA library because the functionalities needed were more related to number theory than geometric computing.

We also plan to assist in the development of CGAL packages as described in Section 5.1.

---


[LiD] LiDIA: a C++ library for computational number theory. Darmstadt University of Technology. [http://www.informatik.tu-darmstadt.de/TI/LiDIA](http://www.informatik.tu-darmstadt.de/TI/LiDIA).

**FGb/RS.** FGb/RS is a C++ integrated suite for exactly solving systems of polynomials equations and inequations\[^{[FGb]}\]. It combines software for efficiently computing Gröbner bases (the FGb part, developed by Jean-Charles Faugère) and for the study of real roots of polynomial equations (the RS part, developed by Fabrice Rouillier). FGb/RS is free for non-commercial use. We currently use FGb/RS as a research tool.

6 Scientific collaborations

6.1 Collaborations within INRIA

Our major scientific collaborations within INRIA are with research teams of Theme Sym B (“Algebraic and geometric structures, algorithms”). In particular, we have a fruitful and long-standing collaboration with people of the group GEOMETRICA (formerly PRISME), principally Olivier Devillers. Common work on 3D visibility has led to several international publications \[^{[7, 15, 16]}\].

We frequently discuss and exchange with researchers of the SPACES research team (and now also with its offspring SALSA) who specialize in polynomial system solving. Indeed, many problems in 3D visibility and effective geometric computing boil down to the study of algebraic varieties and real zeros of polynomials. In recent years, we have been constant providers of new problems for the RS software, developed by Fabrice Rouillier. Also, our work on the parameterization of intersection of quadrics has been conducted jointly with D. Lazard and has led to publications \[^{[21, 26]}\].

We also are in close contact with members of the GALAAD research team on the subject of geometric computing for low-degree curved surfaces. People in the GALAAD team have also worked on intersections and arrangements of quadrics, though from a different perspective. Recently, Monique Teillaud (GALAAD) and Sylvain Pion (GEOMETRICA) have proposed the creation of a new team focused on advancing the CGAL library and improving the transfer of geometric code to the industrial world. A member of the VEGAS team has been invited to join the CGAL Advisory Board.

Outside Theme Sym B, we also maintain close contact with ALICE (research team proposal, offspring of ISA) on the applications of our work to photorealistic rendering, and with REVES and ARTIS (with C. Puech and S. Hornus) on three-dimensional visibility computation.

In recent years, we have had many opportunities to exchange ideas and viewpoints on areas of common interest with people in the above teams, notably in meetings of INRIA ARCs Geometrica (1999-2000), Visi3D (2000-2001), CoSTIC (2000-2001) and Docking (2003-2004), or at the workshops of the ECG European project (2001-2004) to which we were invited. Members of GEOMETRICA, SALSA, REVES, and ARTIS have also participated in the annual international workshop on three-dimensional visibility we have organized in the last four years.

6.2 Other collaborations in France

Michel Pocchiola of the ENS Ulm is one of the principal researchers worldwide in two-dimensional visibility. Although we have no formal collaboration or publications with Michel we have frequent discussions. Also his group’s implementation of the two-dimensional visibility complex will be a key component of our three-dimensional version.

6.3 International collaborations

On the international scene, we have an ongoing formal collaboration in the form of an INRIA associated team with the Computational Geometry Laboratory at McGill University, Montreal, on the subject of three-dimensional visibility (see the web page\[^{[FGb]}\]). In this context, we have co-organized with Sue White-


\[^{5}\]http://www.loria.fr/~everett/McGill-ISA.
sides (of McGill) several international workshops (1st, 2nd, 3rd, and 4th McGill - INRIA Workshops on Computational Geometry in Computer Graphics). Also, we co-supervise with Sue Whitesides at PhD student, Linqiao Zhang.

Through these McGill-INRIA workshops and other exchanges, we have started many long-term and fruitful collaborations. In particular we note the collaboration with Hervé Brönnimann of New York Polytechnic University [15, 16, 17], as well as with mathematicians (algebraic geometers), Ciprian Borcea of Rider University [3, 24] and Frank Sottile of the University of Massachusetts-Amherst [16, 17].

Jeff Erickson of University of Illinois at Urbana-Champaign (UIUC) has also been a participant of the last McGill-INRIA workshop. Along with other colleagues at UIUC (Steve LaValle in robotics, Jean Ponce in computer vision and John Hart in Computer Graphics), he has been the recipient of a NSF ITR grant on “Making 3D Visibility Practical” (2002-2005). We have recently started exchanging on that topic of common interest. Prof. Erickson visited VEGAS members for three months in Fall 2004 and we have since submitted a CNRS-INRIA-UIUC Cooperation proposal.

In Europe, we enjoy a long-term collaboration with Otfried Cheong, of the Technical University of Eindhoven, on planar ray shooting [4], geometric spanners [2], and on line transversals and geometric permutations [5, 20]. This latter work is joint with Hyeon-Suk Na, former postdoc in the ISA research team, now professor at Soongsil University, South Korea. Otfried Cheong will be moving to the Korea Advanced Institute of Science and Technology (KAIST) in 2005 at which time we expect to establish a formal collaboration with South Korea.

We also have many discussions on effective computational geometry with members of the Algorithms and Complexity Group of Max-Planck-Institut für Informatik in Saarbrücken, Germany (Lutz Kettner, Nicola Wolpert and former member Elmar Schömer, now at Johannes Gutenberg Universität Mainz). One particular topic of common interest is arrangements of quadrics. We co-organized with them a Lorraine-Saarland Workshop on Geometry and CAD which took place at MPII in April 2003. Laurent Dupont visited MPII for three months in 2004.

We also have fruitful contacts with Gert Vegter, of the High-Performance Computing and Imaging Group at the University of Groningen, on geometric computing. He participated in our INRIA ARC Visi3D (2000-2001), and he will be visiting us in February 2005 for a month.

Finally more punctual collaboration with researchers from the Freie Univ. of Berlin, Univ. Utrecht, JAIST Tokyo, HKUST Hong Kong, Stanford Univ., and UPC Barcelona resulted in the following publications that are related to visibility [2, 4, 6, 8, 14].

It should noted that, along with specialists of computational geometry (Cheong, Devillers, Erickson, Whitesides), we have an increasing number of collaborations with mathematicians and computer algebraists (Borcea, D. Lazard, Rouillier, Sottile). This is very much in line with the strong theoretical inclination of our scientific objectives.

7 Industrial partners

The geometry group of ISA, now identified as Vegas, has been in existence for only five years. It has spent most of its times establishing itself scientifically and has not actively sought industrial contacts. Accordingly, we have only one example of concrete industrial realization, with the CIRTES company. With the other two industrial partners mentioned below, we have high-level exchanges around the idea of integrating our algorithms in their products. But they are mainly interested in global solutions and our results on 3D visibility and low-degree surfaces are too preliminary at this stage to be transferred.

VSP Technology. VSP Technology (for Visualization Solutions Provider) is a start-up company born from a technology transfer from INRIA which provides software and services for realistic and interactive visualization of complex 3D models. It was created in 2001 by two former PhD students of the ISA.
research team. The core technology marketed by VSP Technology relies on an advanced rendering software developed within ISA.

In the long run, our work on 3D visibility could have a profound impact on the VSP rendering engine. Visibility, which is currently done online and does not exploit the intrinsic consistency of ray space, is often the bottleneck of realistic rendering simulations. It is expected that our global, generic approach to visibility problems will lead to improvements in the performance cost of visibility queries.

**SGDL Systems.** SGDL Systems Inc.\(^7\) develops and licenses core 3D software and hardware to efficiently encode and decode complex objects and scenes resulting in interactive computer graphics with textures, cast shadows and model behavior within an extremely small memory footprint and file size. SGDL commercially exploits a mathematical approach called Arithmetic of Forms, which provides an elegant and powerful solution to solid modeling. In particular, the company markets a 3D engine based on low-degree projective algebraic surfaces to efficiently represent massive scenes.

Our contacts and exchanges with SGDL Systems are long-standing. The technology developed by SGDL is based on the CSG paradigm, i.e., objects are Boolean combinations of basic volumes. To open itself to other technologies and notably to geometric modelers and sculptured models, the SGDL geometry engine must be able to mix its own objects with BRep objects. This involves, in one form or another, conversions of representation that our work on low-degree surfaces and intersections of such surfaces can help accomplish robustly and efficiently.

**CIRTES.** CIRTES\(^8\) is a research company in rapid prototyping. We have exchanges with CIRTES since we started in 2000 co-supervising a CIFRE PhD student, G. Lauvaux.

The objective of fast prototyping in today’s manufacturing industry is to produce a physical model from a virtual one. Such a prototype should be cheap and easy to produce. Two popular technologies are NC machining in which an object is constructed using a 5 degrees of freedom drilling machine and layered manufacturing in which the object is built up from several layers.

Stratoconception\(^9\) is the main layered-manufacturing technique developed at CIRTES [Bar92]. In this technique a polyhedron is manufactured in fairly thick slices (e.g. 1 cm), each slice being manufactured by a two and one-half degrees of freedom tool. Not all objects can be perfectly manufactured that way, because some regions are not accessible to the drill, and choice of drilling direction and position of the slices have a large impact on the quality of the finished product. In fact, if the prototype is made of wood, manufacturing inaccuracies are repaired by manual sanding, a very time-consuming process. Once the positions of the slices have been chosen and the direction of drilling fixed, the error is exactly the same as the volume of the shadow cast on each slice by a light at infinity in the drilling direction.

Computing the optimal slicing strategy and drilling direction so as to minimize the volume of inaccessible regions is an open problem. We have proposed a practical approximate solution to this problem which has been tested with success and is now used by CIRTES [22]. Computing a provably optimal solution remains however an open problem. Note that the notion of 3D visibility is a key in this work since a tool can only drill material that is accessible, that is, that it can “see”.

8 Positioning

8.1 Our originality

Our team is one of the few worldwide to attack all the aspects, from theory to practice, needed for the development of an exact, effective geometric computing dedicated to non-trivial 3-dimensional objects.

\(^7\)http://www.sgdl-sys.com.
\(^8\)http://www.cirtes.fr.

Among the main characteristics of our project proposal are: the belief that the classical geometric and algebraic machinery, when adroitly set into motion, can dramatically enhance the algorithmic knowledge concerning 3D objects; the understanding that practical efficiency is as important as theoretical complexity in algorithmic design; and the willingness to cover the whole spectrum of exact geometric computing going from a proven characterization of geometric degeneracies and the identification of optimal predicates to the production of versatile, reliable and scalable quality software.

8.2 Positioning in the scientific community

Effective computing with advanced geometric primitives is currently a major concern in the international computational geometry community. Suffice it to mention the European Project IST-2000-26473 ECG (“Effective computational geometry for curves and surfaces”), which will probably have a sequel after its end in 2004, and the NSF Program CARGO (“Computational and algorithmic representations of geometric objects”). While more mathematically oriented, and only partially concerned with exact computation, the European Project IST-2001-35512 GAIA II (“Intersection algorithms for geometry based IT-applications using approximate algebraic methods”) should also be mentioned: its stated objectives are to improve intersection algorithms by integrating knowledge from real and classical algebraic geometry in CAD, and to span the whole chain from basic research to industrial prototyping.

Many research teams have embarked on establishing the foundations of a robust geometric computing. Our main “competitors”, with which we enjoy close ties, are to be found among the members of the ECG European project. Outside this sphere, our research on low-degree curved objects adds robustness and effectiveness concerns to related work conducted in more applied fields like CAGD, geometric modeling and computer graphics. The Computer Graphics Group at Hong Kong University (Wenping Wang), the Graphics and Geometric Design lab at Rice University (Ron Goldman) and the GAMMA (Geometric Algorithms for Modeling, Motion and Animation) research group at University of North Carolina, headed by Dinesh Manocha, are some of the important actors on this scene.

The most important groups working on 3D visibility from a theoretical viewpoint are to be found in Tel Aviv (Micha Sharir), Duke University (Pankaj Agarwal) and Utrecht University (Mark de Berg). In the graphics community there are many groups; those most related to our research include the team at MIT (Frédo Durand, Seth Teller, Julie Dorsey) and of course the INRIA teams ARTES and REVES. Finally, the group at UIUC (Jeff Erickson, John Hart, Steven Lavalle) has a new NSF ITR entitled “Making 3D Visibility Practical” which addresses the problem of transferring theoretical visibility results to graphics.

The main international publications we aim at are:


8.3 Positioning with respect to other INRIA projects

Our project falls within the scope of one of the seven great scientific challenges identified in INRIA’s Strategic Plan for the years 2003-2007: “Combining simulation, visualization and interaction”.

The Strategic Plan notes that “interactions between computer science and applied mathematics are continuously on the rise and are essential to take up the new and emerging challenges in Communication
Since geometric computing is creeping into virtually every corner of science and engineering, geometric representations will increasingly play a central role similar to what images have been for many disciplines. A major challenge is thus to “develop a theory of geometric information” combining advances in “geometry, statistics, algorithmic, symbolic computation, scientific computing, and image analysis and synthesis”. Research conducted by VEGAS team members clearly contribute to tackle this challenge.

For reasons that are both historical and related to our preferred application area (computer graphics), we position ourselves with respect to projects in both Theme Sym B (“Algebraic and geometric structures, algorithms”) and Theme Cog D (“Image synthesis and virtual reality”).

**GEOMETRICA (Sym B).** GEOMETRICA (INRIA Sophia-Antipolis) is a follow-up team to PRISME. PRISME is assuredly where most of the action took place on the French computational geometry scene in the last decade. It advanced the state of the art in robustness issues and geometric software engineering. A notable outcome of research conducted within PRISME (in connection with European partners) has been the development of the CGAL library. While pursuing similar lines of research, GEOMETRICA has put more emphasis on building an effective computational geometry for meshed surfaces.

GEOMETRICA is the INRIA research team that has the closest scientific objectives to ours. Among the concerns that we share are the reliability, robustness and scalability of geometric algorithms. There are important differences though. While GEOMETRICA handles curved shapes by controlled geometric approximation, we work on the exact geometry of objects, at the price of being more modest as to the class of surfaces that we consider. Also, research on 3D visibility is a real specificity of our project, though some work in this area has been conducted jointly with Olivier Devillers of the GEOMETRICA research team.

**GALAAD (Sym B).** GALAAD (INRIA Sophia-Antipolis) focuses on effective algebraic geometry and its applications. The objective is to develop algorithmic methods for efficiently solving geometric and algebraic problems arising in such diverse domains as CAD, robotics, computer vision and computational biology.

Research conducted within GALAAD has a much more algebraic flavor than ours. While algebra is quite present in our work, our culture is much more of an algorithmic kind. In particular, we do not conduct research in algebra, we simply use it as a tool (along with many other mathematical disciplines) to characterize and handle degeneracies in robustness issues and solve related problems.

It should however be mentioned that the domains of research of GALAAD, GEOMETRICA and VEGAS are closely related.

**ARTIS and REVES (Cog D).** Historically, the idea of using 3D global visibility data structures in image synthesis came out of the iMAGIS research team (INRIA Rhône-Alpes), around Frédéric Durand, George Drettakis and Claude Puech. In 2003, iMAGIS was replaced by two new projects, one being the ARTIS research team. At the same time, G. Drettakis moved to INRIA Sophia-Antipolis and created REVES.

One of the main goals of the ARTIS team is the definition of a generic framework for the creation of synthetic images, integrating elements of 3D geometry, of 2D geometry, and of appearance (photometry, textures) in particular. Contrary to its predecessor team, ARTIS does not have as stated scientific objective the investigation of fundamental geometric problems. It maintains nonetheless an activity in 3D visibility for dynamic environments, through Claude Puech and his PhD student Samuel Hornus.

REVES’ research focuses on the computer rendering of images and sound, with a view to developing novel algorithms to improve the speed and quality of computer-generated images and spatialized audio. Emphasis is put more on acceleration techniques allowing a real-time use of image synthesis and virtual reality applications than on robustness and exactness. George Drettakis and Florent Duguet have however worked in 2001/2002 on the notion of ε-visibility.
ALICE (Cog D). ALICE is the research team proposed by Bruno Lévy at INRIA Lorraine. It is one of the three offsprings of ISA. The main scientific objective of ALICE is to contribute to a theory of numerical approximation of shape and its properties, taking into account sampling issues and preserving the topology of objects.

As already hinted at, ALICE and VEGAS share common application areas and are both interested in algorithmic issues in geometric computing. But time scale, tools and community are different: numerical geometry and computer graphics on one side, exact, robust geometry and computational geometry on the other side.

9 Scientific animation

Workshops organization. Over the last years we have organized annual international workshops on 3D visibility [W00,W02,W03,W04,W05] (in conjunction with S. Whitesides, McGill Univ., since 2002). These workshops typically regroup around 15 people, both students and renowned international researchers, for a week of work on a chosen set of specific problems. We also organized the french computational geometry days [JGA] in 2002 and co-organized a one-day workshop regrouping our team and the Algorithms and Complexity Group from MPII [LSW].

Participation in the training of highly qualified personnel. H. Everett is professor at Univ. Nancy 2. She and S. Lazard share a DEA (master level) course on computational geometry in Nancy. Two of our students, X. Goaoc and L. Dupont, defended their PhD in 2004, and another one, G. Lavioux, has virtually finished. We also supervise three other students, M. Gtoise, L. Zhang, and M. Pentcheva, who started their PhD in Sept. 2003, Jan. 2004, and Oct. 2004, respectively. In the last years, we supervised two postdocs, C. Lamathe from Nov. 2003 to Nov. 2004 and H.-S. Na from March 2001 to Aug. 2002 who is now assistant professor at Soongsil University, South Korea. Finally, we also regularly supervise Masters students.

Committees. H. Everett was a member of the program committees of the “ACM Symposium on Computational Geometry” (SoCG) in 2000, and of the “Canadian Conference on Computational Geometry” (CCCG) in 1999 and 2002. She was also a member of the NSERC (the NSF equivalent for Canada) grant committee in 2000. S. Petitjean was a member of the program committee of the “International Conference on Computer Vision” (ICCV) in 2001 and 2003, and is a member of the program committee of the “Computer Vision and Pattern Recognition” conference (CVPR) in 2005. He has also been chairman of the PhD thesis award committee of the “Association Française d’Informatique Fondamentale” (AFIF) since 2003 and member of this committee since 2001.

Participation to laboratory life. H. Everett has been a member of the hiring committee of Univ. Nancy 2 since 2000. She is on the “Conseil du laboratoire” since 2001 and has run the international relations

office of the lab between 2001 and 2004. S. Petitjean was member of the hiring committee of Univ. Nancy 1 between 1996 and 2004.

**Initiative actions.** We have conducted the McGill-ISA associated team since 2002. We ran the ARC (INRIA new investigation grant) “3D Visibility: Theory and Applications” in 2000-2001. We participated in the ARCs “Geometrie: Axiomatization du Calcul Geometrique” (1999-2000) and “CoSTIC: Courbes et Surfaces: Traitement, Interprétation, Calcul” (2000-2001). We also participated in the AS CNRS (CNRS working group) “Algorithmic and Discrete Geometry” in 2004. Finally, we conducted a CNRS “Action Thématique et Incitative sur Programme” (ATIP) in 2002-2003 and currently conduct a ministry “Action Concertée Incitative” (ACI) entitled “Effective geometry for realistic visualization of complex scenes” for the period 2003-2006.

**Visiting scientists.** We have had many international visitors in the last years: S. Whitesides (McGill Univ., 9 weeks since 2001), J. Erickson and K. Whittlesey (Univ. of Illinois at Urbana-Champaign, 3 months in 2004), D. Bremner (Univ. of New Brunswick, Canada, 6 weeks in 2003), I. Streinu (Smith College, USA, 1 month in 2003), C. Borcea (Rider Univ., USA, 1 month in 2003), O. Cheong (Utrecht Univ., The Netherlands, 1 month in 2002), S. Wismath (Univ. of Lethbridge, Canada, 5 weeks since 2002).

### 10 Publications

Only the publications related to the topic of the proposal are listed here. Other publications can be found on the home pages of project members. Summaries of representative publications are given in appendix.

**International journals**


**PhD thesis**


**International conferences**


27
Submissions / manuscripts


Appendix: summaries of representative publications

A The Expected Number of 3D Visibility Events is Linear


We show that, amongst \( n \) uniformly distributed unit balls in \( \mathbb{R}^3 \), the expected number of maximal non-occluded line segments tangent to four balls is linear, considerably improving the previously known upper bound of \( O(n^{6/3}) \) \([DDP02]\). Using our techniques we show a linear bound on the expected size of the visibility complex, a data structure encoding the visibility information of a scene, providing evidence that the storage requirement for this data structure is not necessarily prohibitive.

Our results generalize in various directions. We show that the linear bound on the expected number of maximal non-occluded line segments that are not too close to the boundary of the scene and tangent to four unit balls extends to balls of various but bounded radii, to polyhedra of bounded aspect ratio, and even to non-fat 3D objects such as polygons of bounded aspect ratio. We also prove that our results extend to other distributions such as the Poisson distribution. Finally, we indicate how our probabilistic analysis provides new insight on the expected size of other global visibility data structures, notably the aspect graph.

These results imply that the visibility complex is a data structure whose size might be tractable for realistic graphic scenes or, more precisely, that the visibility complex is not necessarily unusable in practice. However computing the visibility complex (or its size) for realistic graphic scenes in a reasonable amount of time would require an output-sensitive algorithm which does not yet exist.

The figure represents \( n \) balls of unit radius whose centers are uniformly distributed in a spherical universe whose radius depends on \( n \) such that \( n \) is equal to the volume of the universe times a constant representing the density of the \( n \) balls. In this figure, the density is 0.01.

---

B Geometric Permutations of Disjoint Unit Spheres


A line transversal for a set $F$ of $n$ pairwise disjoint convex bodies in $\mathbb{R}^d$ is a line $\ell$ that intersects every element of $F$. A line transversal induces two linear orders on $F$, namely the orders in which the two possible orientations of $\ell$ intersect the elements of $F$. Since the two orders are the reverse of each other, we consider them as a single geometric permutation.

Bounds on the maximum number of geometric permutations were established about a decade ago. In two dimensions, a tight bound of $2n - 2$ is known [ES90]. In higher dimension, the number is asymptotically in $\Omega(n^{d-1})$ [KLL92] and in $O(n^{2d-2})$ [Wen90].

A number of particular cases were investigated in an attempt to close this gap. When the objects are spheres [SMS00] or “fat” convex objects [KV01], the number is in $\Theta(n^{d-1})$. For sufficiently large families of unit spheres, the bound is $2$ in $\mathbb{R}^2$ [Asi98] and at most $4$ in higher dimension [KSZ03]. In this paper, we improve this latter bound.

**Theorem.** A family of $n$ disjoint unit spheres in $\mathbb{R}^d$ admits at most two distinct geometric permutations if $n \geq 9$, and at most three otherwise.

The key ingredient of this paper is to bring together combinatorial results and geometrical insight.

We first establish that given three distinct geometric permutations on more than $4$ symbols, $4$ of the symbols must realize a so-called incompatible pair. Then, we prove that the incompatible pairs cannot be simultaneously realized as geometric permutations of disjoint unit spheres in $\mathbb{R}^d$, provided the number of spheres is greater than $9$. This second step involves understanding the intricate geometry of lines intersecting a family of spheres with conditions on the ordering. For instance, we prove that any line that intersects three disjoint unit spheres $A$, $B$, and $C$ in that order makes an angle of at most $\pi/4$ with the line through the centers of $A$ and $C$ (a geometric result that also settles a conjecture in [HKL03]).

A subject closely related to bounding the number of geometric permutations is the study of Helly-type theorems for line transversals. A recent result [HKL03] proves the existence of a number $n_0$ such that for any set $\mathcal{F}$ of disjoint unit spheres in $\mathbb{R}^3$, if every $n_0$ members of $\mathcal{F}$ have a line transversal, then $\mathcal{F}$ has a line transversal. In this paper we also improve the previously known upper bound on $n_0$ from $46$ to $18$.

---


Common Tangents to Spheres in $\mathbb{R}^3$

C. Borcea, X. Goaoc, S. Lazard and S. Petitjean.

A major issue in geometric computing is to handle degenerate inputs properly in order to design robust algorithms. This often requires recognizing such an input to begin with. In 3D visibility problems, objects whose common tangents are of improper dimension are in degenerate configurations. In this paper, we determine all degenerate configurations of four distinct spheres, that is all configurations of four spheres with infinitely many common tangents.

This problem, raised in the early 90’s, has stood as a challenge for several years. A breakthrough came in 2001 with a characterization of the degenerate configurations of unit spheres $^{[MPT01]}$. Since then, Megyesi, Sottile, and Theobald went on to describe the degenerate configurations of $k = 3, 2, 1$ lines and $4 - k$ spheres. In this paper, we completely solve the problem for four spheres.

**Theorem 1.** Four distinct spheres in $\mathbb{R}^3$ have infinitely many real common tangent lines if and only if they have aligned centers and at least one real common tangent.

Our approach is to describe the common tangents to four spheres as the solutions of a polynomial system whose coefficients depend on the centers and radii of the spheres. The problem, then, amounts to determining the values of these parameters for which the polynomial system has infinitely many real solutions. The difficulty, classical in algebraic geometry, resides in eliminating imaginary solutions and solutions at infinity.

Theorem 1 allows us to give a complete geometric characterization of the degenerate configurations of 4 spheres, namely:

**Theorem 2.** If four distinct spheres in $\mathbb{R}^3$ admit infinitely many real common tangent lines then either all four spheres intersect in a circle (possibly degenerating to a point), or each sphere has a circle of tangency with one and the same quadric of revolution with symmetry axis the line passing through all centers. This quadric can be a cone, a cylinder or a hyperboloid of one sheet (see the figure).

We conclude this study by giving a degree-five predicate to detect these degenerate configurations.

---

D Near-Optimal Parameterization of the Intersection of Quadrics

L. Dupont, D. Lazard, S. Lazard and S. Petitjean.

We present an algorithm that computes an exact parametric form of the intersection of two real quadrics in three-dimensional space given by implicit equations with rational coefficients. This algorithm represents the first complete and robust solution to what is perhaps the most basic problem of solid modeling by implicit curved surfaces.

The functions parameterizing the intersection contain no square root whenever it is possible and the coefficients of these functions are algebraic numbers with at most one extra square root. Furthermore, for each geometric type of intersection, the number of square roots in the coefficients is always minimal in the worst case.

These results are formalized in the following theorem.

**Theorem.** In \( \mathbb{R}^3 \) or \( \mathbb{P}^3(\mathbb{R}) \), given two quadrics by their implicit equations whose coefficients are rational, the algorithm of this paper computes a parameterization of their intersection such that each coordinate in projective space is polynomial over an extension field \( \mathbb{K} \) of \( \mathbb{Q} \) if such a parameterization exists, or is a polynomial in \( \mathbb{K}[\xi, \sqrt{\Delta}] \), where \( \xi \) is the (real) parameter and \( \Delta \in \mathbb{K}[\xi] \) is a polynomial in \( \xi \).

In both cases, the parameterization is either optimal in the degree of the field extension \( \mathbb{K} \) needed to represent its coefficients or may involve one (and only one) possibly unnecessary square root. In the latter situation, testing for optimality (i.e., determining whether the extra square root is necessary or not) and finding an optimal parameterization are equivalent to finding rational points on a conic or a degree-eight surface, a long-standing open problem in algebraic geometry.

Our work builds upon a large body of literature on intersections of quadrics, dating back to Levin (1976). Among the main contributions of our approach and the main novelties of this paper are:

- computations take place over the real projective space \( \mathbb{P}^3(\mathbb{R}) \) and no longer over \( \mathbb{R}^3 \) as in most (if not all) past work on the subject;
- we show that any pencil generated by two distinct quadrics with rational coefficients contains at least a ruled quadric with rational coefficients except in one very simple case;
- we give new parameterizations of projective quadrics that are optimal in the worst case in the number of radicals they involve;
- we give a complete classification of the type of intersection in \( \mathbb{P}^3(\mathbb{R}) \) in terms of the number and multiplicity of the multiple roots of some degree-four polynomial and in terms of the rank of the associated matrices.
E Intersecting Quadrics: An Efficient and Exact Implementation

S. Lazard, L. M. Peñaranda and S. Petitjean.

In this paper, we present the first exact, robust, efficient and usable C++ implementation of an algorithm for parameterizing the intersection of two arbitrary quadrics, given in implicit form, with integer coefficients. This implementation is based on the parameterization method described in [21]. It has the following features:

- it computes an exact parameterization of the intersection of two quadrics with integer coefficients of arbitrary size;
- it correctly identifies, separates and parameterizes all the connected components of the intersection and gives all the relevant topological information;
- the parameterization is rational when one exists; otherwise the intersection is a smooth quartic and the parameterization involves the square root of a polynomial;
- the parameterization is either optimal in the degree of the extension of \( \mathbb{Z} \) on which its coefficients are defined or, in a small number of well-identified cases, involves one extra possibly unnecessary square root;
- it is fast and efficient and can routinely compute parameterizations of the intersection of quadrics with input coefficients having ten digits in less than 50 milliseconds on a mainstream PC.

In addition to being optimal from the point of view of the functions involved and near-optimal from the point of view of the extension on which the coefficients of the parameterizations are defined, we show how our implementation was carefully designed to minimize the size of the coefficients. In particular, we illustrate the fact that using whatever structural and geometric information is available to drive the parameterization process can have a dramatic impact on the bit complexity of the result.

We also give various experimental results illustrating the efficiency of the implemented algorithms. For instance, we show that in case the intersection is a smooth quartic (the generic and computationally most demanding case) our implementation computes a parameterization of the intersection in at most 50 milliseconds when the input quadrics have coefficients with 10 digits. We also show, both in theory and in practice, that degenerate intersections, for which a lot of additional geometric information can be obtained \textit{a priori}, are parameterized a lot faster on average. On the chess set example, roughly 1,000 intersections of quadrics with coefficients with between 2 and 7 (a few with 15) digits are computed in 3.4 milliseconds on average.