Arrangements and Their Applications

Part II: Implementation

Dan Halperin

danha@tau.ac.il

Tel Aviv University

Prevailing assumptions in CG theory

- worst-case asymptotic complexity measures
- unit cost of operations on a constant number of simple objects
- the real RAM model, infinite precision real arithmetic
- general position, no degeneracies

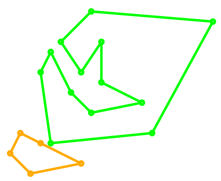
Question

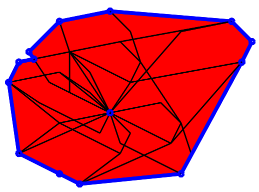
given two lines ℓ_1, ℓ_2 that intersect in a single point p, does p lie on ℓ_1?

line intersection demo, LEDA

Robustness: the problematic assumptions

- infinite precision real arithmetic
- general position





the two issues are intertwined: (near) degenerate configurations incur precision problems

geometric algorithms: interplay between numerics and combinatorics

exactness $\neq$ robustness

Robustness, approaches

- exact computing
  [Karasick et al], [Mehlhorn et al], [Yap et al], [Brönnimann et al]; speedup: floating point filters [Fortune-Van Wyk], [Shewchuk]; symbolic perturbation schemes [Edelsbrunner-Mücke], [Yap], [Emiris-Canny-Seidel]; leave in the degeneracies [Burnikel-Mehlhorn-Schirra], ...

- fixed precision approximation
  [Greene-Yao], [Guibas et al], [Fortune], [Milenkovic], [Sugihara et al], ...

Outline of Part II

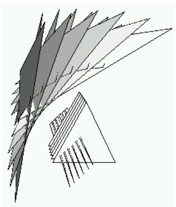
- difficulties
- exact computing
- fixed precision approximation
  - rounding
  - perturbation
- the CGAL library and its arrangements package

Exact computing

- solves the precision problem; implements the ideal computational model
- how is it done?
- for rationals there is a straightforward (but still costly) solution
- line intersection demo, take II
- GMP, LEDA, CORE

Exact rationals

- sometimes the numbers can be quite big
  example: arrangements of triangles

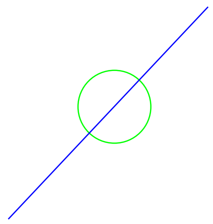


- can we speed up the process?
- predicates vs. constructions
- predicate filters

```
T1:
x= 28027/25243, y= 43613/18457, z= 14423/37273
x= 20353/2617, y= 26497/32299, z= 3673/63667
x= 55897/42403, y= 499/27767, z= 31253/10243
T2:
x= 53593/24763, y= 62501/63317, z= 11827/5693
x= 57143/65423, y= 40483/59447, z= 27739/62327
x= 57283/22027, y= 41231/45817, z= 9433/48673
T3:
x= 5693/48527, y= 11597/7757, z= 58367/44017
x= 2377/59471, y= 23831/3163, z= 57287/25343
x= 16657/46507, y= 57283/14783, z= 9437/6911
Intersection:
x (normalized rational)=
2974289381842164777454668853064172074327166148258603101344807660820042
0261669891047833223434838758837012 / 831908664650212786986421527747500
8390259862075297035835614133471677136475796012631653754008326934339751
```


Beyond rationals

- the intersection of the line $y = x$ with the circle $x^2 + y^2 = 1$



- $(\sqrt{2}/2, \sqrt{2}/2), (-\sqrt{2}/2, -\sqrt{2}/2)$
- what does it mean to compute it exactly?
- how can we tell if another curve passes thru such an intersection point?

Exact algebraic numbers

- the underlying idea: separation bounds
- for an algebraic expression E, a function $\sigma(E)$ such that the value of a non-zero E is lower bounded by $\sigma(E)$
- expression DAGs
- approximating E with "big float" till we can decide definitively whether zero or not
- LEDA real, CORE Expr
- exact arrangements of curves: CGAL, conix/cubix (Exacus, MPI)

Exact computing: pros and cons

pros

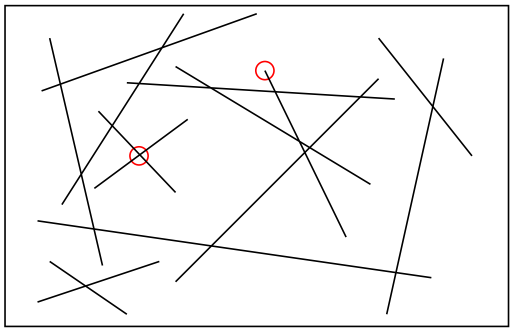
- the truth
- algorithms can be easily transcribed, up to the general position assumption (this is the CGAL approach)

cons

- requires special machinery (non-standard number types); available only for limited types of arrangements
- ever improving but still slow compared with machine arithmetic
- exact numerical output may be huge, when at all possible

Rounding arrangements

- transforming an arbitrary precision arrangement into a fixed precision representation

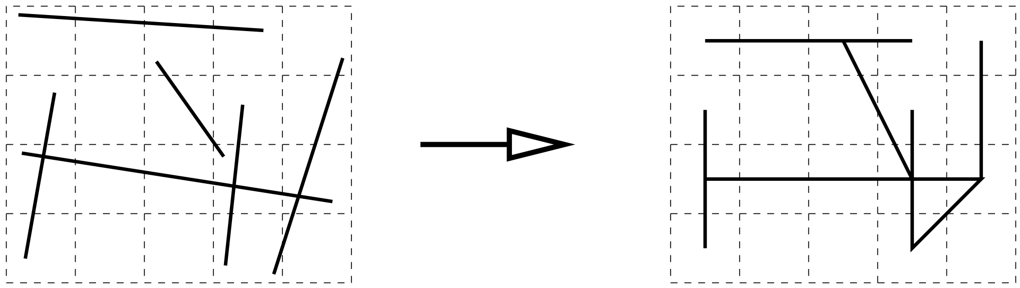


- why round? (1) huge numbers, (2) no finite numerical representation
- is it OK to round?

Issues in rounding

- the representation of the rounded arrg
- what is the relation between the exact arrg and the rounded arrg: geometry, topology/combinatorics
- how does the rounding serve the application?

Snap rounding

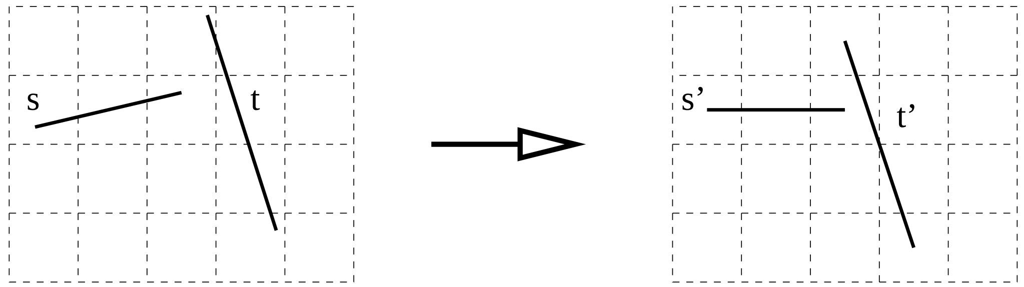


- topology: features may collapse but a curve does not cross over a vertex
- approximation: the rounding chain is in the Minkowski sum of the original segment and a pixel centered at the origin

Snap rounding: A brief history

- [Greene+Yao ’86] rounding polygonal subdivisions
- [Greene], [Hobby ’99] SR arrgs of segments
- [Guibas+Marimont ’98] dynamic SR
- [GGHT ’97] output sensitive SR, segments in 3D
- [Milenkovic ’00] rounding with shortest paths
- also in 3D: [Fortune ’99], [Milenkovic ’90]

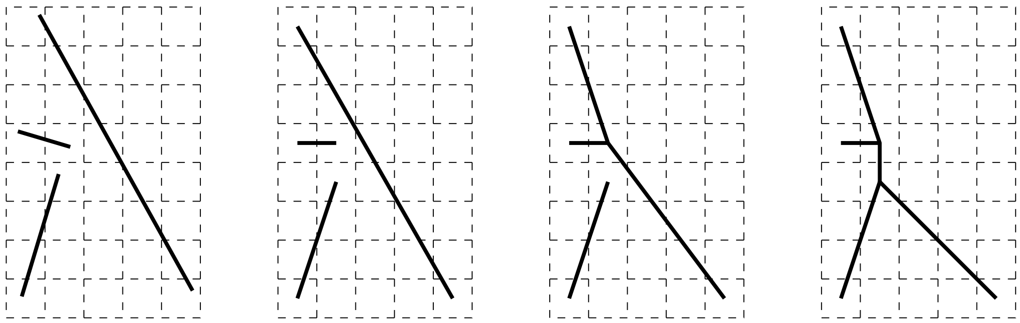
The distance between a vertex and a non-incident edge



$1/\sqrt{(2^b-1)^2+1} \approx 2^{-b}$, where b is the number of bits per coordinate

Iterated snap rounding

[H-Packer 02]

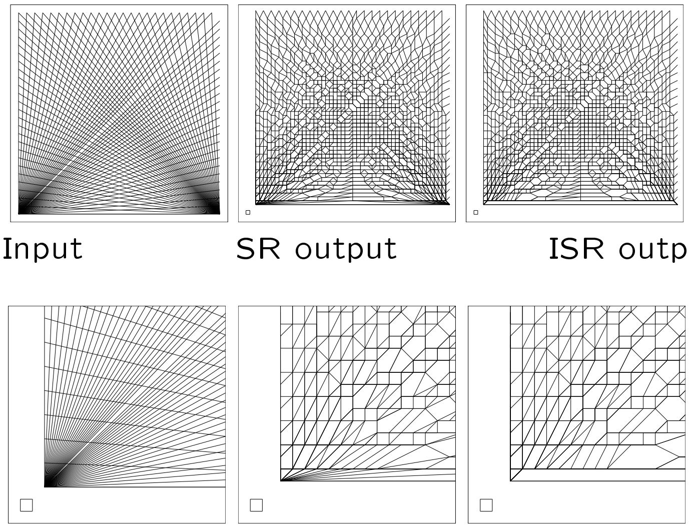


a simple algorithm with an efficient implementation (in CGAL)

Iterated snap rounding: Properties

- a finite process
- ISR = a finite number of rounds of SR
- no new hot pixels created
- preserves the topology in the same sense that SR does
- a vertex is at least half a unit away from any non-incident edge
- the rounding chain is in the Minkowski sum of the original segment and a square of side size k (= depth of the recurrence) centered at the origin

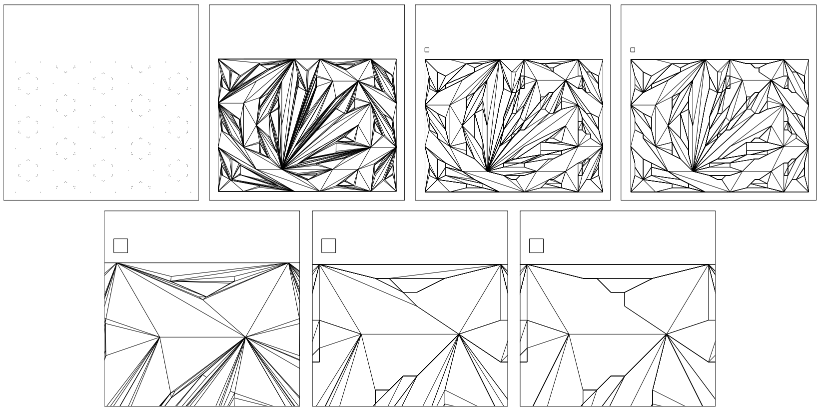
SR vs. ISR: example 1



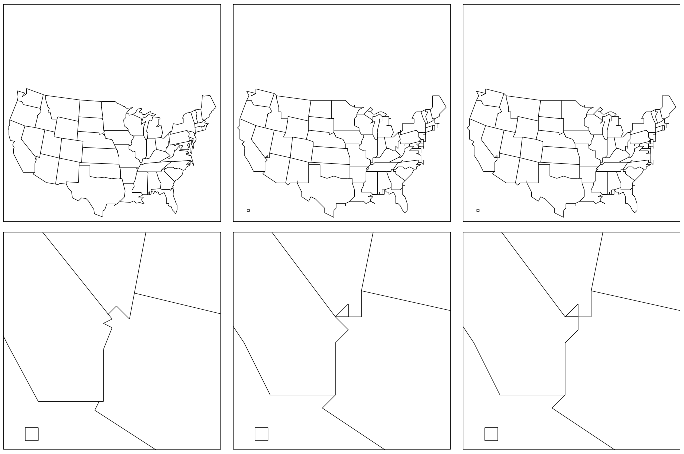
Input SR output ISR output

Input zoom in SR output zoom in ISR output zoom in

SR vs. ISR: example 2



SR vs. ISR: example 3



Snap rounding: pros and cons

pros

- convenient numerical output
- fairly efficient
- preserves certain topological properties

cons

- very limited range of effective applicability
- requires special machinery for preprocessing (is it necessary?)

major challenge

- devise consistent and efficient rounding schemes for planar arrangements of curves and for arrangements of surfaces in higher dimensions

Controlled perturbation

input: a set $\mathcal{C}$ of geometric objects (curves or surfaces), and the floating point precision

goal: perturb $\mathcal{C}$ slightly, $\mathcal{C} \Rightarrow \mathcal{C}'$, such that

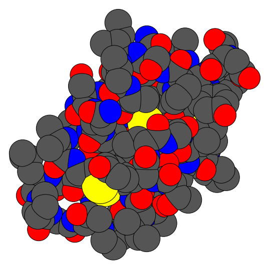
- all the predicates arising in the construction of $\mathcal{A}(\mathcal{C}')$ are computed accurately, and
- $\mathcal{A}(\mathcal{C}')$ is degeneracy free

Controlled perturbation: preliminaries

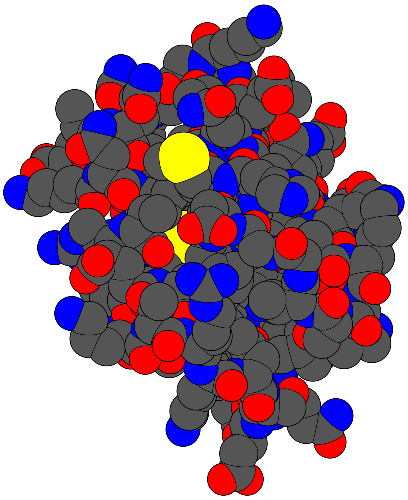
- reminder: a fixed precision approximation
- resolution bound ε, perturbation bound δ (actual perturbation)
- degeneracy := potential degeneracy
- no degeneracy $\Rightarrow$ no perturbation
- otherwise identify and remove all degeneracies
- predicates are accurately computed
- trade-off between perturbation magnitude and computation time

Controlled perturbation, types of arrangements

- spheres in space, molecular modeling [H-Shelton]
- polyhedral surfaces, swept volumes [Raab-H]
- polygons [Packer]
- circles [H-Leiserowitz]

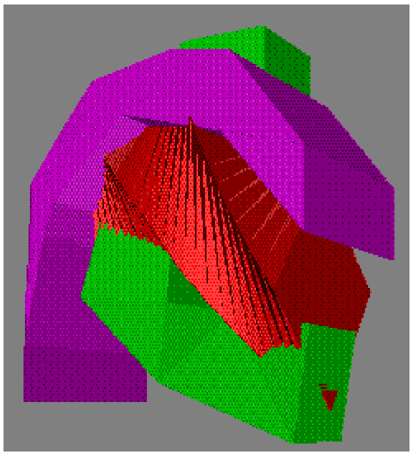


Controlled perturbation, molecular surfaces



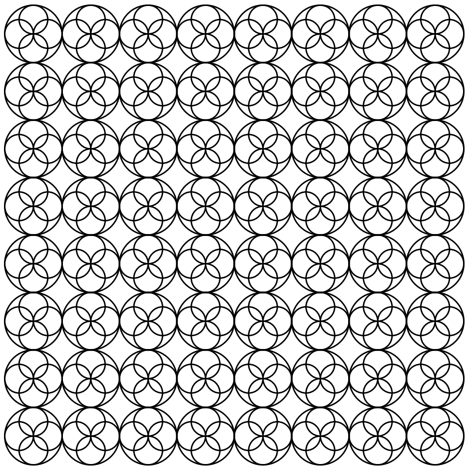
- sparse arrangement
- approximate model

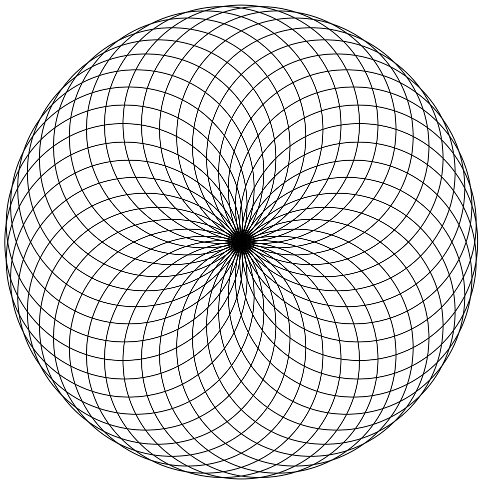
Controlled perturbation, approximate swept volumes



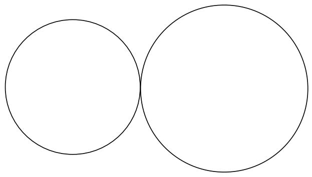
- arrangement of polyhedral surfaces
- approximation

Arrangements of circles





Resolution bound, example



$$[(X_1 - X_2)^2 + (Y_1 - Y_2)^2]^{\frac{1}{2}} = R_1 + R_2$$

$$E = (X_1 - X_2)^2 + (Y_1 - Y_2)^2 - (R_1 + R_2)^2$$

outer tangency $\equiv E = 0$

the minimum distance to move a circle so that the predicate will be safely evaluated to a non-zero value (using fp)

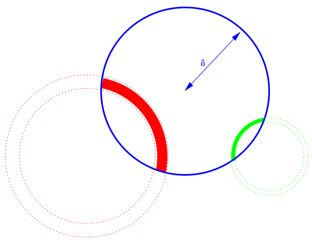
The scheme

- input: $C_1, C_2, \ldots, C_n$ by center coordinates and radii, floating-point precision p
- compute ε, δ
- handle the circles one by one, $C_i \Rightarrow C'_i$
- $\mathcal{C}'_i = \{C'_1, \ldots, C'_i\}$, at the end of stage i, $\mathcal{A}(\mathcal{C}'_i)$ is degeneracy free and C'_i will not be moved again
- if C_{i+1} does not induce any degeneracy with $\mathcal{C}'_i$ then $C'_{i+1} := C_{i+1}$, otherwise

Handling the current circle

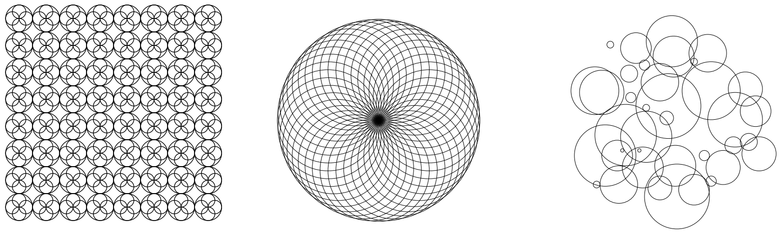
given the resolution bound ε the circle C_{i+1} will be moved by at most $\delta(\varepsilon, \cdot)$ from its original position such that no two 'features' will be less than ε apart

the new location of the center is chosen inside a δ-disc around the original center avoiding forbidden regions

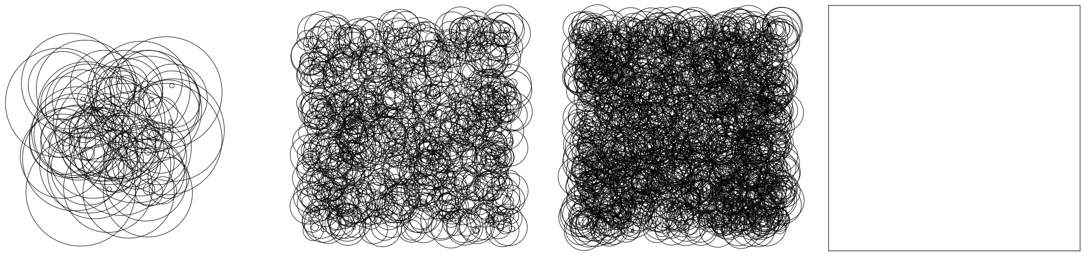


the arrangement of forbidden regions is more complicated than the original arrangement, use randomization

Experiments



grid flower rand_sparse



rand_100 rand_1000 rand_2000 rand_10000

Experiments, cont’d

on an Intel Pentium III 1 GHz with 2 GB RAM (Linux Redhat 7.3, gcc 2.95.3)

name	n	max radius	max coord	avg pert	max pert	time
grid	320	10	140	0.1122	0.6320	0.114
flower	40	100	100	0.8819	2.3360	0.132
rand_sparse	40	20	100	0.0424	0.0493	0.002
rand_100	100	49	100	0.0597	0.4017	0.130
rand_1000	1000	100	1000	0.0497	0.3994	0.556
rand_2000	2000	100	1000	0.1815	1.0856	2.804
rand_10000	10000	35	1000	0.3412	1.4527	9.478

Name	#vertices	#halfedges	#faces
rand_10000	346954	1388506	347301

Controlled perturbation, summary

- a fixed precision approximation method, actual (not symbolic) perturbation; justified in many applications
- guarantees robustness while using floating-point arithmetic
- about 40 times faster than state-of-the-art exact arithmetic
- separation bound for same size input numbers requires ≈ 900 bits
- no degeneracies $\Rightarrow$ no perturbation
- otherwise removes all degeneracies (good for exact computation as well)
- easy to program (less easy to analyze)

Further work on controlled perturbation

- improve the bounds
- dynamic bounds
- moving objects (molecular motion simulation)

Implementing Computational Geometry algorithms

- C++gal (INRIA), PlaGeo, SpaGeo (Utrecht), LEDA-Geometry (MPI Saarbrr̈cken), [XYZ GeoBench (Zurich), . . .]
- large effort, requires unique expertise and more research
  - 1995: the CGAL kernel
  - 1996: the official beginning of CGAL
  - 1998: GALIA, a continuation of CGAL
  - 2001: ECG
  - CGAL development still goes on

CGAL = Computational Geometry Algorithms Library

- ETH Zurich
- FU Berlin
- Trier University
- INRIA Sophia Antipolis
- MPI Saarbrücken
- Tel Aviv University
- Utrecht University

Tel Aviv contributors (arrgs): Eyal Flato, Iddo Hanniel, Sigal Raab, Shai Hirsch, Efi Fogel, Ron Wein, Oren Nechsushtan, Eti Ezra, Tali Zvi, Eran Leiserowitz, Eli Packer

CGAL: major design goals

- robustness
- generality, flexibility
- efficiency

CGAL: structure

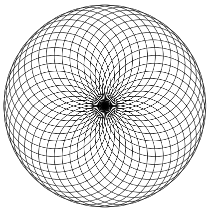
- kernel: primitive geometric objects and predicates
- basic library: geometric data structures and algorithms such as convex hulls, triangulations, kd-trees, planar maps and arrangements
- support library: I/O, visualization, random generators

manuals, “getting started”, developers manual

supported on a variety of compilers and operating systems

Maps and arrangements in CGAL

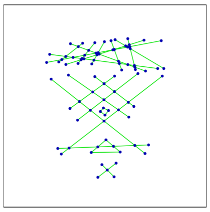
- (i) topological maps, (ii)planar maps, (iii) planar maps with intersections, (iv) arrangements

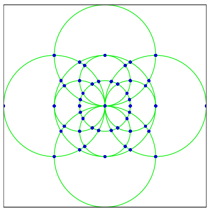


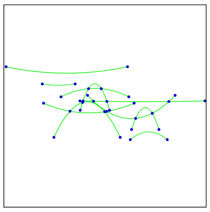
- services: map construction (incremental, sweep), traversal over the entire map, over the edges around a face or a vertex, point location, envelopes
- representation: DCEL (doubly connected edge list), trapezoidal decomposition search structure

The separation of combinatorics and numerics

- traits classes: basic objects and operations on them







- objects: point, curve
- example predicate: does a point lie below / on / above an x-monotone curve
- predefined “traits”: segments (rational, filters), conics, polylines, faster seg-circle, seg-hyperbola

Bibliography

- Robustness and precision issues in geometric computation, Schirra, in Elsevier Handbook (Sack-Urrutia), 2000
- Robust geometric computation, Yap, in CRC Handbook (Goodman-O'Rourke), 1997, 2nd edition forthcoming
- The LEDA Platform of Combinatorial and Geometric Computing, Melhorn-Näher, Cambridge University Press, 1999, www.mpi-sb.mpg.de/LEDA/leda.html
- Robust geometric computing in motion, Halperin, International Journal of Robotics Research, 2002, robustness,CGAL, arrangements and motion planning

Websites

- www.cgal.org
- www.mpi-sb.mpg.de/LEDA
- www-sop.inria.fr/prisme/ECG Effective Computational Geometry for Curves and Surfaces
- www.cs.nyu.edu/exact/core
- http://www.mpi-sb.mpg.de/projects/EXACUS

THE END of PART II