Reliable and Efficient Geometric Computation

Kurt Mehlhorn
Max-Planck-Institut für Informatik

slides and papers are available at my home page

My Waterloo Co-Authors

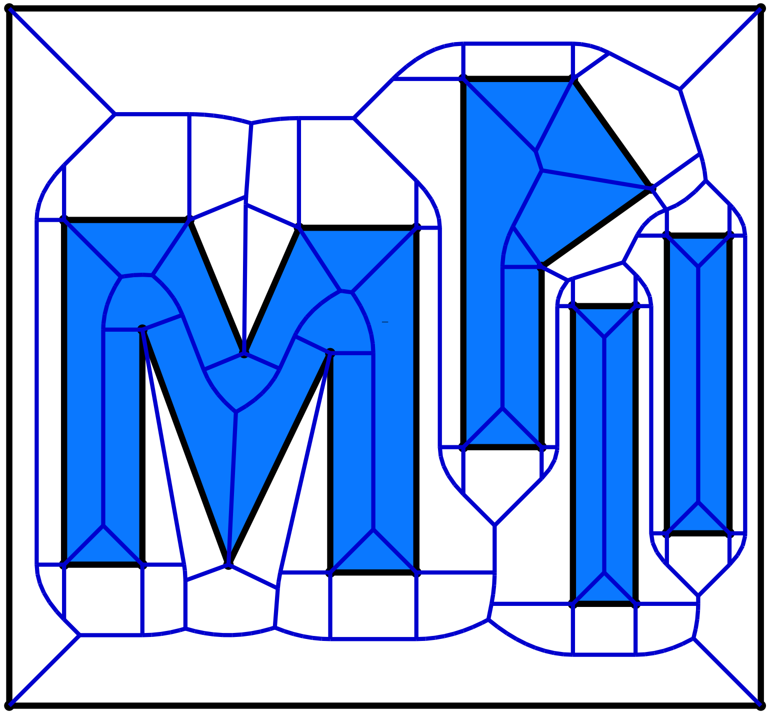
Cheriyan: Maximum Flow (SICOMP 96), Algs for Dense Graphs (Algorithmica 96), Highest-Level Selection (IPL 99)

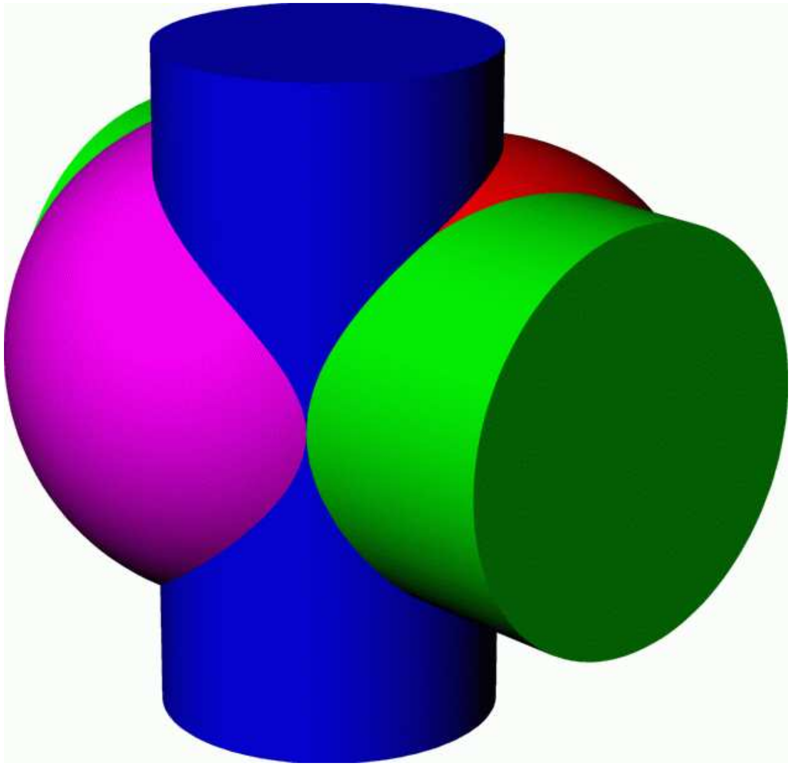
Koenemann: Exact Geometric Computation in LEDA (CompGeo 96)

Munro: Partial Match Retrieval (IPL 84), Random Variates (ICALP 93), Multiple Selection (ICALP 05)

Geometric Computing

MAX-PLANCK-GESELLSCHAFT





The Goal: Reliable and Efficient Geometric Computing

in particular, a reliable and efficient CAD kernel

reliable = produce a sensible output for all inputs

sensible output =

- the mathematically correct output or
- something provably close to the correct output

efficient = at most ten times slower than existing unreliable implementations

Why am I interested?

- mathematically challenging
- industrially relevant
- I blundered once: the first release of geometry in LEDA was unreliable

State of the Art

MAX-PLANCK-GESELLSCHAFT

Most existing implementations (commercial or academic) are unreliable

- may crash or produce non-sensical answers see next slide

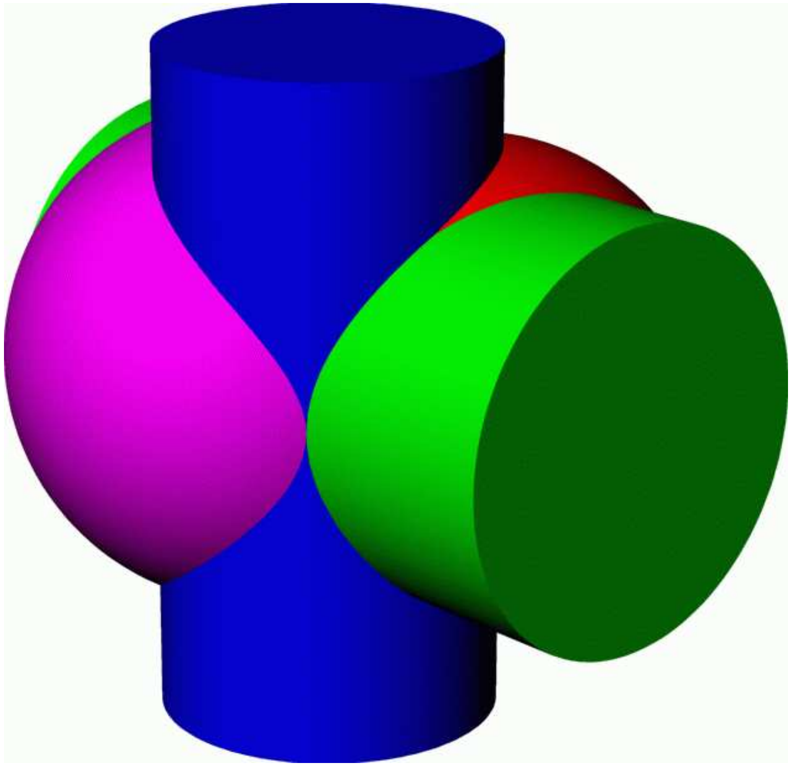
Where do we stand?

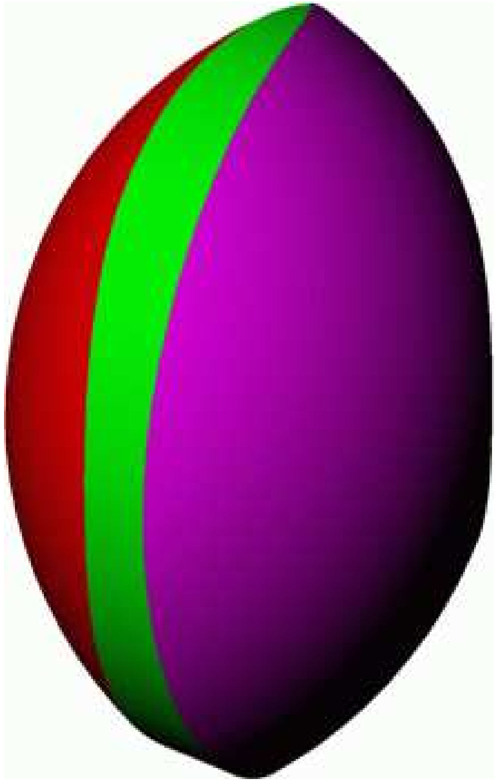
- we = reliable geometric algorithms project at MPI + EU-projects CGAL, GALIA, ECG and ACS
- linear (lines, planes, points) geometry in 2d and 3d: nice academic work + first industrial impact
- curved geometry in 2d: nice academic work + first industrial impact
- curved geometry in 3d: nice academic work
- implementations available in LEDA, CGAL, and EXACUS (ESA 2005)

How do we work?

- develop the required theory and system architecture and build prototypical systems to validate the theory and to have impact beyond our own community

Examples I: Intersection of 3d-Solids

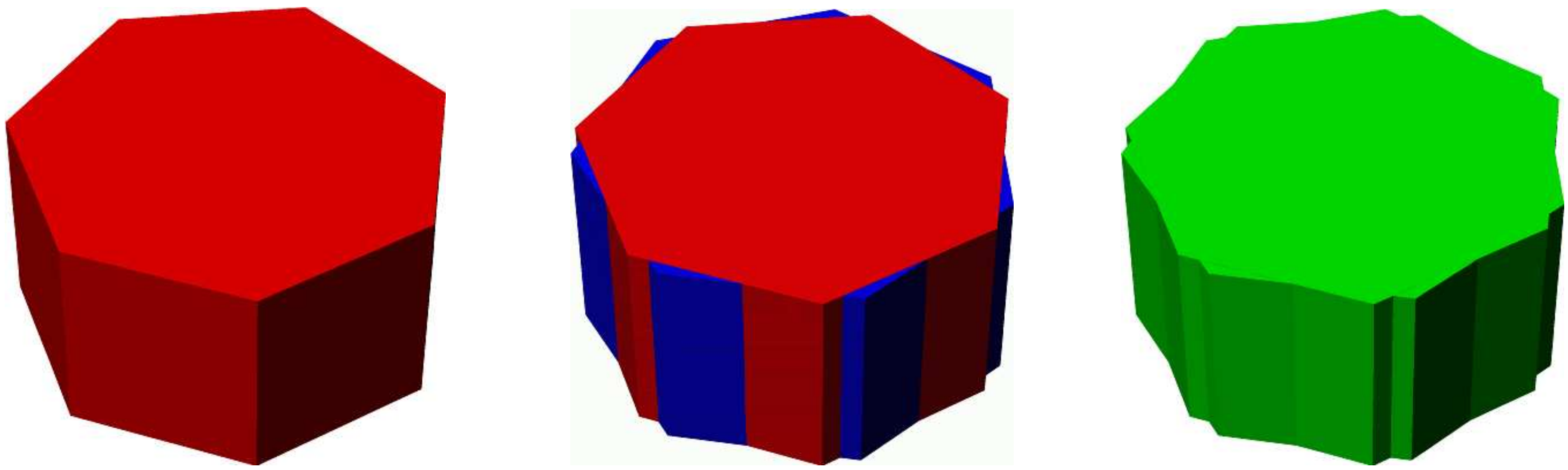




Rhino3d crashes on this input   the correct output

Examples II: Intersection of Planar 3d-Solids

Task: construct a regular cylinder P (base = regular n-gon) obtain Q from P by a rotation by α degrees about its center, and compute the union of P and Q

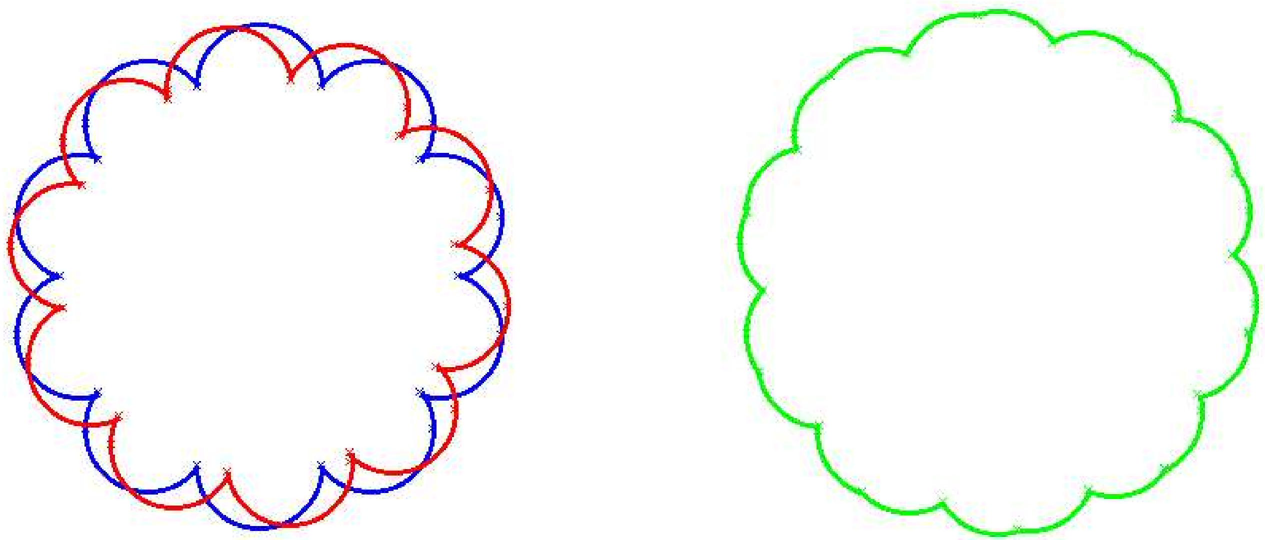


System	n	α	time	output
ACIS	1000	1.0e-4	30 sec	correct
ACIS	1000	1.0e-6	30 sec	incorrect
CGAL/LEDA	1000	1.0e-6	44 sec	correct
CGAL/LEDA	2000	1.0e-7	900sec	correct

Granados/Hachenberger/Hert/Kettner/Mehlhorn/Seel: ESA 2003

Hachenberger/Kettner: ESA 2005

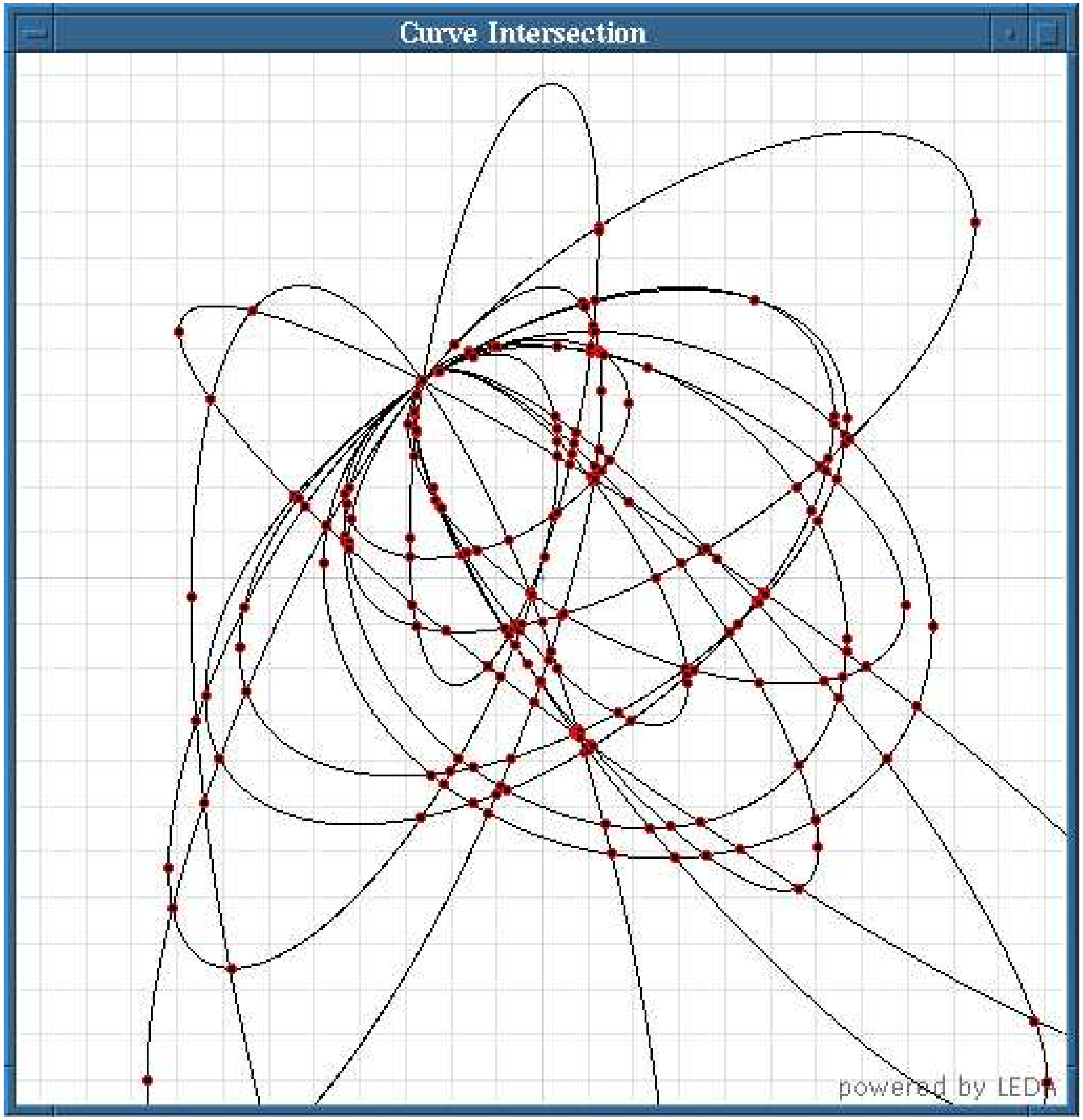
Example III: Curved Polygons



- the green polygon is the union of the red and the blue polygon
- edges are half-circles (more generally, conic arcs)
- computation takes about 30 seconds for polygons with 1000 edges
- requires extension of sweep line algorithm and exact computation with algebraic numbers of degree at most four

Berberich/Eigenwillig/Hemmer/Hert/Mehlhorn/Schömer: ESA 2002

Example IV: Degeneracies



A highly degenerate example:

- many curves have a common point
- different slopes
- same slope, different curvature,
- same slope and curvature, diff . . .

algorithm computes a planar map and not only a picture

Berberich/Eigenwillig/Hemmer/Schömer/W

CompGeo 2004

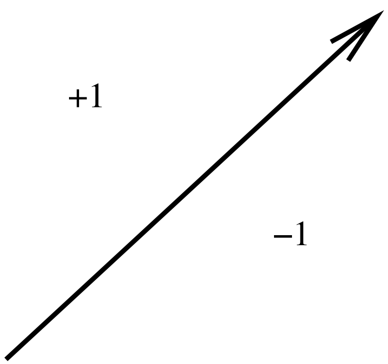
What is difficult?

MAX-PLANCK-GESELLSCHAFT

- algs are designed for the real-RAM and non-degenerate inputs
  - real-RAM = machine computes with real numbers in the sense of mathematics: exact roots of polynomials, sine, cosine, . . .
  - non-degenerate inputs: no three points on a line, no three curves through a point, . . .
- but real inputs are frequently degenerate and
- real computers are not real-RAMs (32 bit integer and double precision floating point arithmetic)
- the next three slides illustrate the pitfalls of floating point computation

The Orientation Predicate

- three points p, q, and r in the plane either lie on a common line or form a left or right turn $orient(p,q,r) = 0,\ +1,\ -1$



- analytically

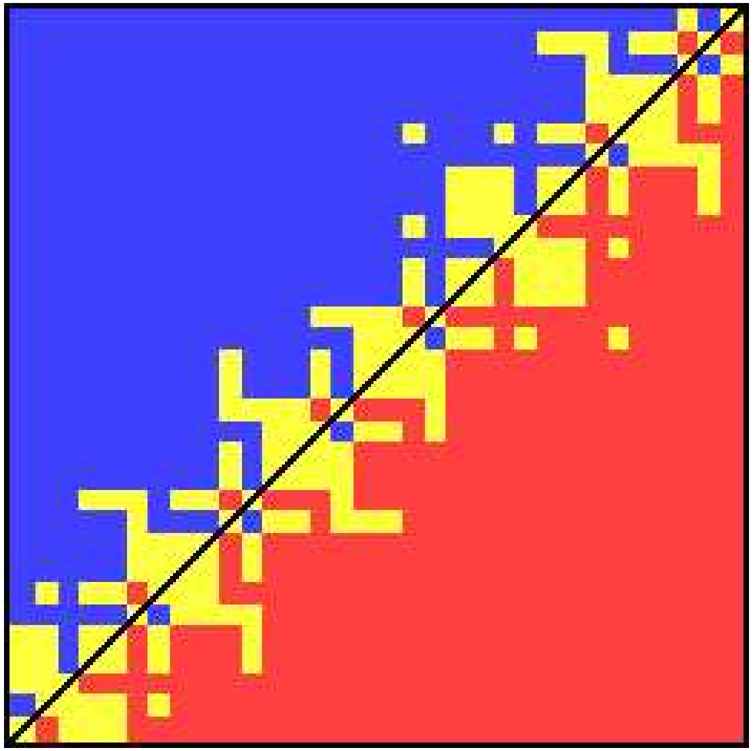
$$orient(p,q,r) = sign(\det \begin{bmatrix} 1 & p_x & p_y \\ 1 & q_x & q_y \\ 1 & r_x & r_y \end{bmatrix})$$

$$= sign((q_x - p_x)(r_y - p_y) - (q_y - p_y)(r_x - p_x)).$$

- det is twice the signed area of the triangle (p,q,r)
- $float\_orient(p,q,r)$ is result of evaluating $orient(p,q,r)$ in floating point arithmetic

Geometry of Float-Orient

$p = (0.5, 0.5)$, $q = (12, 12)$ and $r = (24, 24)$



0.5 $\qquad\qquad$ $0.5 + 255 \cdot 2^{-53}$

picture shows

$$float\_orient((p_x + xu, p_y + yu), q, r)$$

for $0 \leq x, y \leq 255$, where $u = 2^{-53}$.

the line $\ell(q, r)$ is shown in black

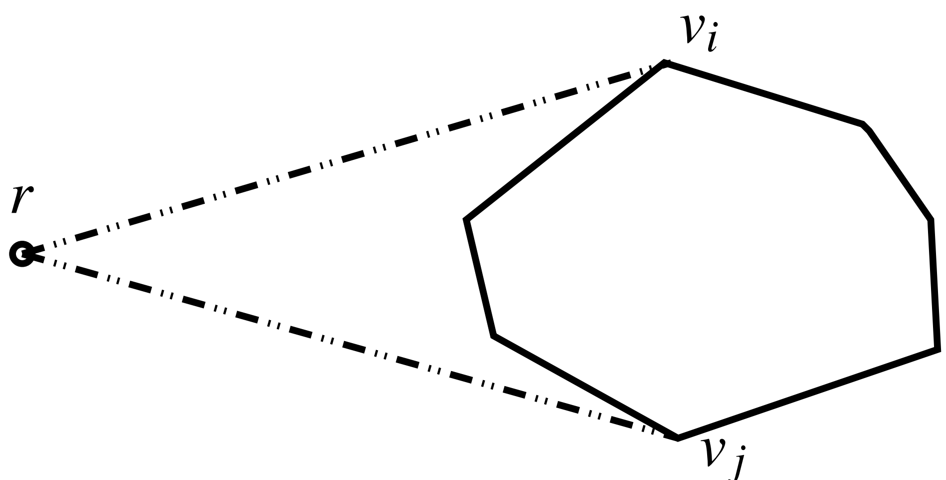
near the line many points are mis-classified

Kettner/Mehlhorn/Pion/Schirra/Yap: ESA 2004

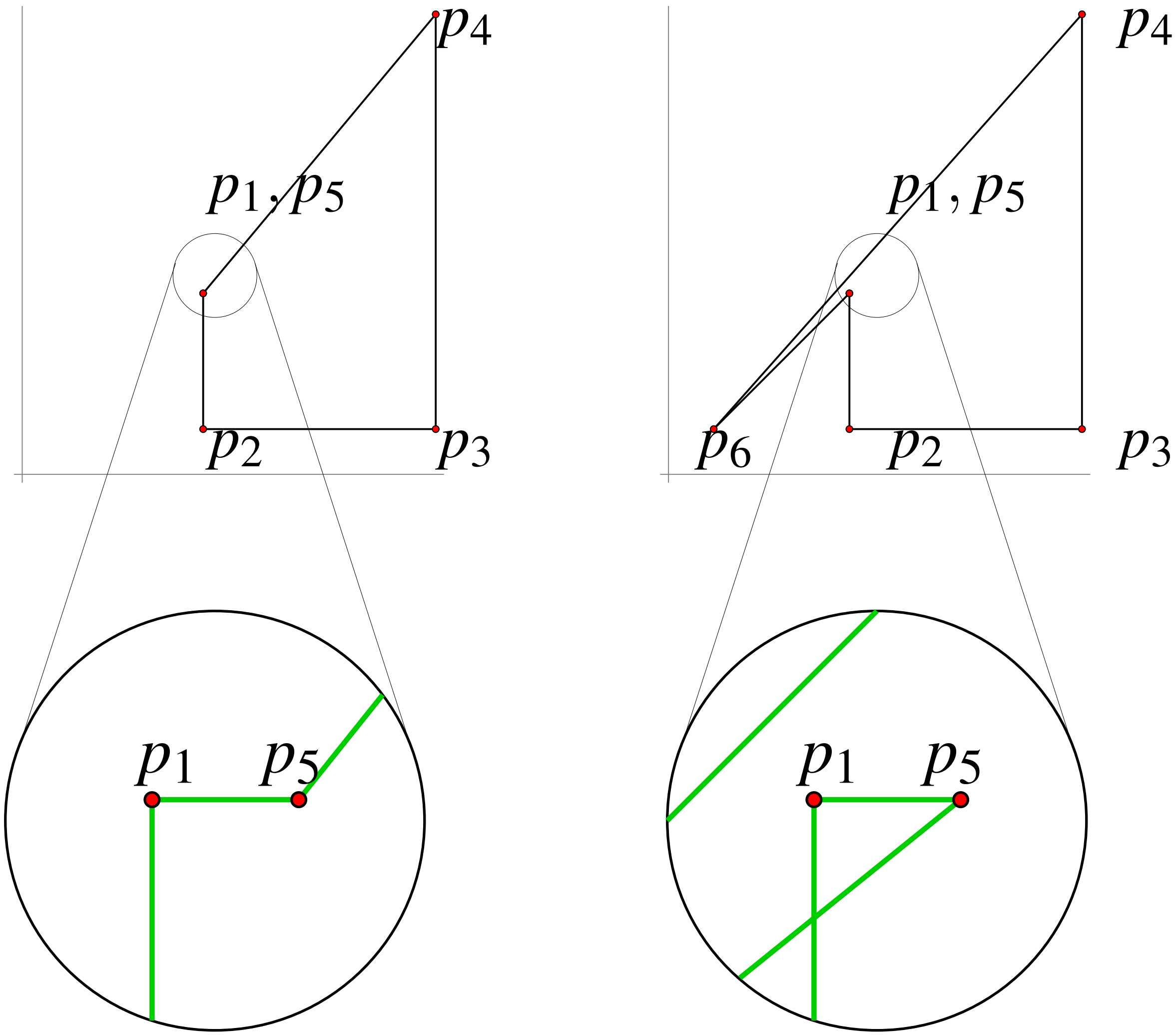
A Simple Convex Hull Algorithm

- alg considers the points one by one, maintains vertices of current hull in counter-clockwise order

- Initialize L to the counter-clockwise triangle (a,b,c).
  for all $r \in S$ **do**
  if there is an edge e visible from r **then**
  compute the sequence $(v_i,\ldots,v_j)$ of edges visible from r.
  replace the subsequence $(v_{i+1},\ldots,v_{j-1})$ by r.
  end if
  end for

- 

The Effect on a Simple Convex Hull Algorithm



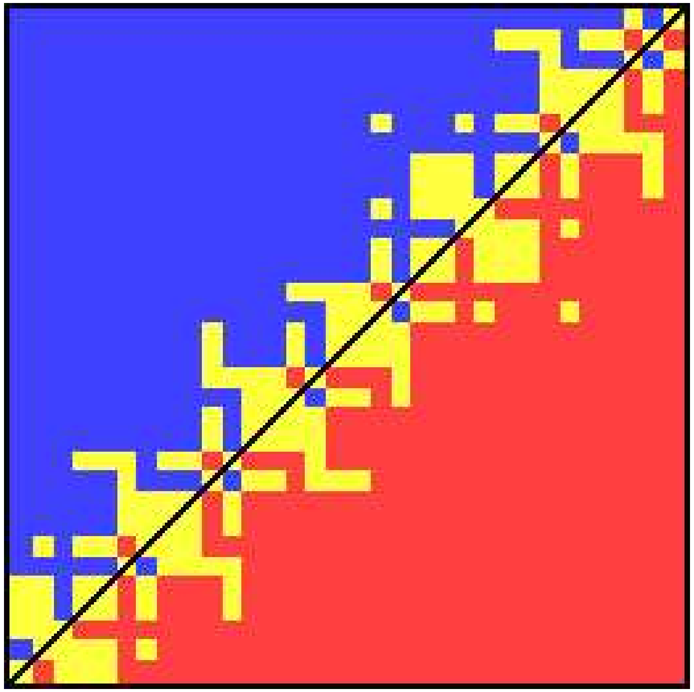
- the hull of p_1 to p_4 is correctly computed
- p_5 lies close to p_1, lies inside the hull of the first four points, but float-sees the edge (p_1, p_4). The magnified schematic view below shows that we have a concave corner at p_5.
- point p_6 sees the edges (p_1, p_2) and (p_4, p_5), but does **not** see the edge (p_5, p_1).
- we obtain either the hull shown in the figure on the right or ...

Solutions

- Solutions for single algorithms.
- The Exact Geometric Computation Paradigm (ECG)
  - implement a Real-RAM to the extent needed in computational geometry the challenge is efficiency
  - redesign the algorithms so that they can handle all inputs and have small arithmetic demand
  - Exact Computation Paradigm applies to all geometric algorithms
  - basis for LEDA, CGAL, and EXACUS
- Approximation via Controlled Perturbation
  - compute the correct result for a slightly perturbed input
  - initiated by Danny Halperin and co-workers and refined and generalized by us
  - Controlled perturbation applies to a large class of geometric algorithms
  - successfully used for Delaunay, Voronoi, arrangements of circles and spheres

Controlled Perturbation

Geometry of Float-Orient

- 

- picture shows

  $$\mathit{float\_orient}((p_x + xu, p_y + yu), q, r)$$

  for $0 \le x, y \le 255$, where $u = 2^{-53}$.

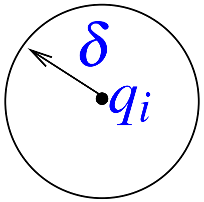
  the line $\ell(q, r)$ is shown in black
- near the line many points are mis-classified

- **outside a narrow strip around the curve of degeneracy, points are classified correctly !!!**
  - how narrow is narrow?
  - true for all geometric predicates?
  - if true, can we exploit to design reliable algorithms

Basics

- our program operates on points q_1 to q_n

  to perturb a point q_i:



- move it to random point p_i in the disk $B_\delta(q_i)$ of radius δ centered at q_i
- programs branch on the sign (+1, 0, -1) of expressions
- we use floating point arithmetic with mantissa length L
- the maximum error in evaluating an expression E is M_E
- $M_E = \text{something} \cdot 2^{-L}$
- if $|E| > M_E$, it is safe to evaluate E with floating point arithmetic and to branch on the sign of the result
- we have a geometric program that works for all non-degenerate inputs (if executed with exact real arithmetic)

Converting a Program to Controlled Perturbation

- guard every predicate evaluation, i.e.,

  replace branch on sign of E by

  if $(|E| \leq$ max error in evaluation of $E)$ stop with exception;
  branch on sign of E

- and then run the following master program
  - initialize δ and L to convenient values
  - loop
    - perturb input
    - run the guarded algorithm with floating point precision L
    - if the program fails, double L and rerun

- observe that program needs to be changed only slightly
  - guards for predicates and master loop
- guards can be avoided by use of interval arithmetic

Converting a Program to Controlled Perturbation

MAX-PLANCK-GESELLSCHAFT

- guard every predicate evaluation, i.e.,

  replace branch on sign of E by

  if $(|E| \leq$ max error in evaluation of $E)$ stop with exception;
  branch on sign of E

- and then run the following master program
  - initialize δ and L to convenient values
  - loop
    - perturb input
    - run the guarded algorithm with floating point precision L
    - if the program fails, double L and rerun

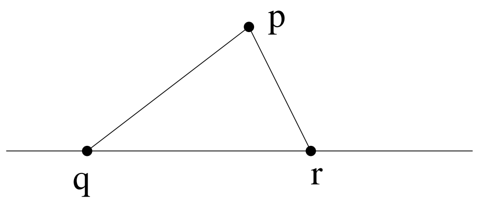
Theorem: For a large class of geometric programs: modified program terminates and returns the exact result for the perturbed input. Moreover (!!!), can quantify relation between δ and L.

Mehlhorn/Osbild/Sagraloff: ICALP 06

How Narrow is Narrow?

MAX-PLANCK-GESELLSCHAFT

- $orient(p,q,r) = sign((q_x - p_x)(r_y - p_y) - (q_y - p_y)(r_x - p_x)) = sign(E)$
- $E = 2\cdot$ signed area Δ of the triangle (p,q,r)
- if coordinates are bounded by M, maximal error in evaluating E with floating point arithmetic with mantissa length p is $28 \cdot M^2 \cdot 2^{-L}$
- if $2|\Delta| > 28 \cdot M^2 \cdot 2^{-L}$, *float_orient* gives the correct result



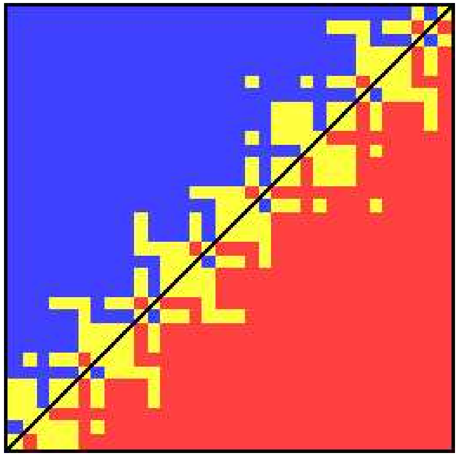
- $|\Delta| = (1/2) dist(q,r) \cdot dist(\ell(q,r),p)$
- if $dist(q,r) \cdot dist(\ell(q,r),p) > 28 \cdot M^2 \cdot 2^{-L}$,

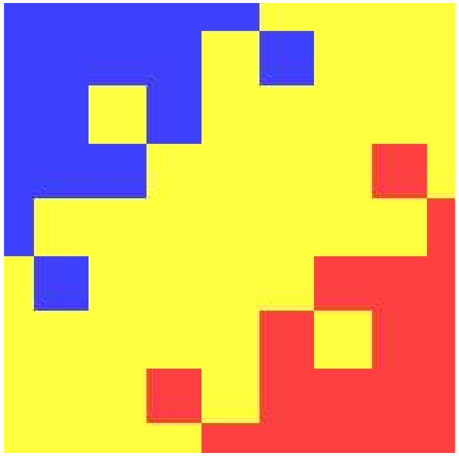
  float_orient gives the correct result
- if $dist(\ell((q,r),p)) \geq 28 \cdot M^2 \cdot 2^{-L} / dist(q,r)$,

  $float\_orient(p,q,r)$ gives the correct result.

How Narrow is Narrow?

- $orient(p,q,r) = sign((q_x - p_x)(r_y - p_y) - (q_y - p_y)(r_x - p_x)) = sign(E)$
- $E = 2\cdot$ signed area Δ of the triangle (p,q,r)
- if coordinates are bounded by M, maximal error in evaluating E with floating point arithmetic with mantissa length p is $28 \cdot M^2 \cdot 2^{-L}$
- Punch Line: if
  $dist(\ell((q,r),p)) \geq 28 \cdot M^2 \cdot 2^{-L}/dist(q,r)$,
  $float\_orient(p,q,r)$ gives the correct result.





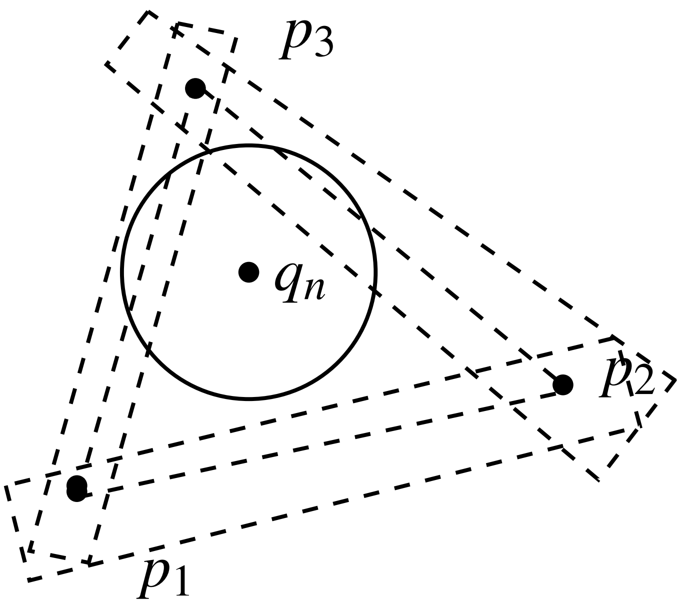
on the right, q and r have one third the distance than in figure on the left

Controlled Pertubation I

- consider algorithms using only the orientation predicate
- input points $q_1, \ldots, q_n$: perturb into $p_1, \ldots, p_n$ such that all evaluations for the perturbed points are f-safe.

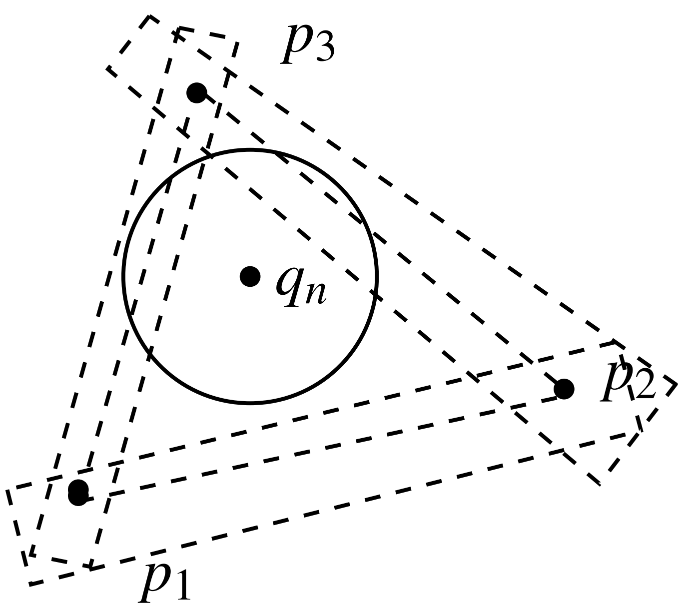
Controlled Pertubation I

- consider algorithms using only the orientation predicate
- input points $q_1, \ldots, q_n$: perturb into $p_1, \ldots, p_n$ such that all evaluations for the perturbed points are f-safe.
- assume p_1 to p_{n-1} are already determined:
  - choose p_n in a circle of radius δ about q_n such that whp
  - p_n lies outside all strips of half-width $28 \cdot M^2 \cdot 2^{-L}/dist(p_i, p_j)$ about $\ell(p_i, p_j)$ for $1 \leq i < j \leq n-1$



Controlled Pertubation I

- consider algorithms using only the orientation predicate
- input points $q_1, \ldots, q_n$: perturb into $p_1, \ldots, p_n$ such that all evaluations for the perturbed points are f-safe.
- assume p_1 to p_{n-1} are already determined:
  - choose p_n in a circle of radius δ about q_n such that whp
  - p_n lies outside all strips of half-width $28 \cdot M^2 \cdot 2^{-L}/dist(p_i, p_j)$ about $\ell(p_i, p_j)$ for $1 \leq i < j \leq n-1$



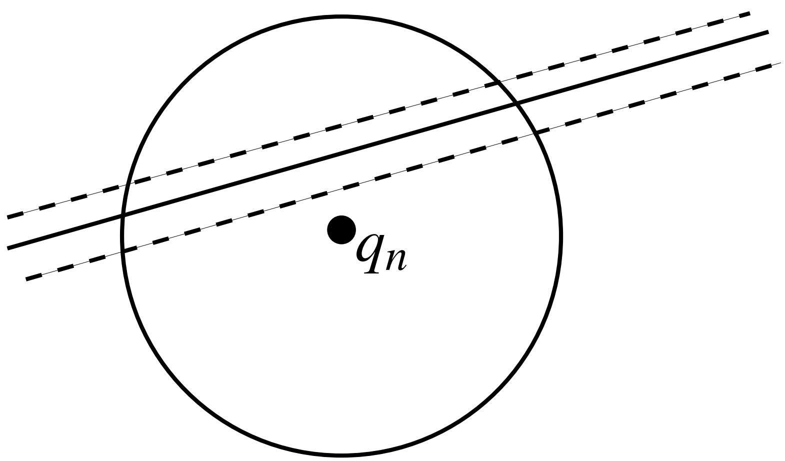
- whp = (choice of p_n fails with prob $\leq 1/(2n)$)
- prob, some choice fails is $\leq 1/2$
- with prob $1/2$, perturbed points are f-safe
- need that strips cover at most fraction $1/(2n)$ of ball $B_\delta(q_n)$

Controlled Perturbation II

- assume p_1 to p_{n-1} are already determined:
  - choose p_n in a circle of radius δ about q_n such that whp
  - p_n lies outside all strips of half-width $28 \cdot M^2 \cdot 2^{-L}/dist(p_i, p_j)$ about $\ell(p_i, p_j)$ for $1 \leq i < j \leq n-1$
  - need that strips cover at most fraction $1/(2n)$ of ball $B_\delta(q_n)$

Controlled Perturbation II

- assume p_1 to p_{n-1} are already determined:
  - choose p_n in a circle of radius δ about q_n such that whp
  - p_n lies outside all strips of half-width $28 \cdot M^2 \cdot 2^{-L}/dist(p_i, p_j)$ about $\ell(p_i, p_j)$ for $1 \leq i < j \leq n-1$
  - need that strips cover at most fraction $1/(2n)$ of ball $B_\delta(q_n)$
- A small problem : strips can be arbitrarily wide
- IDEA: also guarantee $dist(p_i, p_j) > \gamma$ for some γ
- then size of forbidden region $\leq n \cdot \pi \cdot \gamma^2 + n^2 \cdot (28 \cdot M^2 \cdot 2^{-L}/\gamma) \cdot 2 \cdot \delta$



Controlled Perturbation II

MAX-PLANCK-GESELLSCHAFT

- assume p_1 to p_{n-1} are already determined:
  - choose p_n in a circle of radius δ about q_n such that whp
  - p_n lies outside all strips of half-width $28 \cdot M^2 \cdot 2^{-L}/dist(p_i, p_j)$ about $\ell(p_i, p_j)$ for $1 \leq i < j \leq n-1$
  - need that strips cover at most fraction $1/(2n)$ of ball $B_\delta(q_n)$
- IDEA: also guarantee $dist(p_i, p_j) > \gamma$ for some γ
- then size of forbidden region $\leq n \cdot \pi \cdot \gamma^2 + n^2 \cdot (28 \cdot M^2 \cdot 2^{-L}/\gamma) \cdot 2 \cdot \delta$
- want: size of FR $\leq \pi \cdot \delta^2/(2n)$

Controlled Perturbation II

MAX-PLANCK-GESELLSCHAFT

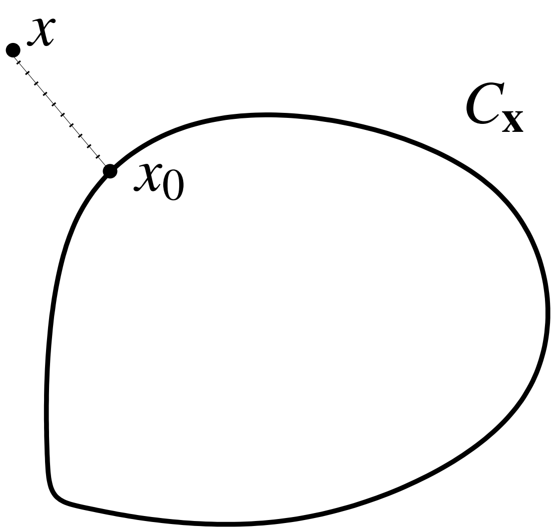
- assume p_1 to p_{n-1} are already determined:
  - choose p_n in a circle of radius δ about q_n such that whp
  - p_n lies outside all strips of half-width $28 \cdot M^2 \cdot 2^{-L}/dist(p_i, p_j)$ about $\ell(p_i, p_j)$ for $1 \le i < j \le n-1$
  - need that strips cover at most fraction $1/(2n)$ of ball $B_\delta(q_n)$
- IDEA: also guarantee $dist(p_i, p_j) > \gamma$ for some γ
- then size of forbidden region $\le n \cdot \pi \cdot \gamma^2 + n^2 \cdot (28 \cdot M^2 \cdot 2^{-L}/\gamma) \cdot 2 \cdot \delta$
- want: size of FR $\le \pi \cdot \delta^2/(2n)$
- fix γ so as to minimize FR and obtain

any $L \ge 2\log(M/\delta) + 4\log n + 9$ works

- $M = 1000,\ \delta = 0.001,\ n = 1000,\ L \ge 2 \cdot 20 + 4 \cdot 10 + 9 = 89$

Generalization to All (??) Geometric Predicates

general	orientation
predicate $P(x_1,\ldots,x_k) = \operatorname{sign} f(x_1,\ldots,x_k)$	$orient(p,q,r)$
x_1 to x_k points (in the plane)	
$\mathbf{x} = (x_1,\ldots,x_{k-1})$ fixed, $x = x_k$ variable	q, r fixed, p variable
$C_\mathbf{x} = \{x : f(\mathbf{x},x) = 0\}$, curve of degeneracy	$C = \{p : orient(p,q,r) = 0\}$
$C_\mathbf{x}$ is either the entire plane or a curve	plane or $\ell(q,r)$



Relate $f(\mathbf{x},x)$ to the distance of x from $C_\mathbf{x}$.

$$f(\mathbf{x},x) \geq g(\mathbf{x}) \cdot dist(C_\mathbf{x},x)$$

Forbidden region becomes tubular neighborhood of $C_\mathbf{x}$ of width $M_f/g(\mathbf{x})$

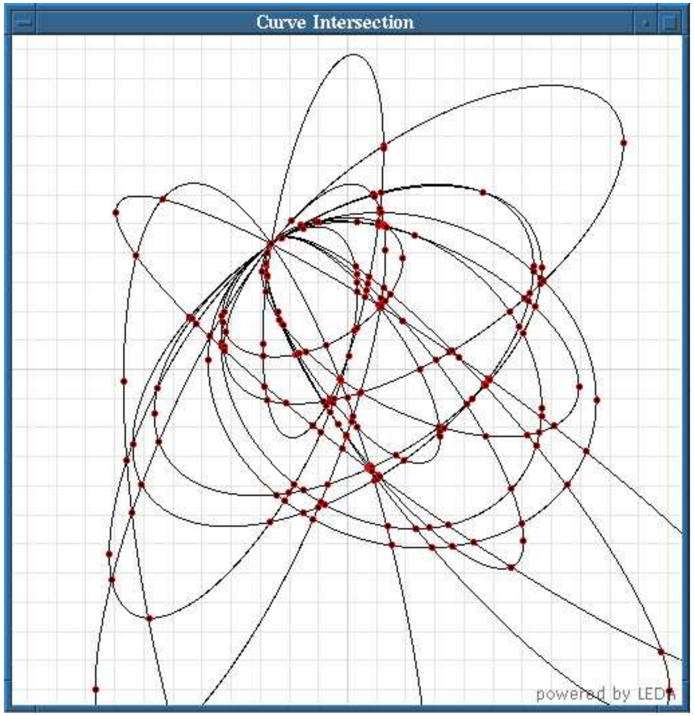
analyse $g(\mathbf{x})$ recursively

Generalization II

- in ICALP 06 paper, we show how to analyse a large class of predicates in the same way
  - predicates with a fixed number of arguments
- controlled perturbation applies to any algorithm
  - using only predicates as above and
  - whose running time is bounded as a function of number of input points
- most algorithms in CGAL are covered

The Exact Computation Paradigm

Improved Algs: Arrangements of Algebraic Curves



- algebraic curve = zero set of a polynomial in variables x and y
- assume rational coefficients
- $x^2 + y^2 = 9$ defines circle of radius 3
- compute x-coordinates of event points (vertical tangents, singularities, intersections)
- event point are algebraic numbers

- substitute x-values into algebraic curves and determine the roots of the resulting equations in y
- this requires to determine roots of polynomials with algebraic coefficients

- Seidel/Wolpert: CompGeo 2005: can do with roots of polynomials with rational coefficients

Efficient Computation with Algebraic Numbers

MAX-PLANCK-GESELLSCHAFT

- $p(x) = \sum_{0 \le i \le n} p_i x^i$, a polynomial of degree n
- $p_n \ge 1$, $p_i \le 2^\tau$ for all i — τ bits before binary point
- $sep(p)$ = minimum distance between any two roots of p, the root separation of p.
- **Theorem:** Isolating intervals for real roots can be computed in time polynomial in n and $\tau + \log 1/sep(p)$.
- more precisely, $O(n^4(\tau + \log(1/sep(p)))^2)$ bit operations
  requires $O(n(\tau + \log(1/sep(p))))$ bits of each coefficient
- for integer coefficients, our algorithm has the same complexity as previous algs
- experiments: $p(x)$ a polynomial with integer coefficients
  running times on $p(x)$, $\pi \cdot p(x)$, and $\sqrt{2} \cdot p(x)$ are essentially the same

Eigenwillig/Kettner/Krandick/Mehlhorn/Schmitt/Wolpert: CASC 2005

Summary

Most existing implementations (commercial or academic) are unreliable

- may crash or produce non-sensical answers

Where do we stand?

- we = reliable geometric algorithms project at MPI +
  EU-projects CGAL, GALIA, ECG and ACS
- linear (lines, planes, points) geometry in 2d and 3d: nice academic work + industrial impact
- curved geometry in 2d: nice academic work + industrial impact
- curved geometry in 3d: nice academic work
- implementations available in LEDA, CGAL, and EXACUS (ESA 2005)

How do we work?

- develop the required theory and build prototypical systems to validate the theory and to have impact beyond our own community