Computational Geometry 54 (2016) 1–16

Contents lists available at ScienceDirect

Computational Geometry: Theory and Applications

www.elsevier.com/locate/comgeo

Faster geometric algorithms via dynamic determinant computation ☆

Vissarion Fisikopoulos [a,*], Luis Peñaranda [b,*]

[a] Université Libre de Bruxelles, CP 216, Boulevard du Triomphe, 1050 Brussels, Belgium
[b] Universidade Federal do Rio de Janeiro, Departamento de Ciência da Computação, Av. Athos de Silveira Ramos 274, Rio de Janeiro 21941-916, Brazil

ARTICLE INFO

Article history:
Received 17 December 2014
Received in revised form 16 July 2015
Accepted 2 December 2015
Available online 14 December 2015

Keywords:
Determinant algorithms
Orientation predicate
Volume computation
Rank-1 updates
Experimental analysis

ABSTRACT

The computation of determinants or their signs is the core procedure in many important geometric algorithms, such as convex hull, volume and point location. As the dimension of the computation space grows, a higher percentage of the total computation time is consumed by these computations. In this paper we study the sequences of determinants that appear in geometric algorithms. The computation of a single determinant is accelerated by using the information from the previous computations in that sequence.

We propose two dynamic determinant algorithms with quadratic arithmetic complexity when employed in convex hull and volume computations, and with linear arithmetic complexity when used in point location problems. We implement the proposed algorithms and perform an extensive experimental analysis. On one hand, our analysis serves as a performance study of state-of-the-art determinant algorithms and implementations. On the other hand, we demonstrate the supremacy of our methods over state-of-the-art implementations of determinant and geometric algorithms. Our experimental results include a 20 and 78 times speed-up in volume and point location computations in dimension 6 and 11 respectively.

1. Introduction

Computing the sign of a determinant, or in other words evaluating a determinant *predicate*, is in the core of many important geometric algorithms. For example, convex hull algorithms use *orientation* predicates, and Delaunay triangulation algorithms involve *in-sphere* predicates. Furthermore, the computation of the value of a determinant, or in other words a determinant *construction*, is also important in some geometric algorithms. For example, the exact volume computation of a convex polytope using either triangulation or sign decomposition method relies on the computation of the volume of simplices, which reduces to computing the value of a determinant.

In other words predicates encapsulate decisions in contrast to constructions that involve computation of new numerical values. In general dimension d, the orientation predicate of $d+1$ points is the sign of the determinant of a matrix containing

☆ This work is partially supported by project "Computational Geometric Learning", which acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number: 255827. The main part of this work was done while the authors were at the University of Athens.

* Both authors are corresponding authors.
E-mail addresses: vfisikop@ulb.ac.be (V. Fisikopoulos), luisp@ufrj.br (L. Peñaranda).

http://dx.doi.org/10.1016/j.comgeo.2015.12.001
0925-7721/

the homogeneous coordinates of the points as columns. On the other hand, the volume of a simplex is the value of the determinant of a matrix containing the homogeneous coordinates of the $d+1$ vertices of the simplex. In practice, as the dimension grows, a higher percentage of the computation time is consumed by these core procedures.

In this paper, we study effective algorithms and implementations for the computation of the determinant predicates and constructions that appear in geometric computations. The model we follow is the exact computation paradigm presented in [1] and advocated by the Computation Geometry Algorithms Library (`CGAL`) [2], a state-of-the-art library for geometric computations. Note that in geometric algorithms the naive use of floating point arithmetic may lead to incorrect results [3]. There are two main scenarios regarding exactness. The first provides exact predicates but not necessarily exact constructions while the second provides both exact predicates and exact constructions. In this paper we study the second scenario. We give a particular emphasis on exact division and division-free algorithms. Avoiding divisions is crucial when working on a ring that is not a field, *e.g.*, integers or polynomials.

The main idea of our approach is to study the *sequence* of computations of determinants or signs of determinants that appear in geometric algorithms. A single computation can be accelerated by using the information from the previous computations in this sequence. The essential case is the sequence of computations of the orientation predicates that appear in convex hull algorithms. The convex hull problem is probably the most fundamental problem in discrete and computational geometry. In fact, the problems of regular, Delaunay triangulations and Voronoi diagrams reduce to it by computing a convex hull in one dimension higher [4]. Additionally, in the course of an incremental convex hull algorithm like Beneath-and-Beyond [5] we compute the volume of the polytope as a by-product of the computation. See [6] for a survey on volume computation and relevant implementations.

Since we will study *in practice* the performance of geometric and algebraic algorithms, it is important to classify the test cases. Especially, one of the parameters we will use is the dimension. We will refer throughout the paper to dimensions $d < 5$ as *low*, to dimensions $5 \leq d \leq 25$ as *medium* and to dimensions $d > 25$ as *high*, unless otherwise stated. In our experiments, we focus on medium dimensions for determinant computations and "small to medium" for geometric algorithms, *i.e.*, 6 to 11 depending on the application.

1.1. Previous work

There is a variety of algorithms and implementations for computing the determinant of a $d \times d$ matrix. Let us denote by $O(d^\omega)$ the complexity of matrix multiplication. First, we consider the case where the matrix has values from a field. For $\omega > 2$, an algorithm for matrix multiplication imply an algorithm for determinant computation with the same ω [7]. The best current ω is 2.3728639 [8].

An important class of determinant computation algorithms are the algorithms which use *exact divisions*, *i.e.*, divisions known to have remainder zero. An application of them is the computation of the determinant of a matrix with integer entries using only integer arithmetic. A typical example of this is Bareiss algorithm [9].

Division-free algorithms form another category. They use no divisions at all, *e.g.*, when matrix coefficients are elements of an abstract commutative ring. The best current ω in this category is 2.697263 [10]. Here, it is worth mentioning a family of determinant algorithms that use combinatorial approaches. They were introduced by Mahajan and Vinay [11], and are based on *clow* (closed ordered walk) sequences. Several similar methods with complexity $O(d^4)$ are surveyed by Rote [12]. Based on the idea of clow sequences Bird introduced a simpler algorithm that uses matrix operations [13]. Its complexity is $O(dM(d))$, where $M(d)$ is the complexity of matrix multiplication. Urbańska conceived a method that uses fast matrix multiplication [14] to obtain a complexity $O(d^{3.03})$ [15]. However, in practice when d is small, Bird's algorithm behaves better than other division-free algorithms, as it will be discussed in Section 4.3.

Determinants of matrices over a ring arise in combinatorial problems [16], in algorithms for lattice polyhedra [17] and secondary polytopes [18] or in computational algebraic geometry problems [19]. A special case of the latter is the computation of resultant polytopes that have applications in polynomial system solving [20] and geometric modeling [21].

Good asymptotic complexity does not imply good behavior in practice for low and medium dimensions. For instance, `LinBox` [22], which implements algorithms with state-of-the-art asymptotic complexity, introduces a significant overhead in low and medium dimensions, and seems most suitable in high dimensions (see Section 4.3 for more details).

`Eigen` [23] implements LU decomposition, of complexity $O(d^3)$, and seems to be suitable for low and medium dimensions. `Eigen` was designed with floating-point computations in mind, where it uses hardware floating-point vectors to attain great speed.

In addition, there exists a variety of algorithms for determinant sign computation [24–28]. Kaltofen and Villard [29] present a complete survey on the matter. One tool commonly used for sign computations is *filtering*: arithmetic operations are done using fixed-precision floating-point interval arithmetic, switching to exact arithmetic only when the sign is unknown. Filtered computations are widely used because they provide a simple approach to avoid performing exact operations in many cases. While filtered computation performs well in low dimensions, there is no experimental study on the efficiency of current methods in medium dimensions (see Section 4.6).

The problem of computing sequences of determinants has also been studied. TOPCOM [18] is the reference software for enumerating all regular triangulations of a set of points in general dimension. It efficiently pre-computes all orientation determinants that will be needed in the computation and stores their signs. Emiris et al. [30] study a similar problem in the context of computational algebraic geometry. In particular, the computation of several regular triangulations for different