Robust Geometric Computations
(plus some other graphics stuff)

John Keyser
Department of Computer Science
Texas A&M University
Robustness Issues

• Robustness problems: failure of assumption

• Two main sources in geometric computation:
  – Numerical Error
  – Degeneracies
Numerical Error

• Numerical computations are exact, unit time

• Sources:
  – Initial approximations
  – Intermediate calculations
  – Magnified greatly
Degeneracies

- General Position
  - Small perturbations won’t change nature of interaction

- Intentional and Accidental

- Numerical error creates/removes
Why is Robustness an Issue in Geometry?

• Geometric problems expressed as predicates
  – e.g. Is a point above or below a line?
• Predicate results determine branching in a program (degeneracies are the “=“ branch)
• When wrong branch is taken, objects can be inconsistent
Importance of Accuracy

Input Primitives

Intersection Curves in Patch Domain
Why is Robustness Worth Working On?

- Robustness problems have become the major issue in need of improvement
  - Speed is less of a factor with faster computers
  - Robustness (in this sense) does not increase (much) with improved technology
  - While user training might address most problems, this is expensive and unreliable
Does Anyone Actually Care About Robustness?

• In particular, model exchange is a major concern
  – Estimate from a few years ago: model exchange problems cost US Auto industry > $1 billion per year!
  – There’s much more going on here than just geometric robustness, but that’s a major role
  – Standards help but don’t solve
Methods for Dealing with Numerical Error

• Tolerancing – easy to incorporate
  – e.g. if $A-B < \varepsilon$, set $A=B$
  – Non-transitive: $A=B$, $B=C$, $A\neq C$
  – Confusion: tolerancing to deal with numerical error vs. tolerancing in design

• Interval Arithmetic
  – Keep each number as an interval, guaranteed to bound the “real” number
  – Intervals grow with operations, can eventually become unusable.
Methods for Dealing with Numerical Error

• Exact Computation (my favorite…)
  – Take some interpretation of data, then compute exactly (as much precision as necessary)
  – Original interpretation could be bad
  – Can be very slow/memory intensive
  – Does not solve model transfer issue
Methods for Dealing with Degeneracies

• Case by case basis
  – List all possible degeneracies, find ways to detect them, then apply some “fix”
  – Lots of work/cases – easy to make mistakes or miss certain cases
  – Usually spend far more time (especially coding) handling the exception rather than the rule.

• Perturbation Methods
  – More general, but tougher to put into practice
  – Modify data by small (symbolic) amount, and compute on modified data.
Broad Robustness Approaches

• Take a particular interpretation of the data
  – This interpretation might be wrong
  – Might require “cleaning up” object first
  – But, all computations can be done, using “standard” geometric algorithms

• Represent things “fuzzily”
  – Fuzziness might be so great that decisions are useless
  – But, when it works, you know you have good results
Some Recent Directions

- Formalized definitions
- Algebraic computation
- Determining meaningful description of error
  - What can you say if you know you compute with some error
- Exchange-oriented work: e.g. reconstructing design history
My Work

- Exact computation, particularly for solid modeling applications

- Theme:
  - Ensure correctness, then work to improve performance vs.
  - Get an efficient implementation, then make it more robust
Exact Computation

• For Solid Modeling, common operations involve intersecting curves and surfaces.

• Most operations can be phrased as algebraic computations.

• So, exact geometric computation often boils down to computer algebra techniques.
Exact Geometric Representations

• Represent surfaces with arbitrary-length coefficients
• Intersections of surfaces become (non-parametric) curves
• Intersections of curves are algebraic points
  – Represent using algebraic numbers,
    e.g. “root of $x^2 - 2 = 0$ in the range $[1,2]$”
Exact Geometric Representations
Rational Univariate Reduction

• Given $m$ polynomials in $n$ variables
  – Rational coefficients
• Determine all roots of system by finding a set of univariate polynomials:
  
  \[
  h(x): \quad \text{minimal polynomial}
  \]
  
  \[
  h_1(\alpha), h_2(\alpha), ..., h_n(\alpha) \quad \text{coordinate polynomials}
  \]
  
  ($\alpha$ is a root of $h(x)$)
Exact Solid Modeling

• RUR can be used as an alternative representation for points.
  – Works better around degeneracies

• Numerical perturbation techniques for dealing with degeneracies

• Speedup techniques
Results

- Can compute boundary for solids that crash earlier modelers due to degeneracies and near-degeneracies
Future/Ongoing Work in Robust Solid Modeling

- Representing non-algebraic objects
- Incorporating more recent algebraic geometry advances
- Hybrid systems/computation
- Geometric filtering
- More stable modeling paradigms
- Determining user intent
Other Graphics Work

• Physically-based simulation
Other Graphics Work

- Physically-Based Simulation
Other Graphics Work

- Terrain representation, modeling
Other Graphics Work

• GPU-based geometry calculations
Other Graphics Work

- Brain Networks Lab – Visualization
Questions?

• Thanks for listening