

Graphics Interface'96

**Surface intersection using
affine arithmetic**

Luiz Henrique de Figueiredo

Computer Systems Group
Department of Computer Science
University of Waterloo

May 1996

Introduction

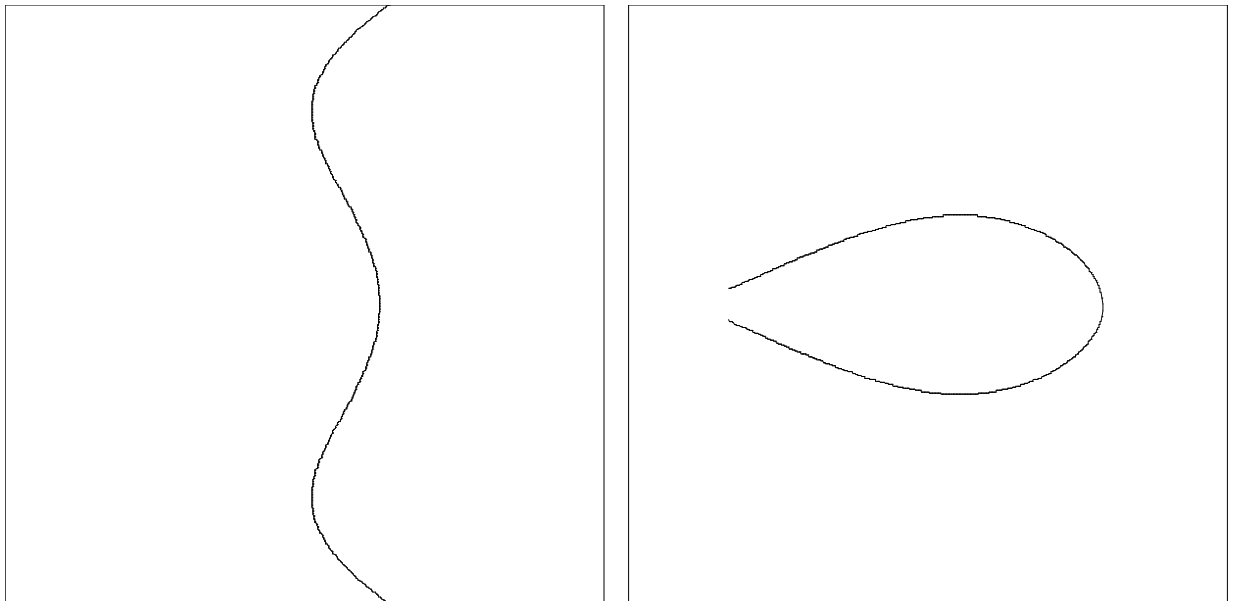
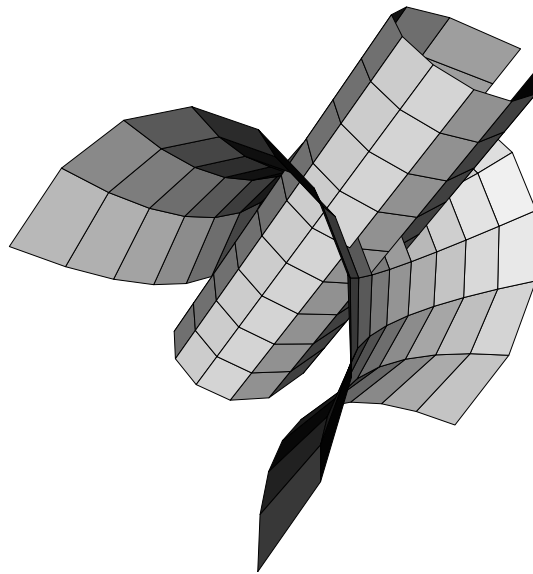
- Parametric surfaces very popular in CAGD
 - ◇ easy to approximate and render
 - ◇ many special classes suitable for design

- Parametric surfaces in CSG modeling
 - ◇ requires robust methods for intersection

- Computing surface intersection
 - ◇ continuation (local)
 - ◇ decomposition (global)

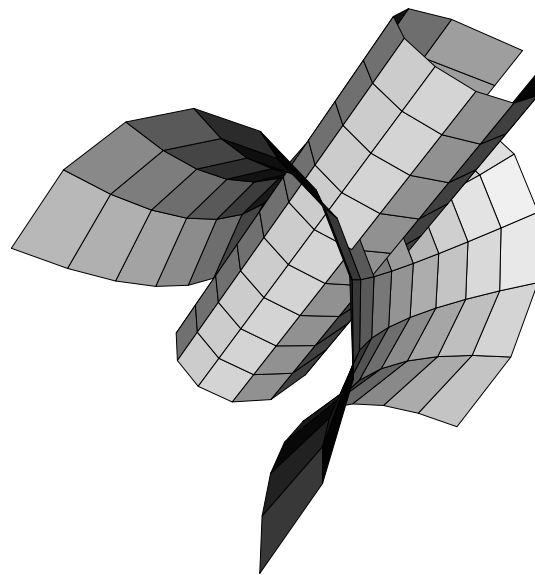
Continuation methods

- Find starting point
- March along intersection curve
- Map curve back for trimming



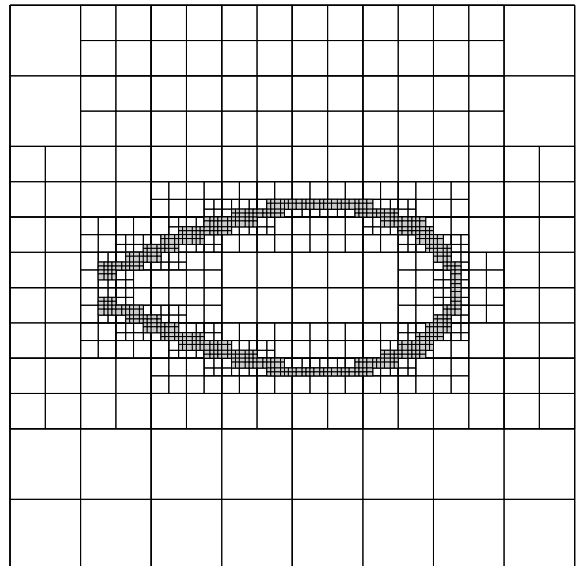
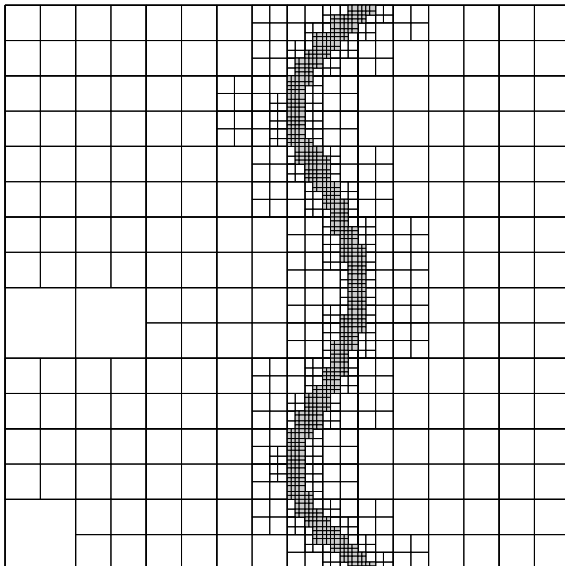
Decomposition methods

- Naive method
 - ◇ uniform polygonal approximations
 - ◇ too expensive to refine
- Adaptive decomposition methods
 - ◇ adaptive polygonal approximations
 - ◇ not general
- Global decomposition method
 - ◇ quickly discard non-intersecting patches
 - ◇ range analysis (Gleicher & Kass, GI'92)



Gleicher-Kass algorithm (GI'92)

- Quadtree domain decomposition
- Compute bounding boxes for patches
- If bounding boxes do not intersect
patches cannot intersect
- If bounding boxes do intersect
patches may intersect
subdivision + recursion
- Keep track of patches that might intersect



Interval arithmetic

- Quantities represented by intervals of floating-point numbers

$$\bar{x} = [a .. b] \Rightarrow x \in [a .. b]$$

- Primitive operations on intervals

$$[a, b] + [c, d] = [a + c, b + d]$$

$$[a, b] - [c, d] = [a - d, b - c]$$

$$[a, b] \times [c, d] = [\min(M), \max(M)]$$

$$[a, b] / [c, d] = [a, b] \times [1/d, 1/c]$$

$$M = \{ac, ad, bc, bd\}$$

- Similar formulas for sin, cos, log, exp, ...
- Automatic interval extensions

$$z = f(x_1, \dots, x_n)$$

$$\bar{z} = \bar{f}(\bar{x}_1, \dots, \bar{x}_n)$$

$$x_i \in \bar{x}_i \Rightarrow z \in \bar{z}$$

- ◇ elegant with operator overloading

Interval arithmetic

- Provides robust estimates
- Easier than Lipschitz bounds

$$|f(x) - f(y)| \leq L|x - y|$$

$$L = \max |f'(x)|$$

- Too conservative
- Assumes operands are independent

$$\bar{x} = [4 .. 6]$$

$$10 - \bar{x} = [10 .. 10] - [4 .. 6] = [4 .. 6]$$

$$\bar{x}(10 - \bar{x}) = [16 .. 36]$$

$$\text{exact range} = [24 .. 25]$$

- Error explosion in long computations
 - ◇ centered forms
 - ◇ affine arithmetic (Comba & Stolfi, '93)

Affine arithmetic

- Quantities represented by affine forms

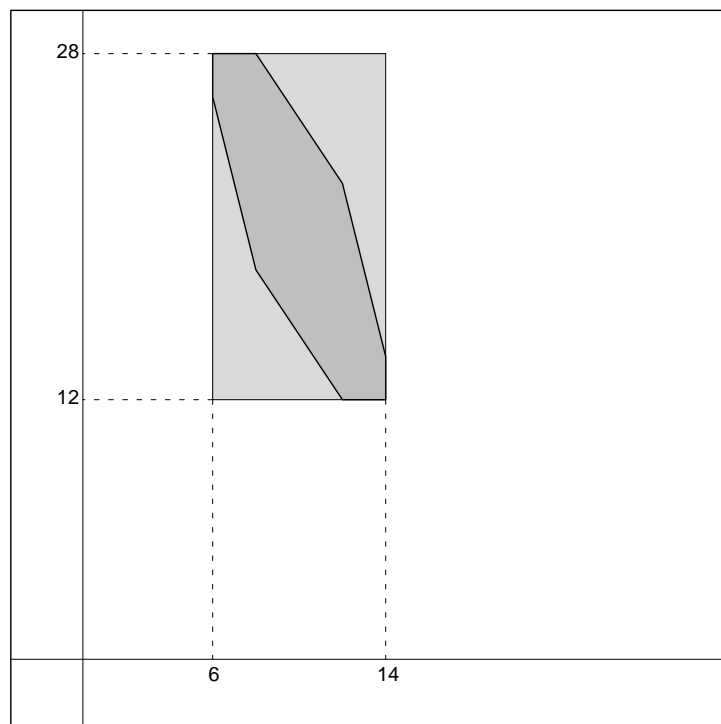
$$\hat{x} = x_0 + x_1\varepsilon_1 + x_2\varepsilon_2 + \cdots + x_n\varepsilon_n$$

$$\varepsilon_i \in [-1 .. 1]$$

- Simple conversion IA \leftrightarrow AA
- Keeps track of *correlations*

$$\hat{x} = 10 + 2\varepsilon_1 + 1\varepsilon_2 - 1\varepsilon_4$$

$$\hat{y} = 20 - 3\varepsilon_1 + 1\varepsilon_3 + 4\varepsilon_4$$



Computing with affine arithmetic

- Compute $z \leftarrow f(x, y)$ as $\hat{z} \leftarrow \hat{f}(\hat{x}, \hat{y})$
- Simple for affine functions

$$\hat{x} \pm \hat{y} = (x_0 \pm y_0) + \cdots + (x_n \pm y_n)\varepsilon_n$$

$$\alpha\hat{x} = (\alpha x_0) + (\alpha x_1)\varepsilon_1 + \cdots + (\alpha x_n)\varepsilon_n$$

$$\hat{x} \pm \alpha = (x_0 \pm \alpha) + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n$$

- In general, use best affine approximation and append extra term to represent error

$$\hat{z} = z_0 + z_1\varepsilon_1 + \cdots + z_n\varepsilon_n + z_k\varepsilon_k$$

- Example revisited

$$\hat{x} = 5 + 1\varepsilon_1$$

$$10 - \hat{x} = 5 - 1\varepsilon_1$$

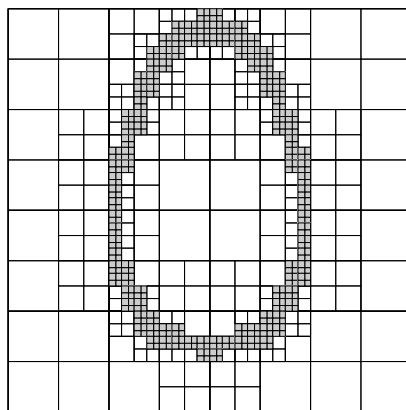
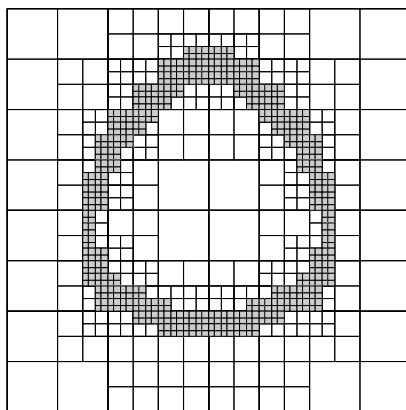
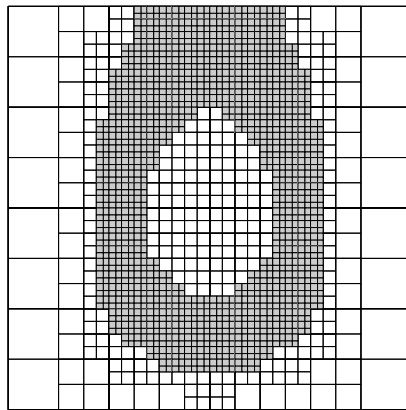
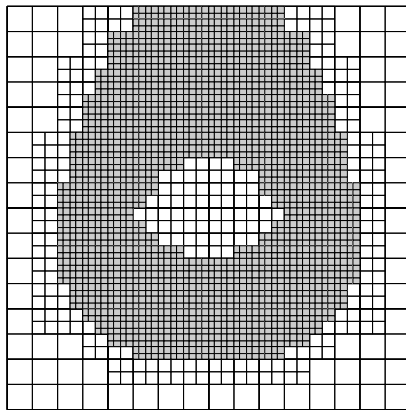
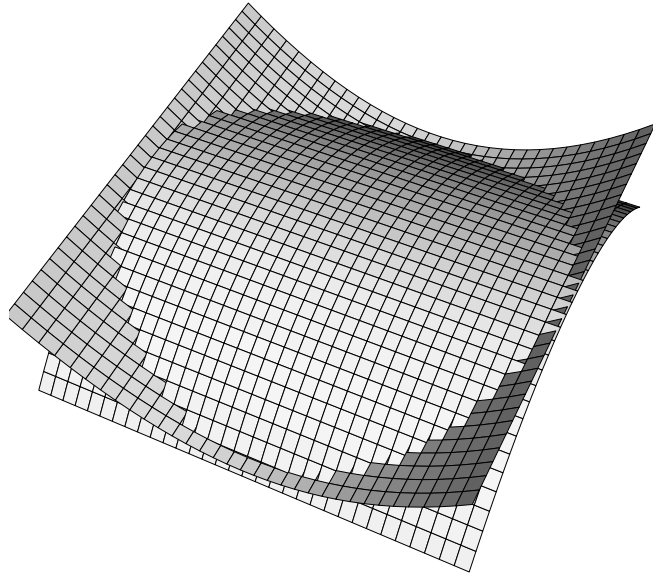
$$\hat{z} = \hat{x}(10 - \hat{x}) = 25 + 0\varepsilon_1 - 1\varepsilon_2.$$

$$[\hat{z}] = [24 \dots 26]$$

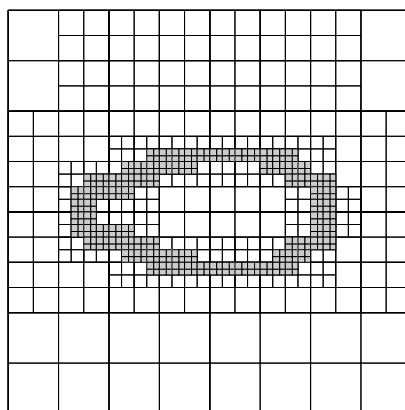
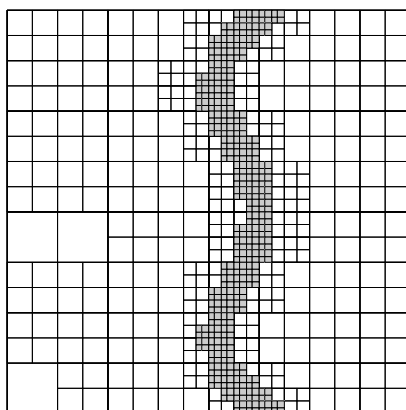
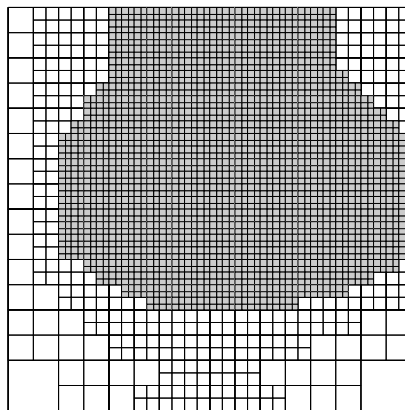
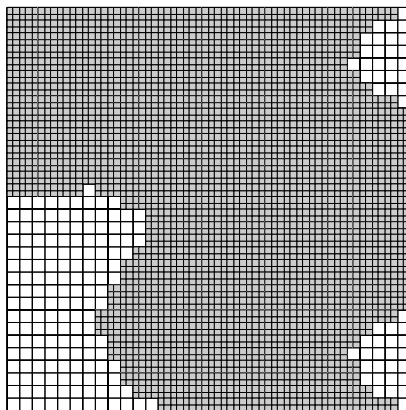
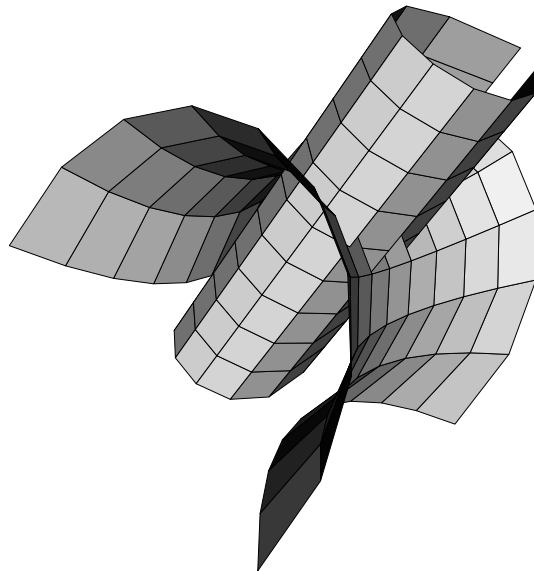
$$\text{exact range} = [24 \dots 25]$$

$$\text{IA result} = [16 \dots 36]$$

Example: Lofted parabolas



Example: Bicubic patches



Statistics for examples

- Lofted parabolas

	computed	remained	time
IA	5314	3360	3.0
AA	1930	968	1.0
IA/AA	2.8	3.5	3.0

- Bicubic patches

	computed	remained	time
IA	8038	5508	3.7
AA	1786	728	1.0
IA/AA	4.5	7.6	3.7

- AA more accurate than IA
- AA locally more expensive than IA
- AA globally more efficient

Conclusion

- Gleicher-Kass algorithm
 - ◇ robust
 - ◇ simple to implement
 - ◇ easy to replace IA with AA
- AA more complex and expensive than IA
 - ◇ AA primitives typically 4–5 times slower
- higher accuracy of AA worth extra cost
 - ◇ smaller quadtrees
 - ◇ overall faster
- future work
 - ◇ improve performance of AA
 - ◇ use AA in other problems that have IA solutions: raytracing