Project number IST-25582

CGL

Computational Geometric Learning

Optimizing the computation of sequences of determinantal predicates

STREP

Information Society Technologies

Period covered:	November 1, 2010–October 31, 2011
Date of preparation:	October 29, 2011
Date of revision:	October 29, 2011
Start date of project:	November 1, 2010
Duration:	3 years
Project coordinator name:	Joachim Giesen (FSU)
Project coordinator organisation:	Friedrich-Schiller-Universität Jena Jena, Germany

Optimizing the computation of sequences of determinantal predicates

Ioannis Z. Emiris* Vissarion Fisikopoulos* Luis Peñaranda*

October 29, 2011

Abstract

The Orientation predicate is the bottleneck of some important geometric algorithms, such as convex hull and triangulation computations, since it is computed repeatedly. In these and other applications, the matrices whose determinants are computed along the execution of an algorithm have many columns in common. Besides Orientation, this concerns all determinantal predicates such as InSphere, or determinantal primitives, such as volume computation. We exploit this structure, and introduce a method that hashes intermediate computations to reduce the time of subsequent determinant computations. Our technique is also applicable in other settings where, e.g., the matrices differ only in a few rows. We implemented our technique in the context of CGAL triangulations, and obtained a speedup of about 2 in computing silhouettes of the secondary polytope.

Keywords: Orientation predicate, determinant, convex hull, triangulation, CGAL implementation.

1 Introduction

The Orientation predicate is the bottleneck of some important geometric algorithms, such as convex hull and triangulation computations. In general dimension d, the Orientation of $d+1$ points is computed as the determinant of a matrix containing the coordinates of the points as columns, plus one last row full of ones. In certain applications, the matrices whose determinants are to be computed along the execution of an algorithm have many columns in common. A typical example of this is the resultant polytope algorithm in [EFK11]. In it, many convex hulls of sets of similar points need to be computed.

We exploit in this paper this particular structure of the matrices, and introduce a method that hashes the determinants of the minors of the matrices computed so far. We show that this caching reduces the time of subsequent determinant computations, and the overall execution time of the algorithms. This technique is also applicable in more generic settings, for instance, in the case when the matrices whose determinants are to be computed differ only in one row, such as convex hulls of lifted points. It can be extended to the case of a few different rows, as well as to any other determinantal predicate or primitives, such as InCircle/InSphere and volume calculations.

The report is organized as follows. Section 2 describes our approach. Section 3 gives some details on the CGAL implementation. Section 4 shows some experimental evidence, obtained by running a version of a geometric algorithm using the hashed determinants scheme and a version of the same algorithms that use CGAL's LU decomposition algorithm to compute determinants. We conclude the article with some future work.

*National and Kapodistrian University of Athens, Department of Informatics and Telecommunications, Athens, Greece. {emiris,vissarion,lpenaranda}@di.uoa.gr

1.1 Previous Work.

TOPCOM [Ram02] implements efficiently the computation of secondary polytopes. It is currently the reference software for computing regular or all triangulations of a set of points. It computes all Orientation determinants that will be needed and stores their signs, i.e. computes the pointset's chirotope. It employs smart strategies for computing many related determinants when a few different columns are added to a fixed submatrix.

Our focus is on determinantal predicates; below, we choose the Laplace expansion to evaluate determinants. We considered computing determinants with several methods. Algorithms with good asymptotic complexity [BH74, KV05] do not behave well in practice for small and medium dimensions. Decomposition methods (LU, Cholesky, QR) have complexity $O(n^3)$, but they are slower in practice than Laplace expansion in small dimensions.

In addition, we considered linear algebra libraries LinBox [DGG$^+$02] and Eigen [1] in different geometric scenarios. LinBox seems most suitable in very high dimensions (typically > 100), it implements algorithms with state-of-the-art asymptotic complexity, and it introduces a significant overhead in medium dimensions, which is our focus. Eigen seems to be suitable for medium to high dimensions, whereas CGAL [CGA] provides efficient determinant computation in low to medium dimensions. Hence, we restrict attention on the latter.

Determinant sign computation has attracted a lot of attention. There have been techniques optimizing bit complexity, by focusing on the arithmetic issues. Record results were obtained, at the time of their publication, by methods based on Gram-Schmidt decomposition [Cla92], on Chinese remaindering using certified floating-point computation [BEPP99], as well as on Hensel lifting [ABM99] in very large dimension.

2 Hashing of Determinants

This section discusses the method to achieve output-sensitivity and avoid duplication of computation in algorithms that imply computing sequences of determinants. Examples of such algorithms are convex hull and triangulation computations. These two algorithms rely on the sign of Orientation. The Orientation predicate between $d+1$ points in dimension d consists of constructing a $(d+1) \times (d+1)$ matrix, on which each column contains the d coordinates of each point, plus a one on the last place, and compute the sign of its determinant.

Our main motivation is the algorithm in [EFK11]. Given a system of $n+1$ polynomials in n variables, this algorithm computes the Newton polytope (or a projection of it) of the resultant of this system. For this, it defines a new pointset $\mathcal{A}$ in dimension $2n$. Then, it computes many regular triangulations in dimension $2n+1$, where the points of each triangulation are liftings of the original $2n$-dimensional points. Thus, it needs to compute determinants of matrices of size $2n+2$ which, when n increases, typically dominate the whole computation. Many of the matrices whose determinants are to be computed will be similar, because the points to be triangulated differ only in the lift coordinate. This situation is common in many geometric algorithms. We propose an efficient method to enumerate many regular triangulations with similar characteristics to take advantage of this.

The Laplace expansion of the determinant. Consider the $2n \times |\mathcal{A}|$ matrix with the points of $\mathcal{A}$ as columns. Define P as the extension of this matrix by adding lifting values $\widehat{w}$ as the last row. We use the Laplace (or cofactor) expansion along the last row for computing the determinant of the square submatrix formed by any $2n+1$ columns of P; wlog these are the first $2n+1$ columns $a_1, \ldots, a_{2n+1}$. Let $\langle a_1, \ldots, a_{2n+1} \rangle \setminus i$ be the vector $\langle a_1, \ldots, a_{2n+1} \rangle$ without its i-th element; $P_{\langle a_1, \ldots, a_{2n+1} \rangle \setminus i}$ is the $(2n) \times (2n)$ matrix obtained from the $2n$ first elements of

[1] http://eigen.tuxfamily.org/

the columns whose indices are in $\langle a_1, \ldots, a_{2n+1}\rangle \setminus i$, and $P_{i,2n+1}$ is the $(2n+1)$-th element of column a_i. If $|M|$ denotes the determinant of matrix M, then

$$|P_{\langle a_1,\ldots,a_{2n+1}\rangle}| = \sum_{i=1}^{2n+1} (-1)^{i+2n+1} P_{i,2n+1} |P_{\langle a_1,\ldots,a_{2n+1}\rangle \setminus i}|. \tag{1}$$

When $2n+1 = 1$, the determinant $|P_{\langle a\rangle}|$ is the first element of column a.

Orientation and Volume. We now state the two main predicates of [EFK11], which motivated our work. Volume is the primitive expressed by the determinant

$$|P^{hom}_{\langle a_1,\ldots,a_{2n+1}\rangle}|$$

of the matrix constructed by the columns $a_1, \ldots, a_{2n+1}$ of P when we replace its last row by $\vec{1} \in \mathbb{R}^{2n+1}$. Orientation is the predicate expressed by the sign of determinant

$$|P^{hom}_{\langle a_1,\ldots,a_{2n+2}\rangle}|$$

of the matrix constructed by columns $a_1, \ldots, a_{2n+2}$ of P when we add $\vec{1} \in \mathbb{R}^{2n+2}$ as last row. Following the Laplace expansion, these two operations reduce to computing $2n \times 2n$ minors of P. Our technique consists in maintaining a hash table with the computed minors, which will be reused at subsequent steps of the algorithm. We store all minors of sizes between 2 and $2n$. All of them can be used for later computations, given the structure of the determinants that are computed in our problem. First, recall from Alg. 3 in Sect. 3 of [EFK11] that, wlog, all, except the first m, coordinates of $\widehat{w}$ are zero. Thus, the only change in P when we change $\widehat{w}$ are the first m coordinates of its last row. This means that several minors appearing in these computations are repeated many times. Additionally, the last row of P has at most m nonzero elements; this results in avoiding many determinant computations when we expand along the last row.

Example 1. Consider the following modification of the polynomials of Example 5 of [EFK11], $f_0 := c_{00} - c_{01}x_1x_2 + c_{02}x_2$, $f_1 := c_{10} - c_{11}x_1x_2^2 + c_{12}x_2^2$, $f_2 := c_{20} - c_{21}x_1^2 + c_{22}x_2$ resulting to the following matrix

$$P = \left|\begin{array}{ccccccccc} 0 & 0 & 0 & 1 & 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 0 & 1 & 2 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ w_1 & w_2 & w_3 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right| \begin{array}{l} \} \text{ support coordinates} \\ \} \text{ Cayley trick coordinates} \\ \} \ \widehat{w} \end{array}$$

given a lifting vector $\widehat{w}$. We reduce the computation of determinants in computations of minors of the matrix defined by P if we delete the last row. Computing an Orientation predicate using Laplace expansion consist of computing $\binom{6}{4} = 15$ minors. On the other hand, if we compute $|P^{hom}_{\langle 1,2,3,4,5,6\rangle}|$ then the computation of $|P^{hom}_{\langle 1,2,3,4,5,7\rangle}|$ need the computation of only $\binom{6}{4} - \binom{5}{4} = 10$ new minors. More interestingly, by giving a new lifting vector $\widehat{w'}$ we can compute $|P'^{\,hom}_{\langle 1,2,3,4,5,6\rangle}|$ without computing any new minors.

The method. Given a lifting vector w, Alg. 3 of [EFK11] constructs a regular subdivision S of $\mathcal{A}$ as follows: it first computes the convex hull of the lifted $\mathcal{A}$ according to $\widehat{w}$, then projects the upper facets to $\mathbb{R}^{2n}$.

More specifically, the algorithm initially computes the convex hull of the points of the last $|\mathcal{A}| - m$ columns, i.e., the points correspond to 0 coordinates of $\widehat{w}$. Alg. 4 of [EFK11] updates P

with the new $\widehat{w}$ and inserts the points of the first m columns in the convex hull of $|\mathcal{A}|-m$ points in every step. To perform a point location it needs to compute the Orientation predicate of the point to be inserted with $2n+1$ points on the convex hull. That is, $|P^{hom}_{\langle a_1,\ldots,a_{2n+2}\rangle}|$ where a_i's are the indices of the points involved in this predicate. If we expand along the last row we have to compute the determinants $|P_{\langle a_1,\ldots,a_{2n+2}\rangle\setminus i}|$ for every $i \in \{1,\ldots,2n+2\}$. Using equation (1) we reduce the computation of the Orientation predicate to the computation of $O(n^2)$ determinants of size $2n$. In fact, we perform $O(n^2)$ hash table lookups (which can be considered constant both in theory and practice with the right hashing function), and thus compute only the determinants that are not computed in a previous step.

Step 3 of [EFK11, alg.3] computes volumes of triangles. This uses the projection that omits the last coordinate of the upper facets of the convex hull computed above. In fact, it suffices to compute the Volume primitive $|P^{hom}_{\langle a_1,\ldots,a_{2n+1}\rangle}|$ for every facet such that $a_1,\ldots,a_{2n+1}$ are the indices of the facet's points in P. Its sign will determine if it is upper or not, and its value is the volume we need in [EFK11, alg.3]. Similarly as above, using equation (1), we reduce the computation of this predicate to the computation of $O(n)$ determinants of size $2n$.

Lemma 1. *Using hashing of determinants, the complexity of Orientation and Volume is $O(n^2)$ and $O(n)$, respectively, if all minors have already been computed.*

This is better than the best known bit complexity bound of determinant computation, namely $O(n^{2.697263})$ [KV05]. To the other extreme, when none of the minors is precomputed, the complexity becomes $O(n!)$ with Laplace expansion. Experiments show that the algorithm of [EFK11] with hashed determinants largely outperforms the version not using the hash.

The drawback of this method is that the amount of storage needed. When all the determinants are computed it is $O(n!)$, which equals the complexity of the Laplace expansion algorithm. Nevertheless, the hash table can be cleared at any moment to limit memory consumption, at the cost of stopping benefit from precomputed determinants. This means that a trade-off between memory consumption and efficiency can be obtained with our method.

3 Implementation

We implemented our hashed determinants scheme in the context of our implementation of the algorithms of [EFK11]. We modified the CGAL [CGA] experimental package `Triangulation` for triangulations in general dimensional spaces in order to benefit from our hashed determinants scheme. `Triangulation` was shown to be the fastest in up to 6 dimensions [BDH09]. It provides the same functionality as CGAL lower-dimensional triangulation packages, with many algorithmic improvements and a new efficient triangulation data structure. The complete code can be obtained from `http://respol.sourceforge.net`.

The implementation was done in two parts. One is independent of CGAL, which maintains the hash table. The other part consists in the modification of CGAL code in order to add each newly created point to the hash table.

The hash table was implemented as a `boost::unordered_map<Index,NT>`. The `Index`, a typedef to `std::vector<size_t>`, is used to reference the determinant which wants to be computed. Since each point is assigned a column in the table, a index vector will represent a matrix formed by the referenced columns. The field number type `NT` is the number type used to define the coordinates of points, and thus, the field where determinant computations must be carried out. In our case, `NT` is `CGAL::Gmpq`, the CGAL's representation of GMP rational numbers. All these are encapsulated in a template class `FastHashedDeterminant<NT>`. This class provides the methods for inserting points in the hash, and each point is given an index. When wanting to compute a determinant formed by some points in the hash, the user must call the following function.

```
NT FastHashedDeterminant<NT>::determinant(const Index &idx);
```

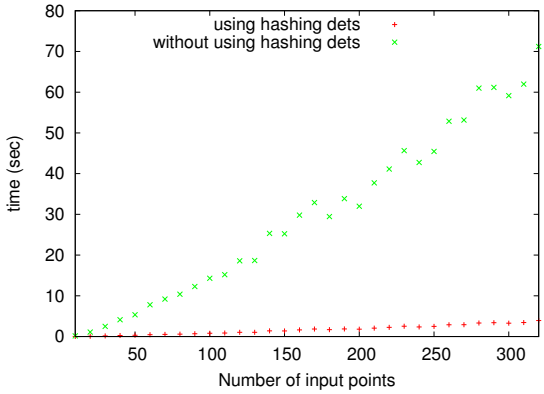

Figure 1: Performance comparisons for hashing versus non hashing determinants when $n = 2$.

This function will take care of computing the needed determinants of minors and will transparently return the result. If what is wanted is to compute an Orientation predicate, one must call the following function.

```
NT FastHashedDeterminant<NT>::homogeneous_determinant(const Index &idx);
```

This function will compute the determinant of the matrix formed by the points whose indices are in `idx` and add a row of ones in the bottom. Finally, to compute the determinant of a matrix of lifted points, given that the non-lifted points were already inserted in the hash, the function to call is the following.

```
NT FastHashedDeterminant<NT>::homogeneous_determinant(const Index &idx
                                                      const std::vector<NT> &r);
```

Here, `r` is the row of the matrix containing the liftings.

We also performed some modifications to CGAL. We modified the d-dimensional kernel in order to add to our hash table each point that is created. And we also replaced the orientation predicate, in order to call our determinant functions.

4 Experiments

We use `Triangulation` to compute regular triangulations in $2n + 1$ dimensions, as well as to maintain the projection of the resultant polytope. We examine the effect of the hashed determinants method to the execution time of the algorithms of [EFK11]. Fig. 1 shows an impressive gain when $n = 2$. For $n > 2$, the hashed determinants method allows the algorithm to compute instances of the problem that would be intractable otherwise. The tests were ran on an Intel Core i5-2400 3.1GHz, with 6MB L2 cache and 8GB RAM running 64-bit Debian GNU/Linux.

We also apply the method of hashing determinants as described in Sect. 2 in the computation of a placing triangulation of a point set $\mathcal{A}$. Note that this computation will also compute the convex hull of $\mathcal{A}$. We compare the efficiency of the CGAL `Triangulation` package with and without using the hashing determinants method. Fig. 2 (right) shows some experimental results of this comparison. The inputs are random points on a hypercube; for simplicity, for a fixed number of input points their coordinates are not identical for the two implementations (hash/no-hash).

Our method increases the efficiency of the implementation when the input points are up to dimension 5. For larger dimensions it results in a poorer speedup. Also note that it uses more memory than the original package. In these experiments, it runs out of memory in when the input is 300 points in dimension 7.

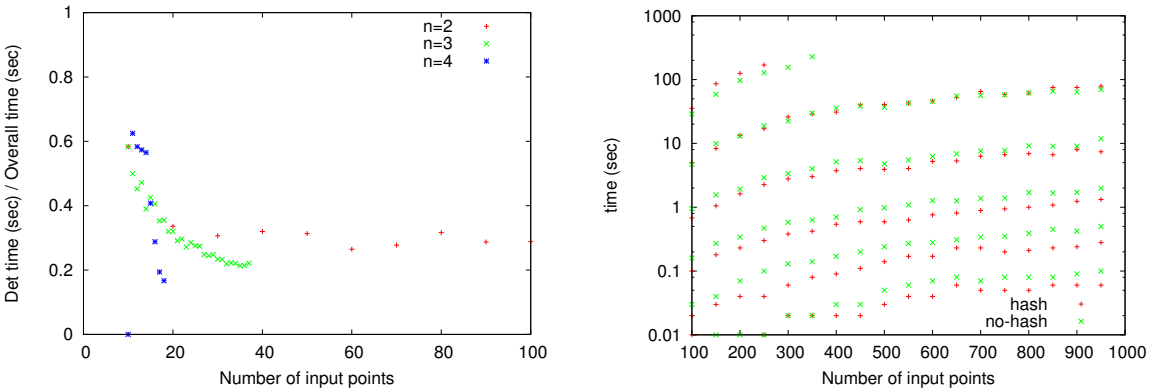

Figure 2: The percentage of computation time of Alg. 1 spent on computation of determinants (left). The time for computing a triangulation of a point set $\mathcal{A}$ as function of $|\mathcal{A}|$. The graphs correspond to different ambient dimensions of the input points i.e. from bottom to top $dim(\mathcal{A}) = 2, \ldots, 7$ (right).

5 Future work

This method can be used in a general manner, not only inside the CGAL package. The next step is, thus, to develop an interface that permits using the hashed determinants technique transparently (for instance, without knowledge of points' indices) and submit as a package to CGAL.

Another interesting aspect on the implementation would be to take advantage of multi-threading. Since matrices are not changed when computing determinants with our method, it should be possible to compute the determinants of the minors in parallel, before the final computation.

Acknowledgment Authors enjoy partial support from project "Computational Geometry 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.

References

[ABM99] J. Abbott, M. Bronstein, and T. Mulders. Fast deterministic computation of determinants of dense matrices. In *Proc. ACM Intern. Symp. on Symbolic and Algebraic Computation*, pages 197–203, 1999.

[BDH09] J.-D. Boissonnat, O. Devillers, and S. Hornus. Incremental construction of the Delaunay triangulation and the Delaunay graph in medium dimension. In *SoCG*, pages 208–216, 2009.

[BEPP99] H. Brönnimann, I.Z. Emiris, V. Pan, and S. Pion. Sign determination in Residue Number Systems. *Theor. Comp. Science, Spec. Issue on Real Numbers & Computers*, 210(1):173–197, 1999.

[BH74] J.R. Bunch and J.E. Hopcroft. Triangular factorization and inversion by fast matrix multiplication. *Math. Comp.*, 28:231–236, 1974.

[CGA] CGAL: Computational geometry algorithms library. http://www.cgal.org.

[Cla92] K.L. Clarkson. Safe and effective determinant evaluation. In *Proc. Annual IEEE Symp. Found. Comp. Sci.*, pages 387–395, 1992.

[DGG+02] J.-G. Dumas, T. Gautier, M. Giesbrecht, P. Giorgi, B. Hovinen, E. Kaltofen, B. D. Saunders, W. J. Turner, and G. Villard. Linbox: A generic library for exact linear algebra. In *Proc. 1st Internat. Congress Math. Software (ICMS)*, pages 40–50, Beijing, China, 2002.

[EFK11] I.Z. Emiris, V. Fisikopoulos, and C. Konaxis. An output-sensitive algorithm for computing projections of resultant polytopes. Technical Report CGL-TR-08, NKUA, 2011.

[KV05] E. Kaltofen and G. Villard. On the complexity of computing determinants. *Computational Complexity*, 13:91–130, 2005.

[Ram02] J. Rambau. TOPCOM: Triangulations of point configurations and oriented matroids. In Arjeh M. Cohen, Xiao-Shan Gao, and Nobuki Takayama, editors, *Math. Software: ICMS*, pages 330–340. World Scientific, 2002.