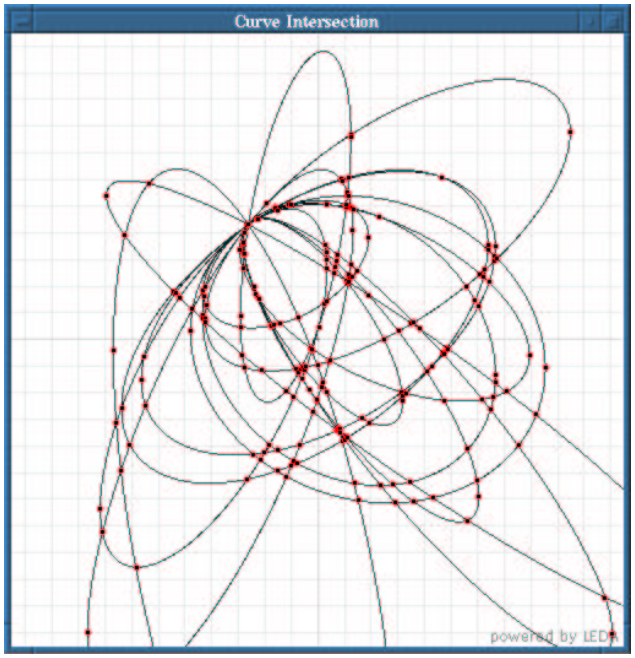
EXACUS

Efficient and Exact Computations with Curves and Surfaces



Eric Berberich Arno Eigenwillig Stefan Funke Michael Hemmer
Susan Hert Lutz Kettner Christian Lennerz Kurt Mehlhorn
Joachim Reichel Elmar Schömer Susanne Schmidt Thomas Warken
Dennis Weber Nicola Wolpert

Max-Planck-Institut für Informatik

Goals

- long term: create the basis for the next generation CAD systems
- more concretely: build a system for exact and efficient computational geometry and solid modeling with curves and surfaces
- develop and extend the underlying theory
- by analogy: a LEDA/CGAL for the curved world

Status

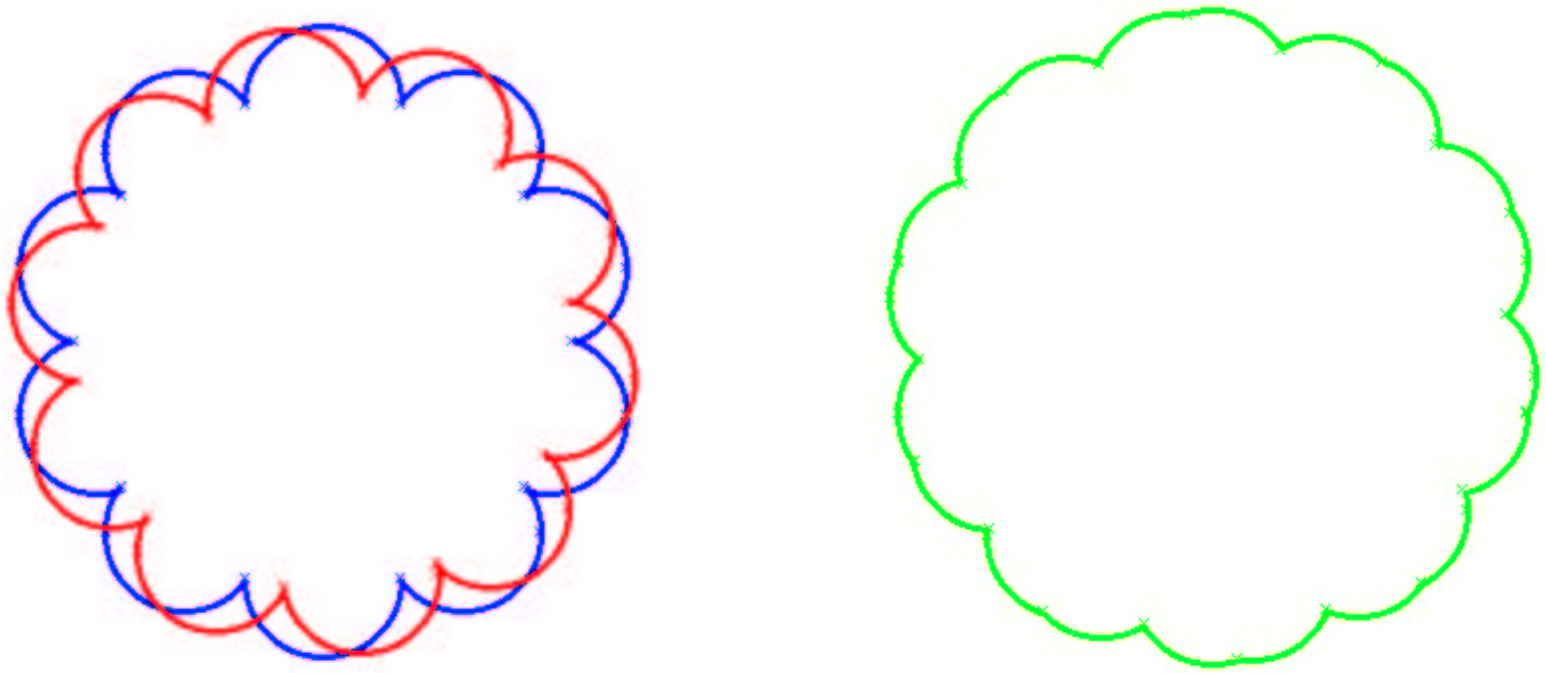
- Elmar, Lutz, and I have taught a course and a seminar in WS 01/02
- now: Seminar, FOPRAs, Diplomarbeiten, Doktorarbeiten, ...
- part of EU-project "Effective Computational Geometry", partners: Tel Aviv, Inria Sophia-Antipolis, ETH Zürich, Groningen, FU Berlin.
- first handful of papers written (I will report on three)
- EXACUS library project started

What is Wrong with Current CAD Systems?

- current CAD systems are **not** reliable
- construct a regular n-gon P, (or cylinder)
- obtain Q from P by a rotation by α degrees about its center,
- compute the union of P and Q (= a $4n$ gon).

Dimension	System	n	α	time	output
3D	ACIS	1000	1.0e-4	5 min	correct
3D	ACIS	1000	1.0e-5	4.5 min	correct
3D	ACIS	1000	1.0e-6	30 sec	problem too difficult
3D	Microstation95	100	1.0e-2	2 sec	correct
3D	Microstation95	100	0.5e-2	3 sec	incorrect answer
3D	Rhino3D	200	1.0e-2	15sec	correct
3D	Rhino3D	400	1.0e-2	–	CRASH
2D	CGAL/LEDA	5000	6.175e-06	30 sec	correct
2D	CGAL/LEDA	5000	1.581e-09	34 sec	correct
2D	CGAL/LEDA	20000	9.88e-07	141 sec	correct

Examples I

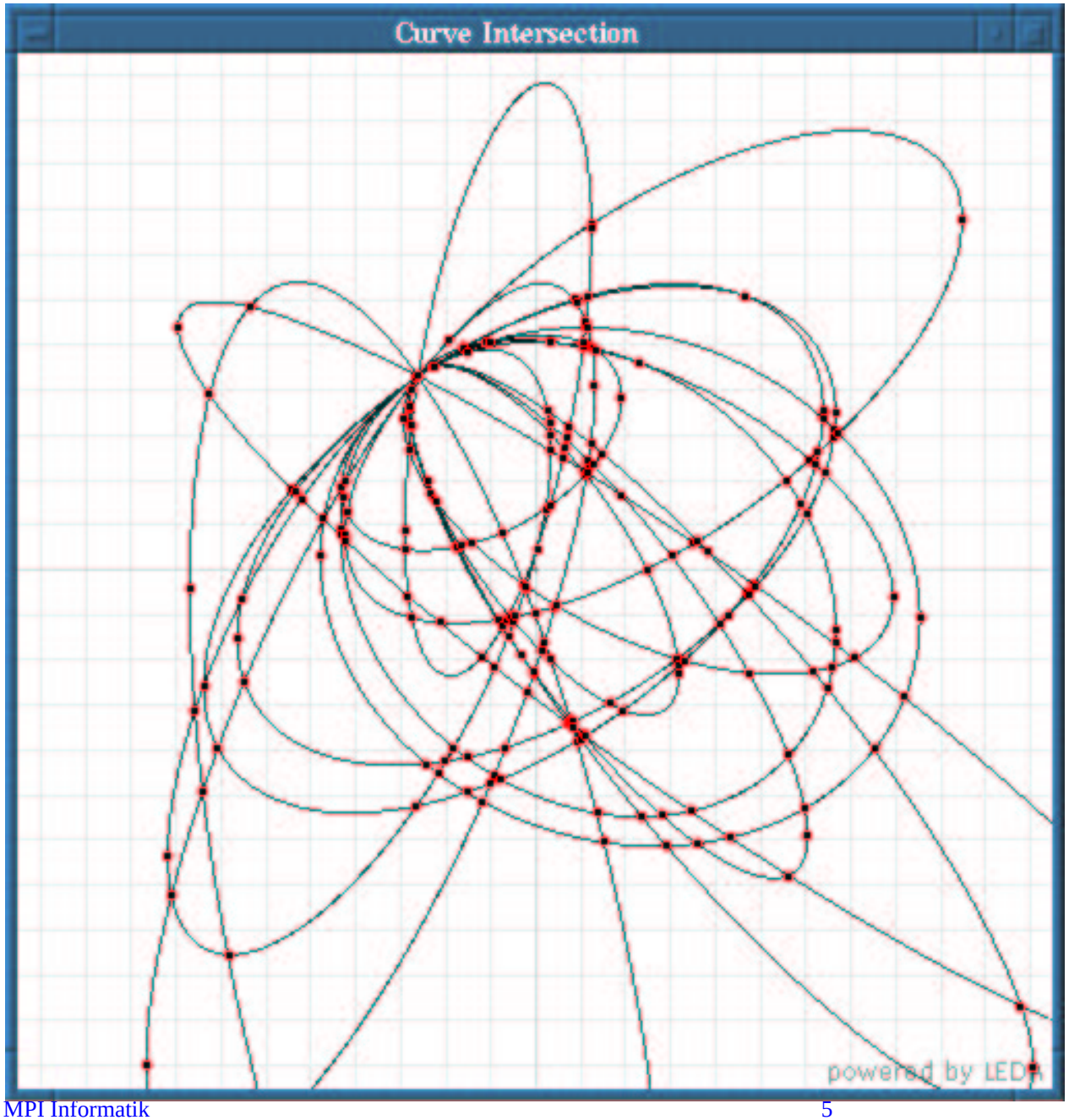


polygons with circular arcs

green polygon is union of red and blue polygon

computation of union takes about 1 minute for two roses with 1000 petals.

Example II



algorithms can handle arbitrary degeneracies

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

computation takes about 15 seconds

Contents

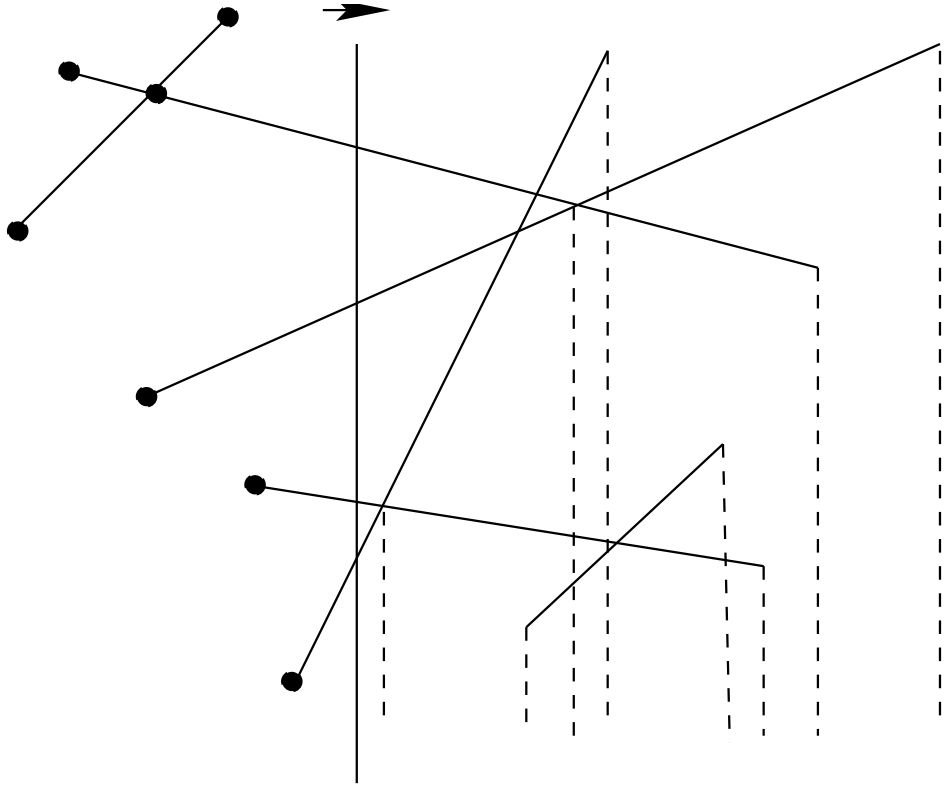
- Bentley-Ottmann sweep for curves
- required predicates and functions
- boolean operations on polygons and curve intersection
- intersections of conics
- algebraic numbers, implicit and explicit
- algorithms for the predicates and optimizations
- open problems

- E. Berberich, A. Eigenwillig, M. Hemmer, S. Hert, K. Mehlhorn, E. Schömer: *A Computational Basis for Conic Arcs and Boolean Operations on Conic Polygons*, submitted
- C. Burnikel, S. Funke, K. Mehlhorn, S. Schirra, S. Schmitt: *A Separation Bound for Real Algebraic Expressions*, ESA 2001, LNCS 2161, 254–263
- K. Mehlhorn: *A Remark on the Sign Variation Method for Real Root Isolation*, to appear in Journal of Symbolic Computation
- N. Geismann, M. Hemmer, E. Schömer: *Computing a 3-dimensional Cell in an Arrangement of Quadrics: Exactly and Actually*, ACM Conference on Computational Geometry 2001
- N. Wolpert: *Algorithms for Arrangement of Quadrics*, PhD-thesis, 2002
- M. Hemmer: *Computational Geometry for Conics*, Diplomarbeit, April 2002

Bentley-Ottmann Sweep for Line Intersection

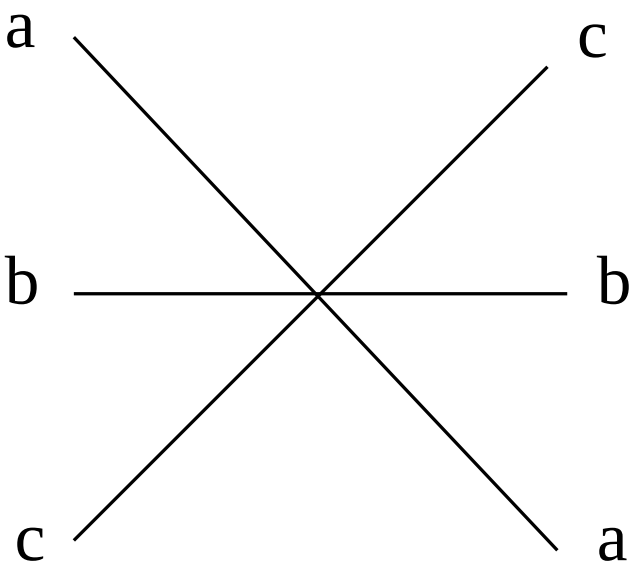
input: a set of line segments

output: the planar map (arrangement) G defined by the segments; G has one vertex for each endpoint and each intersection

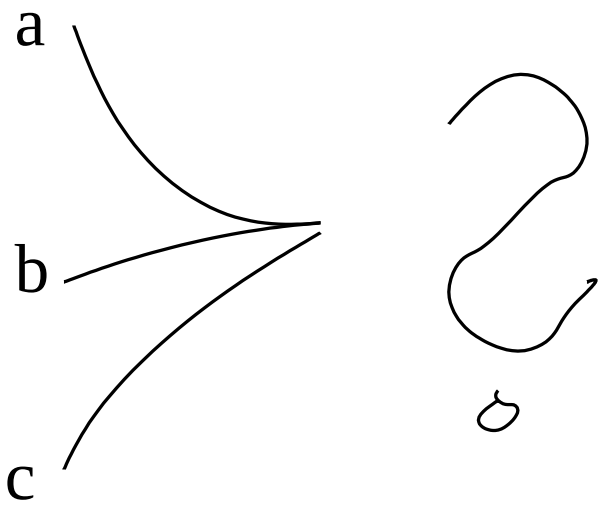


- sweep a vertical line L across the plane and maintain
- Y-structure = sorted sequence of intersections between L and given curves
- X-structure = already known vertices ahead of sweep line
- update at event points
- G emerges to the left of L

Sweeping Through a High Degree Vertex I



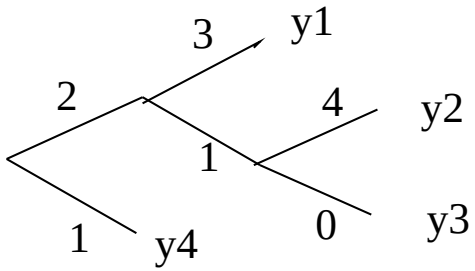
- in the case of *segments*:
  - when we sweep through a vertex, the *y*-order of the segments is reversed
  - and hence the update of the Y-structure is fairly simple
  - linear time in degree of the vertex

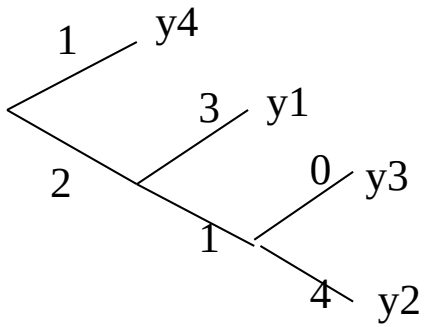


- in the case of *curves*:
  - when we sweep through a vertex, the *y*-order of the curves changes according to an arbitrary permutation
  - or maybe not so arbitrary ?

Sweeping Through a High Degree Vertex II

- assume for simplicity that common point is origin
- (conceptually) write the curves as power series in x and arrange in a trie.
- $y_1 = 2x + 3x^2$ $y_2 = 2x + 1x^2 + 4x^3$ $y_3 = 2x + 1x^2$ $y_4 = 1x + 7x^2$





- trie on left reflects order for small positive x
- order for small negative x is obtained by flipping children at odd depth
- only shape of trie is important
- paths for y_2 and y_3 split at depth two since multiplicity of intersection is three.
- intersection = multiplicity 1, same slope = 2, same curvature = 3, same . . . = 4, . . .
- update time is linear in degree times cost of determining multiplicities of intersection

Predicates and Functions Required for the Sweep

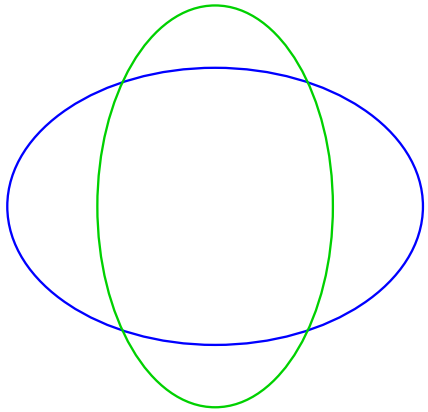
- given two conics C and D, compute their intersection points
- lex-order on intersection points and starting and endpoints

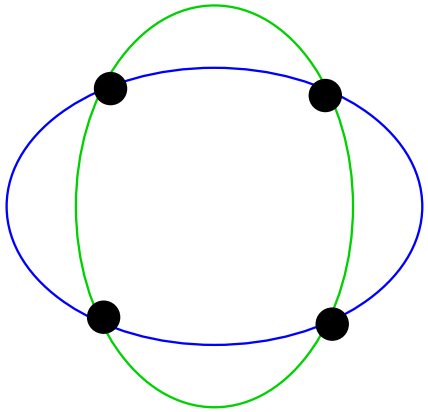
  for order of X-structure and for inserting starting segments into Y-structure
- multiplicity of intersections

  for sweeping through a vertex

Boolean Ops on Polygons and Intersection of Curves

- once the arrangement induced by the boundaries is known, boolean ops reduce to graph traversal





The Arrangement of Two Conics I

- a conic C: $\alpha_1 x^2 + \alpha_2 y^2 + 2\alpha_3 xy + 2\alpha_4 x + 2\alpha_5 y + \alpha_6 = 0$
- assume for simplicity that $\alpha_2 \neq 0$, the other case being simpler
- solve for y and obtain equations for the upper and lower arcs of C

$$C_{1,2}(x) = \frac{-b(x) \pm \sqrt{b(x)^2 - 4a(x)c(x)}}{2a(x)}$$

where $a(x) = \alpha_2 \qquad b(x) = 2\alpha_3 x + 2\alpha_5 \qquad c(x) = \alpha_1 x^2 + 2\alpha_4 x + \alpha_6$.

- x-coordinates of points of vertical tangents are one-root-expressions (OREs): $r + s\sqrt{t}$ with $r, s, t \in \mathbb{Q}$.
- two conics C and D can have up to four intersections
- x-coordinates of intersections are roots of a polynomial R_{CD} of degree at most four
- $R = R_{CD}$ is readily computed from equations of C and D resultant of C and D
- can determine number of real roots of R, their multiplicity, and isolating intervals by methods of computer algebra (we use Uspensky's method)
- $I = [a, b]$ is an isolating interval for a real root x of P iff x is the unique root of P in I.

Computing with Radical Expressions

Let E be an expression with integer operands and operators $+$, $-$, $*$ and $\sqrt{}$. Define

- $u(E) =$ value of E after replacing $-$ by $+$.
- $k(E) =$ number of distinct square roots in E.

Then (BFMS, BFMSS)

$$E = 0 \quad \text{or} \quad |E| \geq \frac{1}{u(E)^{2^{k(E)}-1}}$$

Theorem allows us to determine signs of algebraic expressions by numerical computation with precision $(2^{k(E)} - 1) \log u(E)$.

related work: Mignotte, Canny, Dube/Yap, Li/Yap, Scheinermann

extensions: division, higher-order roots, roots of univariate polynomials

Discussion I

How small can $A - B\sqrt{C}$ be, if non-zero? $A, B, C \in \mathbb{N}$.

$$|A - B\sqrt{C}| = \left| \frac{(A - B\sqrt{C})(A + B\sqrt{C})}{A + B\sqrt{C}} \right| = \frac{|A^2 - B^2 C|}{|A + B\sqrt{C}|} \geq \frac{1}{|A + B\sqrt{C}|} \geq \frac{1}{|A| + |B|\sqrt{C}}$$

This is a special case of the theorem

- $u(E) = |A| + |B|\sqrt{C}$
- $k = 1$

Discussion II

- Consider $E = (\sqrt{x+1}+\sqrt{x})\cdot(\sqrt{x+1}-\sqrt{x})-1$ where x is an arbitrary integer.
- Observe $E = 0$.
- $u(E) \approx 4x+1 \approx 4x$ and $k(E) = 2$.
- Thus

$$E = 0 \quad \text{or} \quad |E| \geq \frac{1}{u(E)^{2^{k(E)}-1}} \approx \frac{1}{(4x)^3}$$

- It suffices to evaluate E with precision $3\log(4x) = 3\log x + 6$.

Numerical Sign Computation

$sep(E) \leftarrow u(E)^{1-2^{k(E)}}$; // bound from previous slide

$k \leftarrow 1$;

while (true)

{ compute an approximation $\widetilde{E}$ with $|E - \widetilde{E}| < 2^{-k}$;

if ($|\widetilde{E}| \geq 2^{-k}$) return $sign(\widetilde{E})$;

if ($2^{-k} < sep(E)/2$) return "sign is zero"; // since $|E| \leq 2^{-k} + 2^{-k} < sep(E)$

$k \leftarrow 2 \cdot k$; // double precision

}

- $\widetilde{E}$ is computed by numerical methods
- worst case complexity is determined by separation bound:

  maximal precision required is logarithm of separation bound
- easy cases are decided quickly (a **big** plus of the separation bound approach)
- strategy above is basis for sign test in LEDA *real*s.

The LEDA Number Type *REAL*

The theorem is packaged in the LEDA data type *real*. It provides exact arithmetic for arithmetic expressions involving roots.

```
real x = ...  some integer ...;
real sx = sqrt(x);
real sxp = sqrt(x+1);
real A = (sxp + sx) * (sxp - sx);  // = 1
real B = A - 1;                    // = 0
cout << A.sign();                  // 1
cout << B.sign();                  // 0
```

If x has 100 binary places this takes less than .1 seconds. Run demo.

Reals are used in many geometric programs, e.g., Voronoi diagrams of line segments, boolean operations on curved polygons, arrangements of ellipsoids, . . .

The Arrangement of Two Conics C and D, Part II

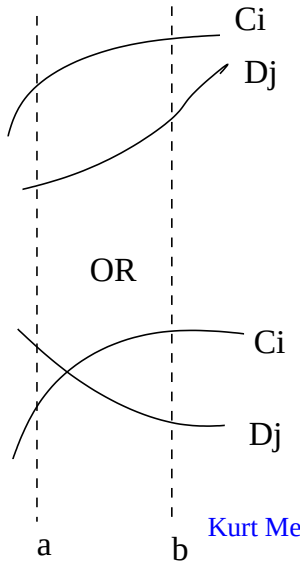
- do arcs C_i and D_j intersect at ORE x?
- a conic C: $\alpha_1 x^2 + \alpha_2 y^2 + 2\alpha_3 xy + 2\alpha_4 x + 2\alpha_5 y + \alpha_6 = 0$
- $$C_{1,2}(x) = \frac{-b(x) \pm \sqrt{b(x)^2 - 4a(x)c(x)}}{2a(x)}$$

  where $a(x) = \alpha_2 \qquad b(x) = 2\alpha_3 x + 2\alpha_5 \qquad c(x) = \alpha_1 x^2 + 2\alpha_4 x + \alpha_6$.
- similarly for $D_{1,2}$
- is $C_i(x) = D_j(x)$? where $x = r + s\sqrt{t}$ with $r, s, t \in Q$
- this question can be answered using LEDA reals, namely

  $$E = 0 \text{ ?} \quad \text{where} \quad E = C_i(r + s\sqrt{t}) - D_j(r + s\sqrt{t})$$
- $k = 3$

The Arrangement of Two Conics C and D, Part III

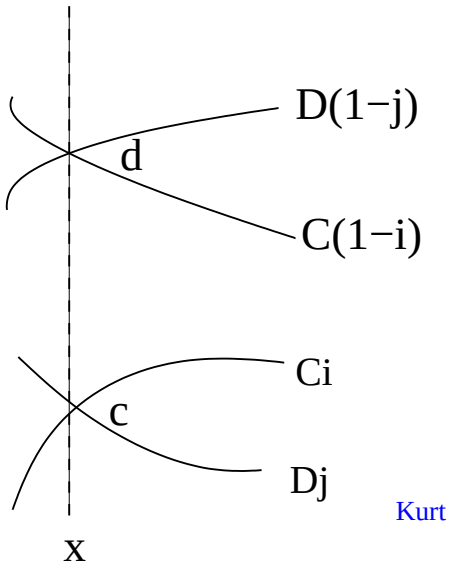
- have a deg four polynomial $R = R_{CD}$: x-coordinates of intersections are roots of R.
- a degree four polynomial either has four distinct roots or roots are OREs.
- if C and D intersect with multiplicity m at $(x,.)$ then x is root of R of multiplicity at least m
- assume x is a root of R
  - do C_i and D_j intersect at x?
  - if x is given as ORE, compare $C_i(x)$ and $D_j(x)$ using LEDA-reals
  - if x is not given as an ORE, x is a simple root of R and hence
    - we have an isolating interval $[a,b]$ for x
    - C_i and D_j intersect at most once in $[a,b]$ and, if so, at x
    - if C_i and D_j intersect at x, they cross at x
    - cross if $sign(C_i(a) - D_j(a)) \neq sign(C_i(b) - D_j(b))$
    - decide the latter using LEDA reals



The Arrangement of Two Conics C and D, Part IV

- assume C_i and D_j intersect at x.
- what is multiplicity c of intersection?
- $m =$ multiplicity of x as root of $R = R_{CD}$; then $m \leq 4$.
- if $m = 1$ we have $c = 1$ and are done
- if $m \geq 2$, we have an ORE for x
- compute tangent vectors for C_i and D_j at x as LEDA-reals by considering partial derivatives
- if not parallel, $c = 1$, otherwise $c \geq 2$.

- $d =$ multiplicity of intersection of C_{1-i} and D_{1-j} at x.
- then $c + d = m$ this requires a little lemma
- determine whether $d = 0$, $d = 1$, $d \geq 2$.
- together with $c \geq 2$ and $m \leq 4$, this determines c



Lexicographic Order of Intersections

- Consider intersections (x_1, C_i, D_j) and (x_2, E_k, F_l) where x_i is either an ORE or a simple root of degree four polynomial.

  Compare x_1 and x_2:

  If equal, compare y-coordinates:

Use Similar Techniques

Correctness of Implementation

- have manually checked output for a small number of examples
- have run alg on a large number of random examples
  - always checked that arrangement computed by the sweep is a planar map, i.e., satisfies Euler's formula

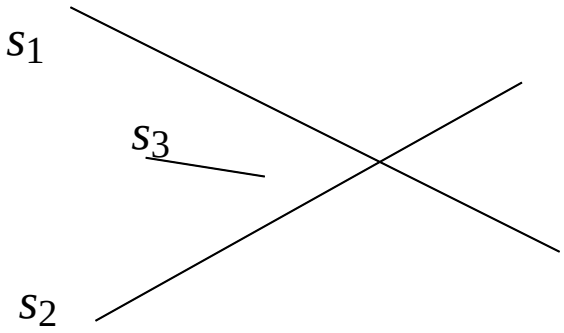
$$number\ of\ vertices - number\ of\ edges + number\ of\ face\ cycles = 2$$

  - for boolean ops on polygons P and Q:
    threw random points p into the plane and checked

$$(p \in P)\ op\ (p \in Q) = p \in (P\ op\ Q)$$

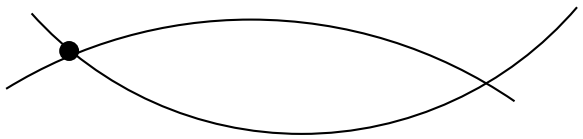
Optimization I: Prefer Combinatorics over Numerics

- dictionary for intersections



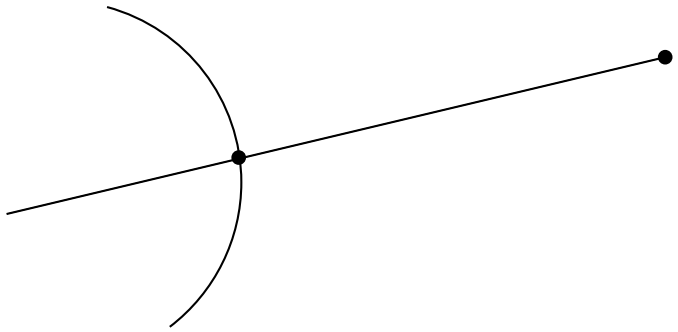
  - $s_1 \cap s_2$ is recomputed and reinserted into X-struct after s_3 is swept
  - it is better to store it in a dictionary (under key (s_1, s_2))
  - optimization is used in LEDAsweep and generic sweep, but now it really pays

- fast equality test for identically constructed points



  - assume sweep point is at left intersection of circular arcs
  - at this point: we compare the sweep point to itself
  - this is extremely costly when done numerically

Optimization II: Retain Geometric Information

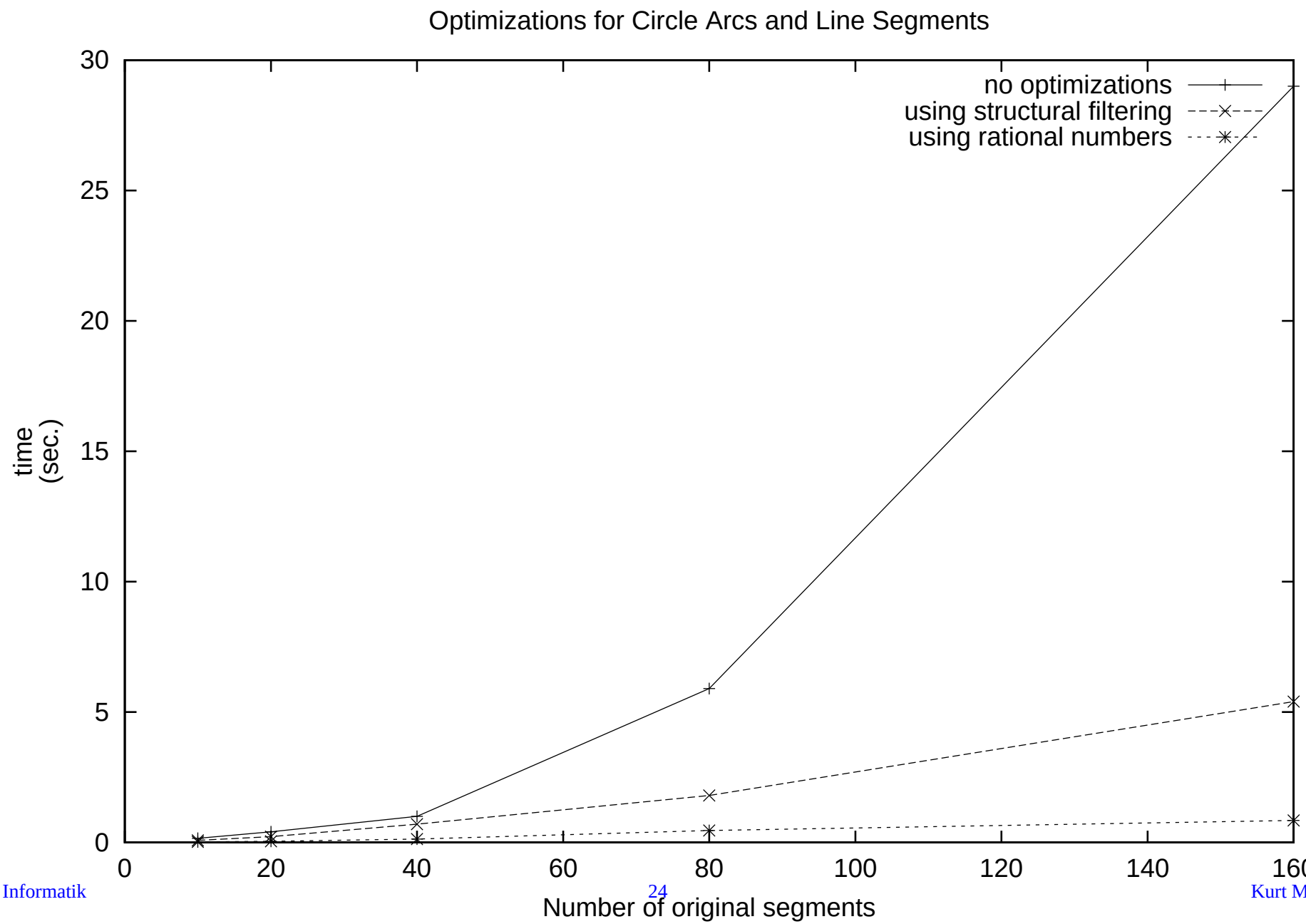


- default implementation of segments: store endpoints
- when endpoints are circle points it is very costly to recover the underlying line
- better implementation: store underlying line and the endpoints
- generalizes to arbitrary curves: store underlying curve and two curve points

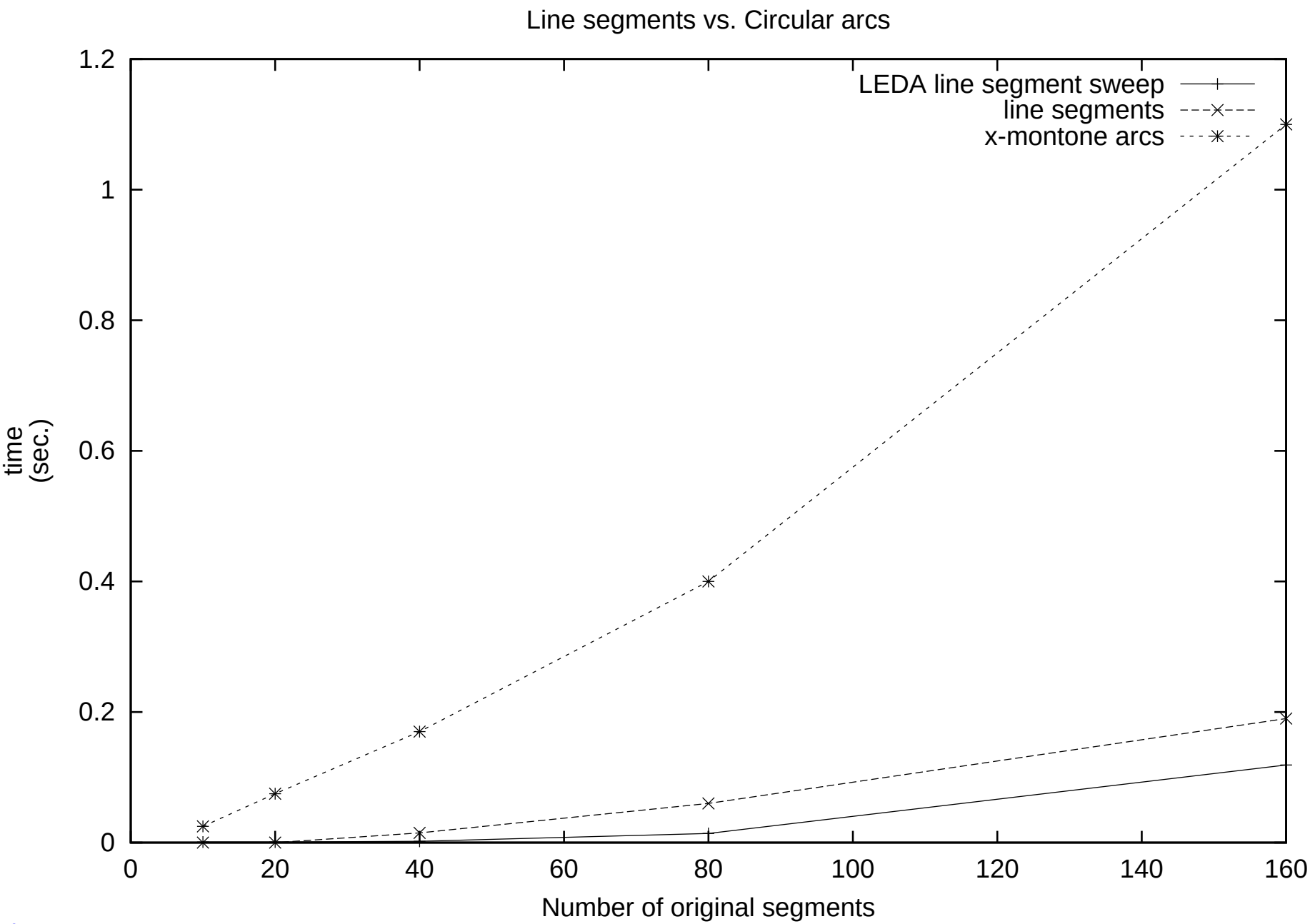
Optimiziation III: Number Types

- always use the simplest possible number type
- use floating point filtering

Timings I


Optimizations for Circle Arcs and Line Segments
no optimizations
using structural filtering
using rational numbers
time (sec.)
Number of original segments
0
5
10
15
20
25
30
0
20
40
60
80
100
120
140
160

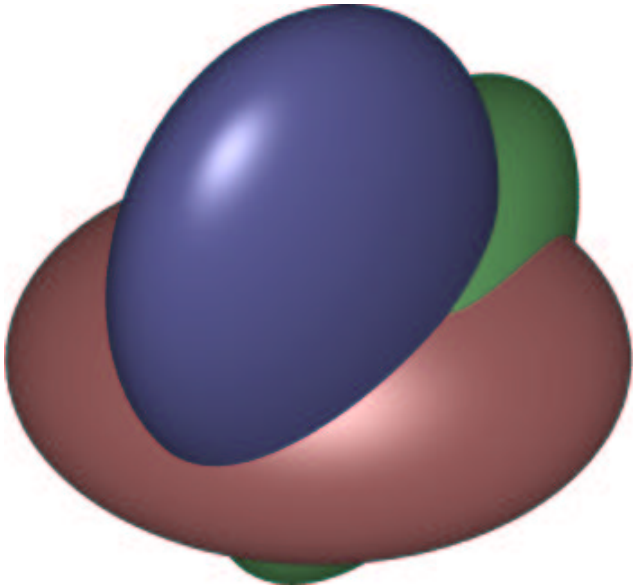
Timings II


Line segments vs. Circular arcs
LEDA line segment sweep
line segments
x-montone arcs
time (sec.)
Number of original segments
0
0.2
0.4
0.6
0.8
1
1.2
0
20
40
60
80
100
120
140
160

Beyond Conics

- computational geometry part works for arbitrary curves
- implementation of predicates and functions requires considerable refinement, e.g.,
  - need arithmetic ops on algebraic numbers given as roots of polynomials
  - need analysis of singularities
  - …

and into 3D



computed by Michael Hemmer, Elmar Schömer, and Nicola Wolpert