Faster Geometric Algorithms via Dynamic Determinant Computation

Vissarion Fisikopoulos

joint work with L. Peñaranda (now IMPA, Rio de Janeiro)

Department of Informatics, University of Athens, Greece

ESA, Ljubljana, 14.Sep.2012

Geometric algorithms and predicates

Setting

- geometric algorithms $\longrightarrow$ sequence of geometric predicates
- many geometric predicates $\longrightarrow$ determinants
- Hi-dim: as dimension grows predicates become more expensive

Geometric algorithms and predicates

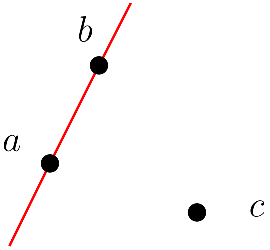
Setting

- geometric algorithms $\longrightarrow$ sequence of geometric predicates
- many geometric predicates $\longrightarrow$ determinants
- Hi-dim: as dimension grows predicates become more expensive

Examples

- Orientation: Does c lie on, left or right of ab?

$$\begin{vmatrix} a_x & a_y & 1 \\ b_x & b_y & 1 \\ c_x & c_y & 1 \end{vmatrix} \gtreqless 0$$



Geometric algorithms and predicates

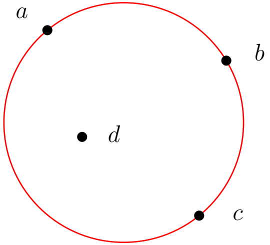
Setting

- geometric algorithms $\longrightarrow$ sequence of geometric predicates
- many geometric predicates $\longrightarrow$ determinants
- Hi-dim: as dimension grows predicates become more expensive

Examples

- inCircle: Does d lie on, inside or outside of abc?

$$\begin{vmatrix} a_x & a_y & a_x^2 + a_y^2 & 1 \\ b_x & b_y & b_x^2 + b_y^2 & 1 \\ c_x & c_y & c_x^2 + c_y^2 & 1 \\ d_x & d_y & d_x^2 + d_y^2 & 1 \end{vmatrix} \gtreqless 0$$



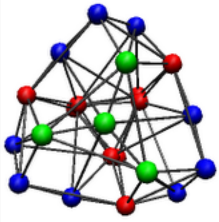
Outline

1. Motivation and existing work
2. Dynamic determinant computation
3. Geometric algorithms: convex hull
4. Implementation - experiments

Outline

Motivation

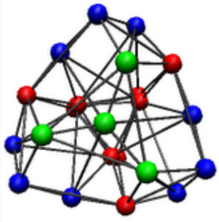
- Algorithms for resultant polytopes [Emiris,F,Konaxis,Peñaranda SoCG'12] [YuJensen'12] (discriminant polytopes [Emiris, F, Dickenstein])
  `Respol`: compute resultant, discriminant polytopes up to dim. 6



4-d resultant polytope

Motivation

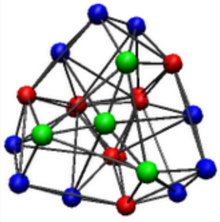
- Algorithms for resultant polytopes [Emiris,F,Konaxis,Peñaranda SoCG'12] [YuJensen'12] (discriminant polytopes [Emiris, F, Dickenstein])
  `Respol`: compute resultant, discriminant polytopes up to dim. 6
- Counting lattice points in polyhedra [Barvinock'99] [DeLoera et.al'04], Integer convex optimization [Grötschel,Lovász,Schrijver'88]



4-d resultant polytope

Motivation

- Algorithms for resultant polytopes [Emiris,F,Konaxis,Peñaranda SoCG'12] [YuJensen'12] (discriminant polytopes [Emiris, F, Dickenstein]) `Respol`: compute resultant, discriminant polytopes up to dim. 6
- Counting lattice points in polyhedra [Barvinock'99] [DeLoera et.al'04], Integer convex optimization [Grötschel,Lovász,Schrijver'88]



4-d resultant polytope

Focusing on ...

- Hi-dim CompGeom ($3 < d < 10$)
- Computation over the integers

Existing work

Determinant: exact computation

Given matrix $A \subseteq \mathbb{R}^{d \times d}$

- Theory: State-of-the-art $O(d^{\omega})$, $\omega \sim 2.37$ [CoppersmithWinograd90]
- Practice: Gaussian elimination, $O(d^3)$

Existing work

Determinant: exact computation

Given matrix $A \subseteq \mathbb{R}^{d \times d}$

- Theory: State-of-the-art $O(d^{\omega})$, $\omega \sim 2.37$ [CoppersmithWinograd90]
- Practice: Gaussian elimination, $O(d^3)$

Division-free determinant algorithms

- Laplace (cofactor) expansion, $O(d!)$
- [Rote01], $O(d^4)$
- [Bird11], $O(d^{\omega+1})$

Existing work

Determinant: exact computation

Given matrix $A \subseteq \mathbb{R}^{d \times d}$

- Theory: State-of-the-art $O(d^{\omega})$, $\omega \sim 2.37$ [CoppersmithWinograd90]
- Practice: Gaussian elimination, $O(d^3)$

Division-free determinant algorithms

- Laplace (cofactor) expansion, $O(d!)$
- [Rote01], $O(d^4)$
- [Bird11], $O(d^{\omega+1})$

Dynamic determinant computation

Dynamic transitive closure in graphs [Sankowski FOCS'04]

Outline

Dynamic Determinant Computations

One-column update problem

Given matrix $A \subseteq \mathbb{R}^{d \times d}$, answer queries for $\det(A)$ when i-th column of A, $(A)_i$, is replaced by $u \subseteq \mathbb{R}^d$.

Dynamic Determinant Computations

One-column update problem

Given matrix $A \subseteq \mathbb{R}^{d \times d}$, answer queries for $\det(A)$ when i-th column of A, $(A)_i$, is replaced by $u \subseteq \mathbb{R}^d$.

Solution: Sherman-Morrison formula (1950)

$$A'^{-1} = A^{-1} - \frac{(A^{-1}(u - (A)_i))\ (e_i^T A^{-1})}{1 + e_i^T A^{-1}(u - (A)_i)}$$

$$\det(A') = (1 + e_i^T A^{-1}(u - (A)_i)\det(A)$$

- Only vector$\times$vector, vector$\times$matrix $\longrightarrow$ Complexity: $O(d^2)$
- Algorithm: dyn_inv

Dynamic Determinant Computations

One-column update problem

Given matrix $A \subseteq \mathbb{R}^{d\times d}$, answer queries for $\det(A)$ when i-th column of A, $(A)_i$, is replaced by $u \subseteq \mathbb{R}^d$.

Variant Sherman-Morrison formulas

- Q: What happens if we work over the integers?

Dynamic Determinant Computations

One-column update problem

Given matrix $A \subseteq \mathbb{R}^{d \times d}$, answer queries for $\det(A)$ when i-th column of A, $(A)_i$, is replaced by $u \subseteq \mathbb{R}^d$.

Variant Sherman-Morrison formulas

- Q: What happens if we work over the integers?
- A: Use the adjoint of A (A^{adj}) $\rightarrow$ exact divisions

Dynamic Determinant Computations

One-column update problem

Given matrix $A \subseteq \mathbb{R}^{d \times d}$, answer queries for $\det(A)$ when i-th column of A, $(A)_i$, is replaced by $u \subseteq \mathbb{R}^d$.

Variant Sherman-Morrison formulas

- Q: What happens if we work over the integers?
- A: Use the adjoint of A (A^{adj}) $\rightarrow$ exact divisions
- Complexity: $O(d^2)$ Algorithm: dyn_adj

Outline

The convex hull problem

Definition

The *convex hull* of $\mathcal{A} \subseteq \mathbb{R}^d$ is the smallest convex set containing $\mathcal{A}$.

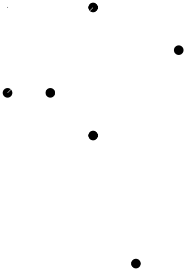
Example

The convex hull problem

Definition

The *convex hull* of $\mathcal{A} \subseteq \mathbb{R}^{\mathrm{d}}$ is the smallest convex set containing $\mathcal{A}$.

Incremental convex hull - Beneath-and-Beyond



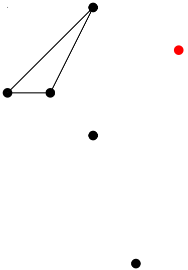
- connect each outer point to the visible segments
- visibility test = orientation test

The convex hull problem

Definition

The *convex hull* of $\mathcal{A} \subseteq \mathbb{R}^d$ is the smallest convex set containing $\mathcal{A}$.

Incremental convex hull - Beneath-and-Beyond



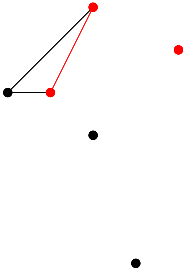
- connect each outer point to the visible segments
- visibility test = orientation test

The convex hull problem

Definition

The *convex hull* of $\mathcal{A} \subseteq \mathbb{R}^d$ is the smallest convex set containing $\mathcal{A}$.

Incremental convex hull - Beneath-and-Beyond



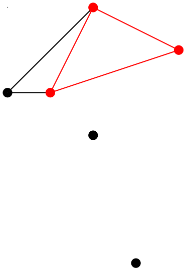
- connect each outer point to the visible segments
- visibility test = orientation test

The convex hull problem

Definition

The *convex hull* of $\mathcal{A} \subseteq \mathbb{R}^{\mathrm{d}}$ is the smallest convex set containing $\mathcal{A}$.

Incremental convex hull - Beneath-and-Beyond



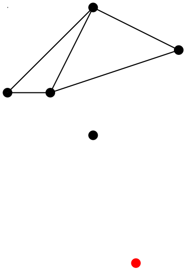
- connect each outer point to the visible segments
- visibility test = orientation test

The convex hull problem

Definition

The *convex hull* of $\mathcal{A} \subseteq \mathbb{R}^{\mathrm{d}}$ is the smallest convex set containing $\mathcal{A}$.

Incremental convex hull - Beneath-and-Beyond



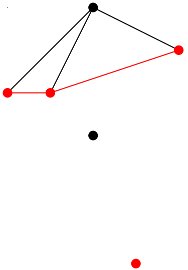
- connect each outer point to the visible segments
- visibility test = orientation test

The convex hull problem

Definition

The *convex hull* of $\mathcal{A} \subseteq \mathbb{R}^d$ is the smallest convex set containing $\mathcal{A}$.

Incremental convex hull - Beneath-and-Beyond



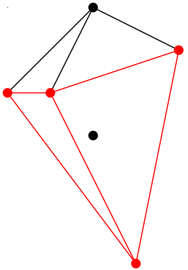
- connect each outer point to the visible segments
- visibility test = orientation test

The convex hull problem

Definition

The *convex hull* of $\mathcal{A} \subseteq \mathbb{R}^{\mathrm{d}}$ is the smallest convex set containing $\mathcal{A}$.

Incremental convex hull - Beneath-and-Beyond



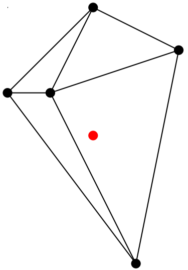
- connect each outer point to the visible segments
- visibility test $=$ orientation test

The convex hull problem

Definition

The *convex hull* of $\mathcal{A} \subseteq \mathbb{R}^d$ is the smallest convex set containing $\mathcal{A}$.

Incremental convex hull - Beneath-and-Beyond



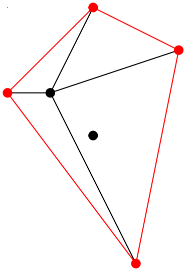
- connect each outer point to the visible segments
- visibility test = orientation test

The convex hull problem

Definition

The *convex hull* of $\mathcal{A} \subseteq \mathbb{R}^{\mathrm{d}}$ is the smallest convex set containing $\mathcal{A}$.

Incremental convex hull - Beneath-and-Beyond



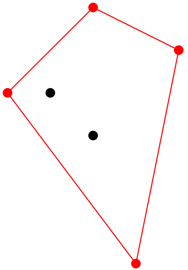
- connect each outer point to the visible segments
- visibility test $=$ orientation test

The convex hull problem

Definition

The *convex hull* of $\mathcal{A} \subseteq \mathbb{R}^d$ is the smallest convex set containing $\mathcal{A}$.

Incremental convex hull - Beneath-and-Beyond



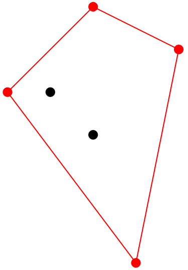
- connect each outer point to the visible segments
- visibility test = orientation test

The convex hull problem

Definition

The *convex hull* of $\mathcal{A} \subseteq \mathbb{R}^{\mathrm{d}}$ is the smallest convex set containing $\mathcal{A}$.

Incremental convex hull - Beneath-and-Beyond



- connect each outer point to the visible segments
- visibility test = orientation test

Similar problems: Delaunay, regular triangulations, point-location

The convex hull problem: REVISITED

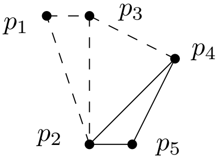
- convex hull $\longrightarrow$ sequence of similar orientation predicates
- take advantage of computations done in previous steps

The convex hull problem: REVISITED

- convex hull $\longrightarrow$ sequence of similar orientation predicates
- take advantage of computations done in previous steps

Incremental convex hull construction $\longrightarrow$ 1-column updates

$$A = \begin{array}{|c|c|c|} \hline p_2 & p_4 & p_5 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$



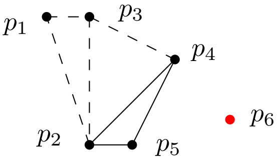
- $\text{Orientation}(p_2, p_4, p_5) = \det(A)$

The convex hull problem: REVISITED

- convex hull $\longrightarrow$ sequence of similar orientation predicates
- take advantage of computations done in previous steps

Incremental convex hull construction $\longrightarrow$ 1-column updates

$$A' = \begin{array}{|c|c|c|} \hline \boxed{p_6} & p_4 & p_5 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$



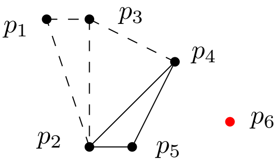
- Orientation$(p_6, p_4, p_5) = \det(A')$ in $O(d^2)$

The convex hull problem: REVISITED

- convex hull $\longrightarrow$ sequence of similar orientation predicates
- take advantage of computations done in previous steps

Incremental convex hull construction $\longrightarrow$ 1-column updates

$$A' = \begin{array}{|c|c|c|} \hline p_6 & p_4 & p_5 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$



- Orientation$(p_6, p_4, p_5) = \det(A')$ in $O(d^2)$
- Store $\det(A)$, A^{-1} + update $\det(A')$, A'^{-1} (Sherman-Morrison)

Dynamic determinants in geometric algorithms

Given $\mathcal{A} \subseteq \mathbb{R}^d$, $n = |\mathcal{A}|$ and $t =$ the number of cells

Theorem

Orientation predicates in increm. convex hull: $O(d^2)$ *(space:* $O(d^2 t)$*)*
Proof: *update det(*A'*),* A'^{-1}

Corollary

Orientation predicates involved in point-location: $O(d)$ *(space:* $O(d^2 t)$*)*
Proof: *query point never enters data-set* $\rightarrow$ *update only det(*A'*)*

Corollary

Incremental convex hull and volume comput.: $O(d^{\omega+1} n t) \rightarrow O(d^3 n t)$

Outline

Determinant implementation

Dynamic determinant computation

- C++, GNU Multiple Precision arithmetic library (GMP)
- Implement dyn_inv & dyn_adj

Exact determinant computation software

- LU decomposition (CGAL)
- optimized LU decomposition (Eigen)
- asymptotically optimal algorithms (LinBox)
- Maple's default determinant algorithm (Maple 14)
- Bird's algorithm (our implementation)
- Laplace (cofactor) expansion (our implementation)

Determinant experiments

1-column updates in $A \subseteq \mathbb{Q}^{d \times d}$ (uniform distr. rational coefficients)

d	Bird	CGAL	Eigen	Laplace	Maple	dyn_inv	dyn_adj
3	.013	.021	.014	.008	.058	.046	.023
4	.046	.050	.033	.020	.105	.108	.042
5	.122	.110	.072	.056	.288	.213	.067
6	.268	.225	.137	.141	.597	.376	.102
7	.522	.412	.243	.993	.824	.613	.148
8	.930	.710	.390	–	1.176	.920	.210
9	1.520	1.140	.630	–	1.732	1.330	.310
10	2.380	1.740	.940	–	2.380	1.830	.430

spec: Intel Core i5-2400 3.1GHz, 6MB L2 cache, 8GB RAM, 64-bit Debian GNU/Linux

Determinant experiments

1-column updates in $A \subseteq \mathbb{Q}^{d \times d}$ (uniform distr. integer coefficients)

d	Bird	CGAL	Eigen	Laplace	Linbox	Maple	dyn_inv	dyn_adj
3	.002	.021	.013	.002	.872	.045	.030	.008
4	.012	.041	.028	.005	1.010	.094	.058	.015
5	.032	.080	.048	.016	1.103	.214	.119	.023
6	.072	.155	.092	.040	1.232	.602	.197	.033
7	.138	.253	.149	.277	1.435	.716	.322	.046
8	.244	.439	.247	–	1.626	.791	.486	.068
9	.408	.689	.376	–	1.862	.906	.700	.085
10	.646	1.031	.568	–	2.160	1.014	.982	.107
11	.956	1.485	.800	–	10.127	1.113	1.291	.133
12	1.379	2.091	1.139	–	13.101	1.280	1.731	.160
13	1.957	2.779	1.485	–	–	1.399	2.078	.184

`spec: Intel Core i5-2400 3.1GHz, 6MB L2 cache, 8GB RAM, 64-bit Debian GNU/Linux`

Convex hull implementation

Hashed dynamic determinants

- dyn_inv & dyn_adj + hash table (dets & matrices)
- plug into geometric software $\longrightarrow$ geometric predicates

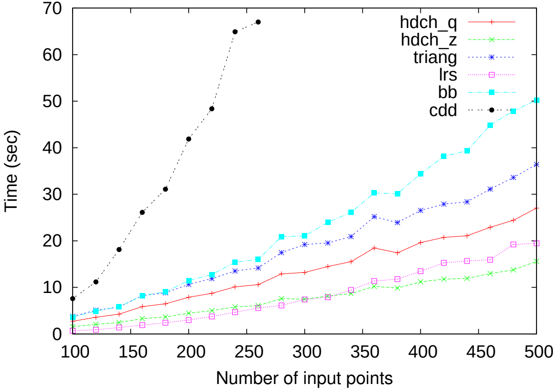
Convex hull software

- randomized incremental (triang/CGAL)
- beneath-and-beyond (bb) (polymake)
- double description method (cdd)
- gift-wrapping with reverse search (lrs)

triang/CGAL + hashed dynamic determinants = `hdch_z` or `hdch_q`

Convex hull experiments

6-dim points, integer coeffs uniformly distributed inside a 6-ball



- hdch_q (hdch_z) rational (integer) arithmetic
- drawback: memory consumption

500 6D points	
triang	0.1GB
hdch_z	2.8GB
hdch_q	3.8GB

spec: Intel Core i5-2400 3.1GHz, 6MB L2 cache, 8GB RAM, 64-bit Debian GNU/Linux

Point location experiments

- triangulation of $\mathcal{A} \subseteq \mathbb{Z}^{\mathrm{d}}$ points uniformly distibuted on a d-ball surface
- 1K and 1000K query points uniformly distibuted on a d-cube

	d	$\lvert\mathcal{A}\rvert$	preprocess time (sec)	ds mem (MB)	# cells triang	query time (sec) 1K	query time (sec) 1000K
`hdch_z`	8	120	45.20	6913	319438	0.41	392.55
`triang`	8	120	156.55	134	319438	14.42	14012.60
`hdch_z`	9	70	45.69	6826	265874	0.28	276.90
`triang`	9	70	176.62	143	265874	13.80	13520.43
`hdch_z`	10	50	43.45	6355	207190	0.27	217.45
`triang`	10	50	188.68	127	207190	14.40	14453.46
`hdch_z`	11	39	38.82	5964	148846	0.18	189.56
`triang`	11	39	181.35	122	148846	14.41	14828.67

- up to 78 times faster using up to 50 times more memory

`spec: Intel Core i5-2400 3.1GHz, 6MB L2 cache, 8GB RAM, 64-bit Debian GNU/Linux`

Conclusions and Future work

- Orientation predicates. CH: $O(d^2)$, point location: $O(d)$
- Efficient (division free) dynamic determinant implementation
- More efficient CH implementation, $78\times$ speed-up in point location
- The code: `http://hdch.sourceforge.net`

Conclusions and Future work

- Orientation predicates. CH: $O(d^2)$, point location: $O(d)$
- Efficient (division free) dynamic determinant implementation
- More efficient CH implementation, $78\times$ speed-up in point location
- The code: http://hdch.sourceforge.net

Future work

- Delaunay triangulations (inSphere predicate)
- overcome large memory consumption (hash table: clean periodically)
- filtered computations
- CGAL submission

Conclusions and Future work

- Orientation predicates. CH: $O(d^2)$, point location: $O(d)$
- Efficient (division free) dynamic determinant implementation
- More efficient CH implementation, $78\times$ speed-up in point location
- The code: `http://hdch.sourceforge.net`

Future work

- Delaunay triangulations (inSphere predicate)
- overcome large memory consumption (hash table: clean periodically)
- filtered computations
- CGAL submission

THANK YOU !!!