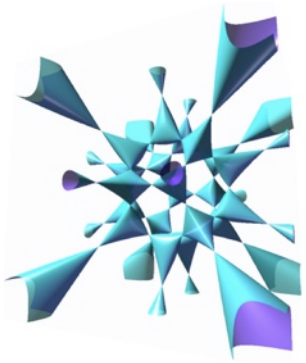
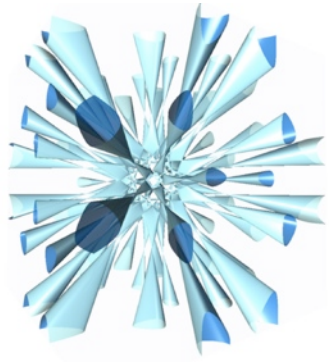
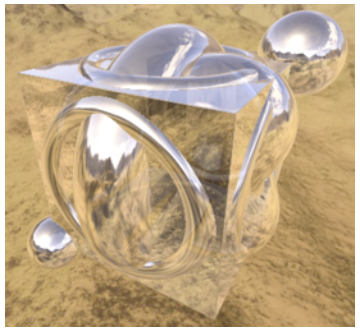
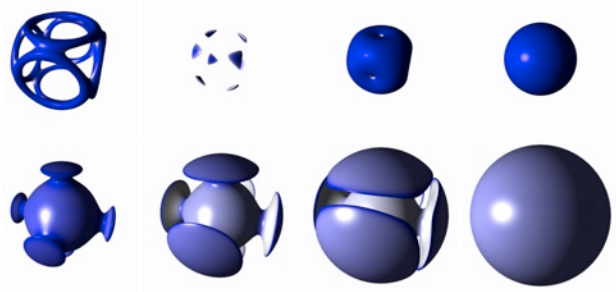
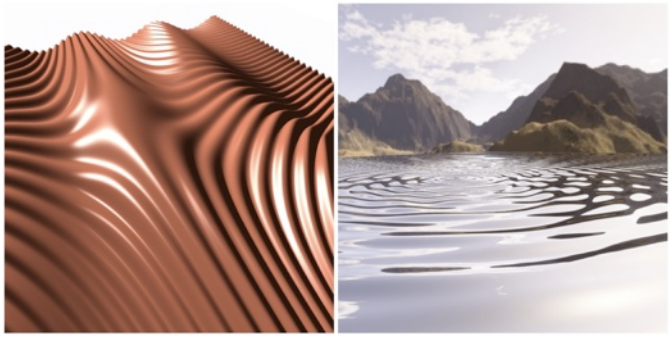





Deutsche Forschungsgemeinschaft
DFG

Interval arithmetic in visualization and computer graphics



Younis Hijazi

IGG, LSIIT, Université de Strasbourg
(Fraunhofer-ITWM Kaiserslautern)

Outline

- Motivation
- Interval Arithmetic (IA)
- Arrangement of arbitrary implicit planar curves
- Ray casting arbitrary implicits
- Conclusions & Future Work

Motivation

- Need of visualization
  - increasing amount of data & complexity of input
  - understand data
- Need of robust and efficient algorithms for large & unstructured data
  - elaborated mathematical tools
  - scalable and grid-free algorithms
- Why implicits?
  - flexible and compact model
  - well-suited for dynamic objects

Outline

- Motivation
- Interval Arithmetic (IA)
  - What is IA and Why IA?
  - Extensions of IA
- Arrangement of arbitrary implicit planar curves
- Ray casting arbitrary implicits
- Conclusions & Future Work

Interval Arithmetic - What is IA?

- Interval Arithmetic (IA) was introduced by Ramon Moore in the 60s
- Basic idea: represent both exact solution x and error ϵ within a single object, the interval I, i.e. $I = [x - \epsilon, x + \epsilon]$
- Definition:
  Let $a = [\underline{a}, \overline{a}]$ and $b = [\underline{b}, \overline{b}]$ be intervals.
  Then, if $op \in \{+, -, \times, \div\}$ we define:

$$S = S(a\ op\ b) = \{x\ op\ y \mid x \in a\ and\ y \in b\}$$

$$a\ op_{[]}\ b = [min\ S,\ max\ S]$$

- Example: $a + b = [\underline{a} + \underline{b}, \overline{a} + \overline{b}]$
- Interval analysis: study of interval algorithms

Interval Arithmetic - Why IA?

- IA is robust
  - it handles all kinds of error (rounding, truncation and input errors)
  - example: Suppose $x \in [-0.532, -0.531]$

    Then $$e^x \in 1 + [-0.532, -0.531] + \frac{[-0.532, -0.531]^2}{2}[0,1]$$

    $$= [0.468, 0.470] + \frac{[0.280, 0.284]}{2}[0,1]$$
    $$= [0.468, 0.470] + [0, 0.142]$$
    $$= [0.468, 0.612]$$

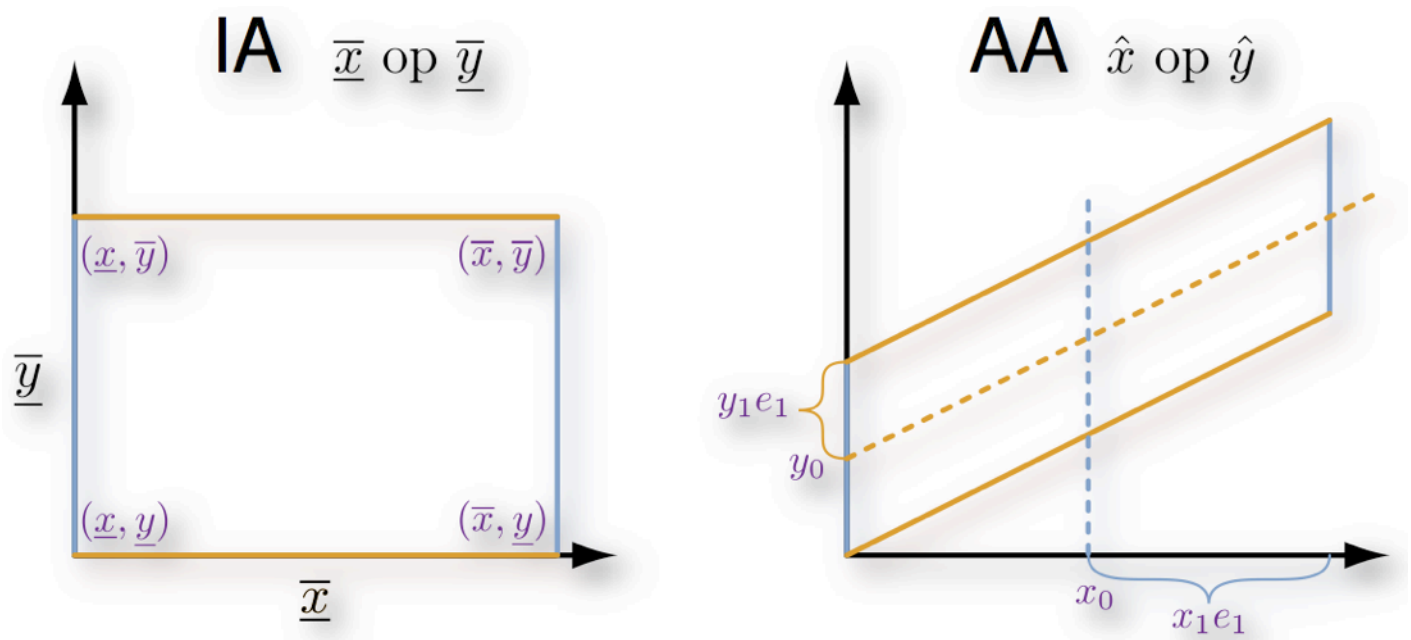
- IA is general
  - Moore's theorem: any computable function can be converted to an IA form (inclusion function)
  - it generalizes to higher dimensions (N-box)

Outline

- Motivation
- Interval Arithmetic (IA)
  - What is IA and Why IA?
  - Extensions of IA
- Arrangement of arbitrary implicit planar curves
- Ray casting arbitrary implicits
- Conclusions & Future Work

Interval Arithmetic - Extensions (1)

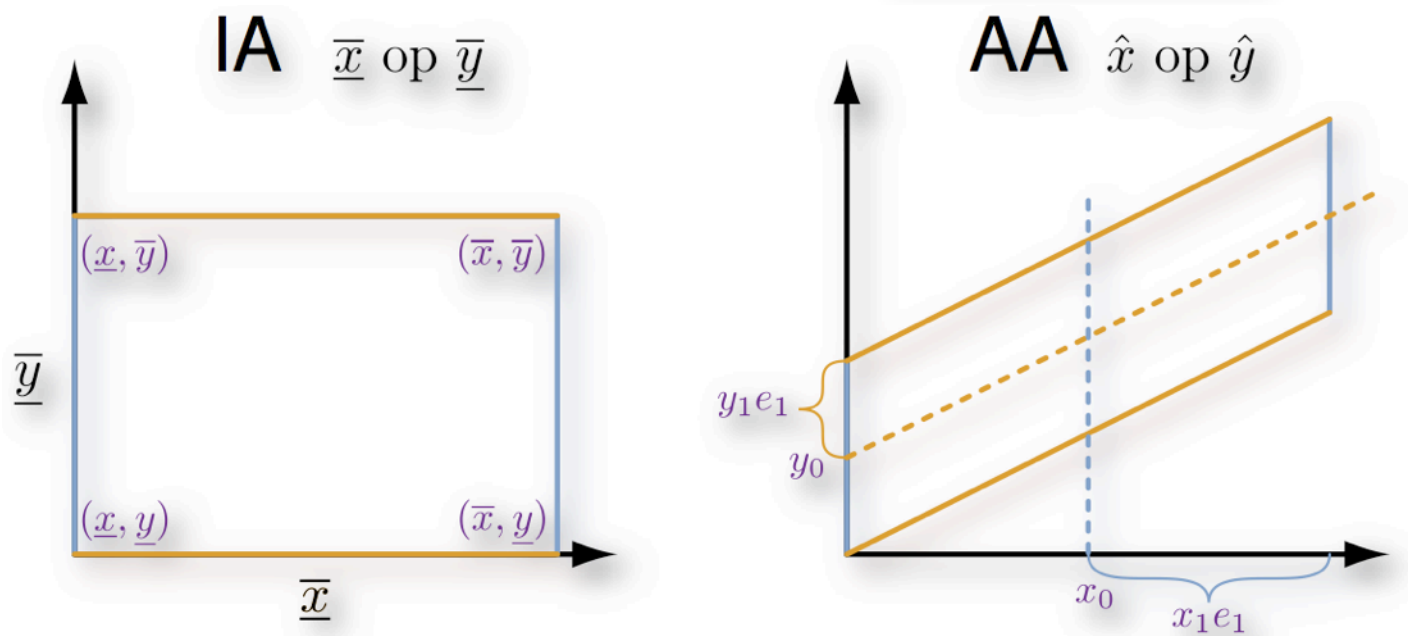
- [Stolfi et al] (90s): Affine Arithmetic (AA)
  - extends IA by keeping track of the correlation terms
  - overcomes IA's overestimation problem
  - more accurate than IA but much more expensive: need of trade-off between speed and accuracy



Interval Arithmetic - Extensions (2)

$$\hat{x} = x_0 + \sum_{i=1}^{n} x_i e_i$$

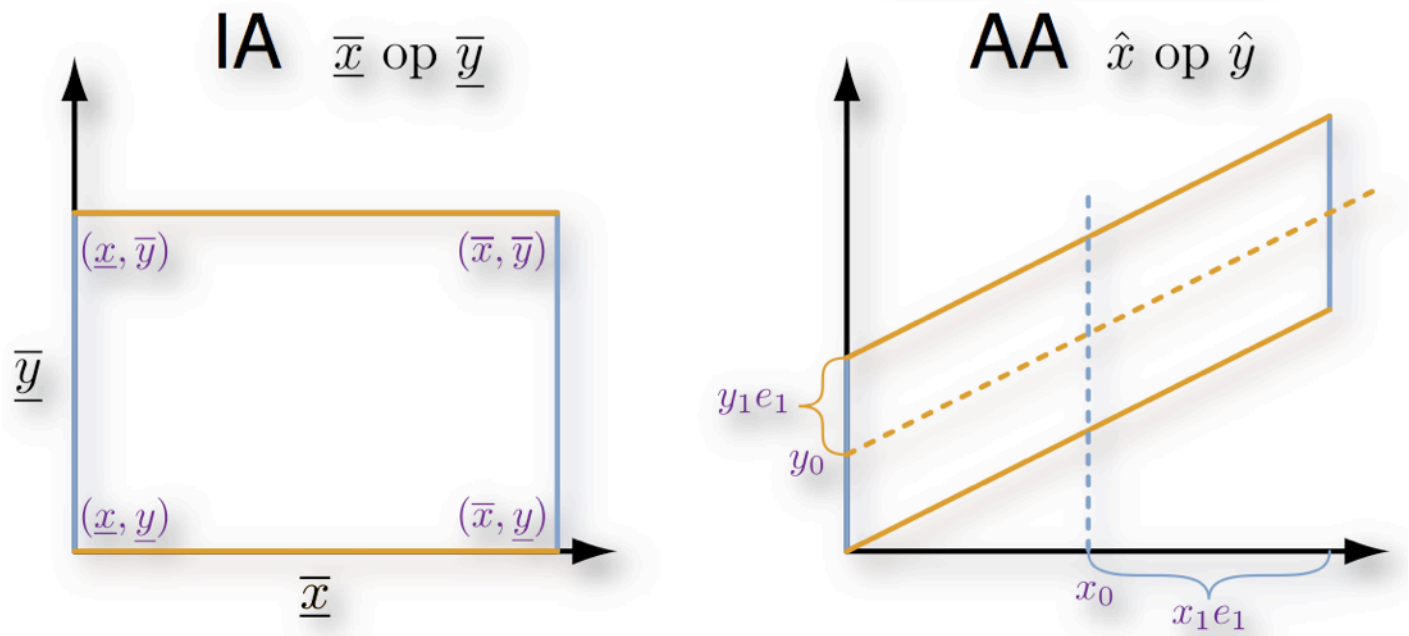
$$e_i \in [-1, 1]$$



Interval Arithmetic - Extensions (2)

$$\hat{x} = x_0 + \sum_{i=1}^{n} x_i e_i$$

$$e_i \in [-1, 1]$$

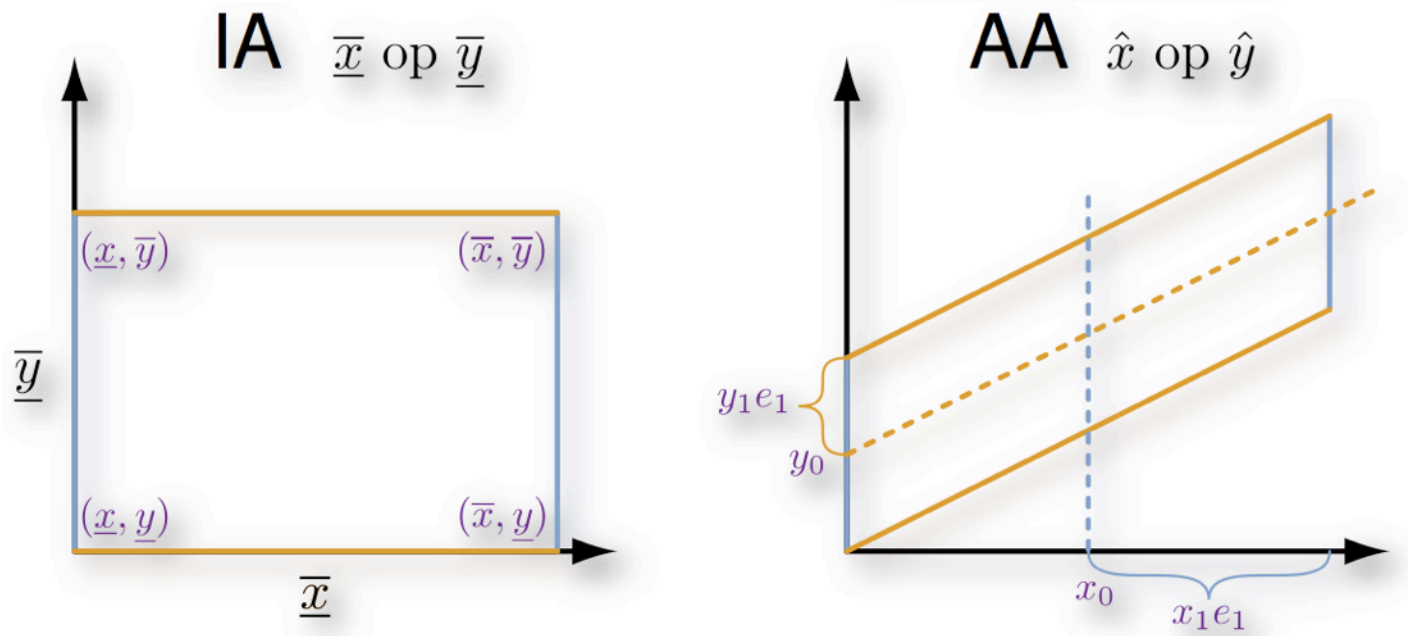


Interval Arithmetic - Extensions (2)

$$x_0 = (\overline{x} + \underline{x})/2$$
$$x_1 = (\overline{x} - \underline{x})/2$$
$$x_i = 0, \quad i > 1$$

$$\hat{x} = x_0 + \sum_{i=1}^{n} x_i e_i$$

$$e_i \in [-1, 1]$$

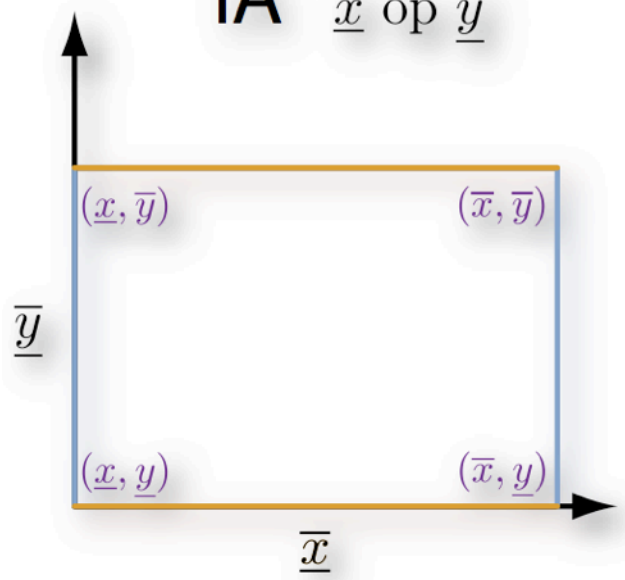


Interval Arithmetic - Extensions (2)

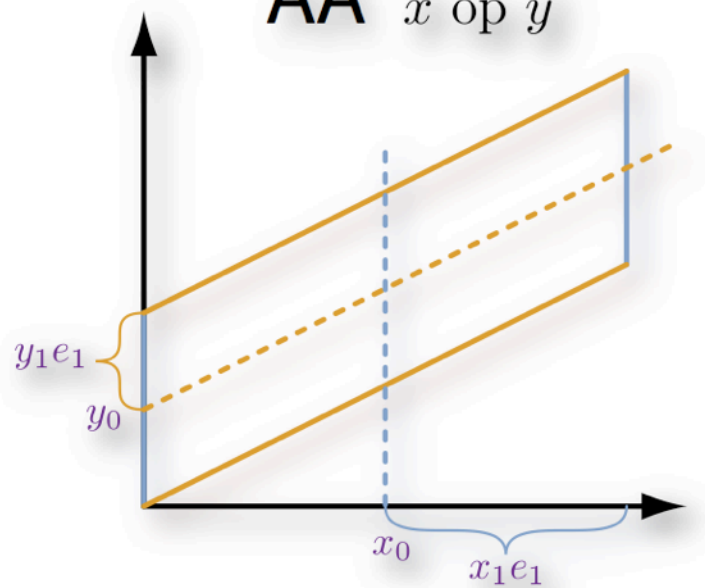
$$x_0 = (\overline{x} + \underline{x})/2$$
$$x_1 = (\overline{x} - \underline{x})/2$$
$$x_i = 0, \quad i > 1$$

$$\hat{x} = x_0 + \sum_{i=1}^{n} x_i e_i$$
$$e_i \in [-1, 1]$$

IA $\overline{x}$ op $\overline{y}$



AA $\hat{x}$ op $\hat{y}$



Interval Arithmetic - Extensions (2)

$$x_0 = (\bar{x} + \underline{x})/2$$
$$x_1 = (\bar{x} - \underline{x})/2$$
$$x_i = 0, \quad i > 1$$

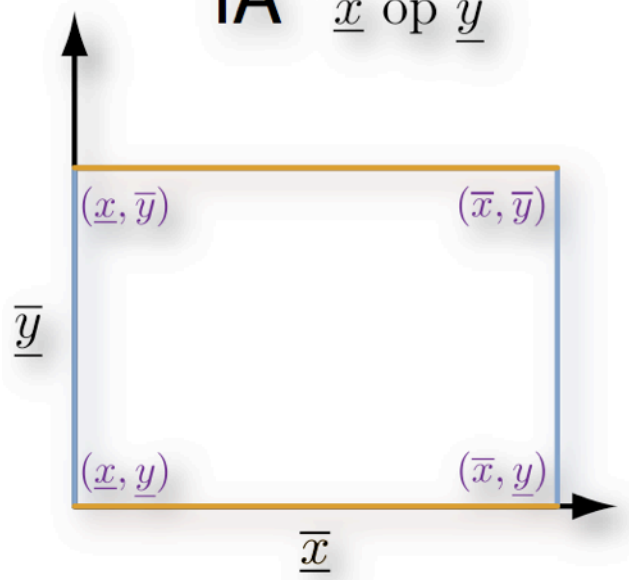
$$\underline{\bar{x}} = [x_0 - rad(\hat{x}), x_0 + rad(\hat{x})]$$

$$rad(\hat{x}) = \sum_{i=1}^{n} |x_i|$$

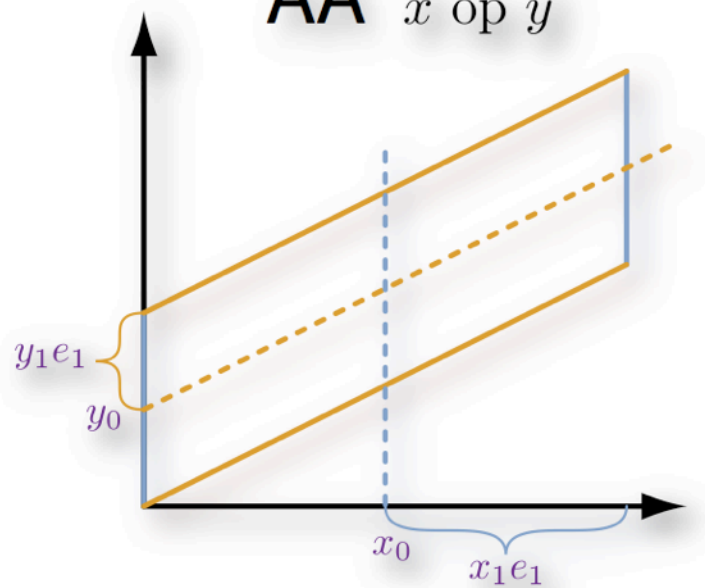
$$\hat{x} = x_0 + \sum_{i=1}^{n} x_i e_i$$

$$e_i \in [-1, 1]$$

IA $\underline{\bar{x}}$ op $\underline{\bar{y}}$



AA $\hat{x}$ op $\hat{y}$



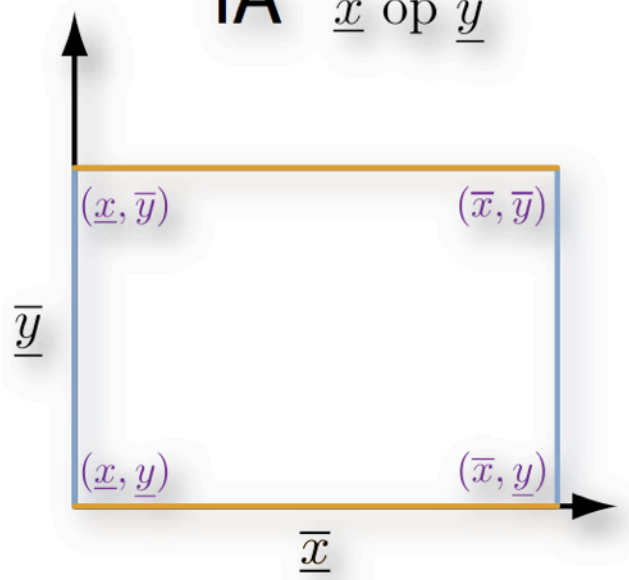
Interval Arithmetic - Extensions (2)

$$x_0 = (\overline{x} + \underline{x})/2$$
$$x_1 = (\overline{x} - \underline{x})/2$$
$$x_i = 0, \quad i > 1$$

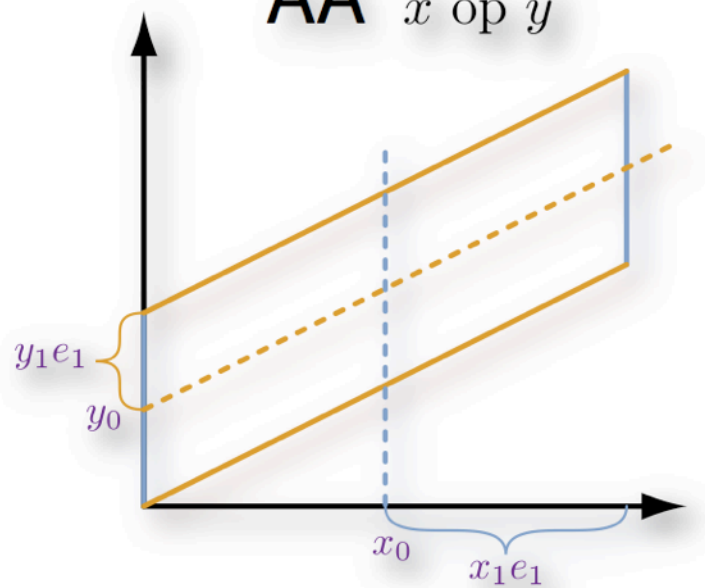
$$\underline{x} = [x_0 - rad(\hat{x}), x_0 + rad(\hat{x})]$$
$$rad(\hat{x}) = \sum_{i=1}^{n} |x_i|$$

$$\hat{x} = x_0 + \sum_{i=1}^{n} x_i e_i$$
$$e_i \in [-1, 1]$$

IA $\underline{x}$ op $\underline{y}$



AA $\hat{x}$ op $\hat{y}$



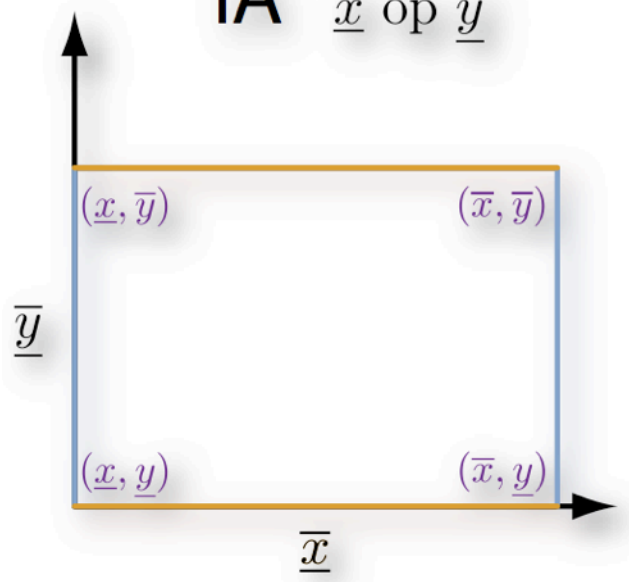
Interval Arithmetic - Extensions (2)

$$x_0 = (\overline{x} + \underline{x})/2$$
$$x_1 = (\overline{x} - \underline{x})/2$$
$$x_i = 0, \quad i > 1$$

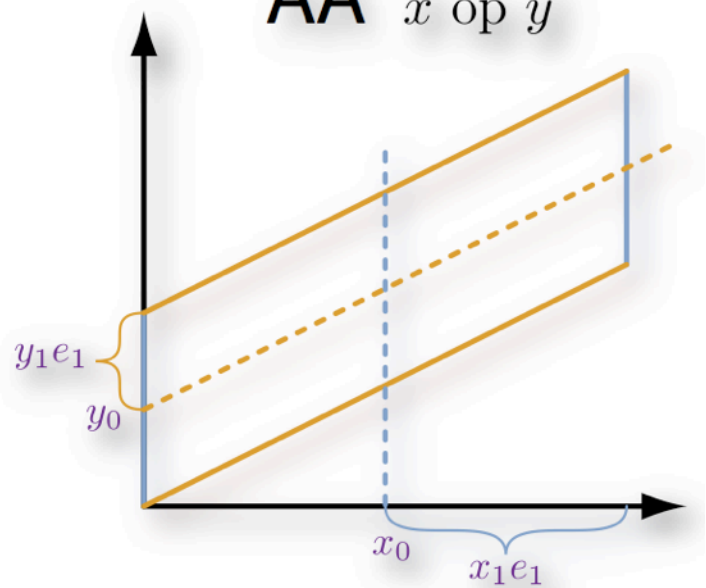
$$\underline{x} = [x_0 - rad(\hat{x}), x_0 + rad(\hat{x})]$$
$$rad(\hat{x}) = \sum_{i=1}^{n} |x_i|$$

$$\hat{x} = x_0 + \sum_{i=1}^{n} x_i e_i$$
$$e_i \in [-1, 1]$$

IA $\underline{x}$ op $\underline{y}$



AA $\hat{x}$ op $\hat{y}$



Interval Arithmetic - Extensions (3)

- [Messine02]: extensions of AA

Affine Form (AF):

$$\hat{x} = x_0 + \sum_{i=1}^{n} x_i e_i$$

AF1:

$$\hat{x} = x_0 + \sum_{i=1}^{n} x_i e_i + x_{n+1} e_{n+1}$$

Interval Arithmetic - Extensions (3)

- [Messine02]: extensions of AA

Affine Form (AF):

$$\hat{x} = x_0 + \sum_{i=1}^{n} x_i e_i$$

$$\mathbf{c} \times \hat{x} = \mathbf{c}x_0 + \mathbf{c}\sum_{i=1}^{n} x_i e_i$$

$$\mathbf{c} \pm \hat{x} = (\mathbf{c} \pm x_0) + \sum_{i=1}^{n} x_i e_i$$

$$\hat{x} \pm \hat{y} = (x_0 \pm y_0) \pm \sum_{i=1}^{n} (x_i \pm y_i) e_i$$

AF1:

$$\hat{x} = x_0 + \sum_{i=1}^{n} x_i e_i + x_{n+1} e_{n+1}$$

Interval Arithmetic - Extensions (3)

- [Messine02]: extensions of AA

Affine Form (AF):

$$\hat{x} = x_0 + \sum_{i=1}^{n} x_i e_i$$

$$\mathbf{c} \times \hat{x} = \mathbf{c}x_0 + \mathbf{c}\sum_{i=1}^{n} x_i e_i$$

$$\mathbf{c} \pm \hat{x} = (\mathbf{c} \pm x_0) + \sum_{i=1}^{n} x_i e_i$$

$$\hat{x} \pm \hat{y} = (x_0 \pm y_0) \pm \sum_{i=1}^{n} (x_i \pm y_i) e_i$$

AF1:

$$\hat{x} = x_0 + \sum_{i=1}^{n} x_i e_i + x_{n+1} e_{n+1}$$

$$\mathbf{c} \times \hat{x} = (\mathbf{c}x_0) + \sum_{i=1}^{n} \mathbf{c}x_i e_i + |cx_{n+1}| e_{n+1}$$

$$\mathbf{c} \pm \hat{x} = (\mathbf{c} \pm x_0) + \sum_{i=1}^{n} x_i e_i + |x_{n+1}| e_{n+1}$$

$$\hat{x} \pm \hat{y} = (x_0 \pm y_0) + \sum_{i=1}^{n} (x_i \pm y_i) e_i + (x_{n+1} + y_{n+1}) e_{n+1}$$

Interval Arithmetic - Extensions (3)

- [Messine02]: extensions of AA

Affine Form (AF):

$$\hat{x} = x_0 + \sum_{i=1}^{n} x_i e_i$$

$$\mathbf{c} \times \hat{x} = \mathbf{c}x_0 + \mathbf{c}\sum_{i=1}^{n} x_i e_i$$

$$\mathbf{c} \pm \hat{x} = (\mathbf{c} \pm x_0) + \sum_{i=1}^{n} x_i e_i$$

$$\hat{x} \pm \hat{y} = (x_0 \pm y_0) \pm \sum_{i=1}^{n} (x_i \pm y_i) e_i$$

$$\hat{x} \times \hat{y} = x_0 y_0 + \sum_{i=1}^{n} (x_i y_0 + y_i x_0) e_i + rad(\hat{x})rad(\hat{y})$$

AF1:

$$\hat{x} = x_0 + \sum_{i=1}^{n} x_i e_i + x_{n+1} e_{n+1}$$

$$\mathbf{c} \times \hat{x} = (\mathbf{c}x_0) + \sum_{i=1}^{n} \mathbf{c}x_i e_i + |cx_{n+1}| e_{n+1}$$

$$\mathbf{c} \pm \hat{x} = (\mathbf{c} \pm x_0) + \sum_{i=1}^{n} x_i e_i + |x_{n+1}| e_{n+1}$$

$$\hat{x} \pm \hat{y} = (x_0 \pm y_0) + \sum_{i=1}^{n} (x_i \pm y_i) e_i + (x_{n+1} + y_{n+1}) e_{n+1}$$

Interval Arithmetic - Extensions (3)

- [Messine02]: extensions of AA

Affine Form (AF):

$$\hat{x} = x_0 + \sum_{i=1}^{n} x_i e_i$$

$$\mathbf{c} \times \hat{x} = \mathbf{c}x_0 + \mathbf{c}\sum_{i=1}^{n} x_i e_i$$

$$\mathbf{c} \pm \hat{x} = (\mathbf{c} \pm x_0) + \sum_{i=1}^{n} x_i e_i$$

$$\hat{x} \pm \hat{y} = (x_0 \pm y_0) \pm \sum_{i=1}^{n} (x_i \pm y_i) e_i$$

$$\hat{x} \times \hat{y} = x_0 y_0 + \sum_{i=1}^{n} (x_i y_0 + y_i x_0) e_i + rad(\hat{x}) rad(\hat{y})$$

AF1:

$$\hat{x} = x_0 + \sum_{i=1}^{n} x_i e_i + x_{n+1} e_{n+1}$$

$$\mathbf{c} \times \hat{x} = (\mathbf{c}x_0) + \sum_{i=1}^{n} \mathbf{c}x_i e_i + |cx_{n+1}| e_{n+1}$$

$$\mathbf{c} \pm \hat{x} = (\mathbf{c} \pm x_0) + \sum_{i=1}^{n} x_i e_i + |x_{n+1}| e_{n+1}$$

$$\hat{x} \pm \hat{y} = (x_0 \pm y_0) + \sum_{i=1}^{n} (x_i \pm y_i) e_i + (x_{n+1} + y_{n+1}) e_{n+1}$$

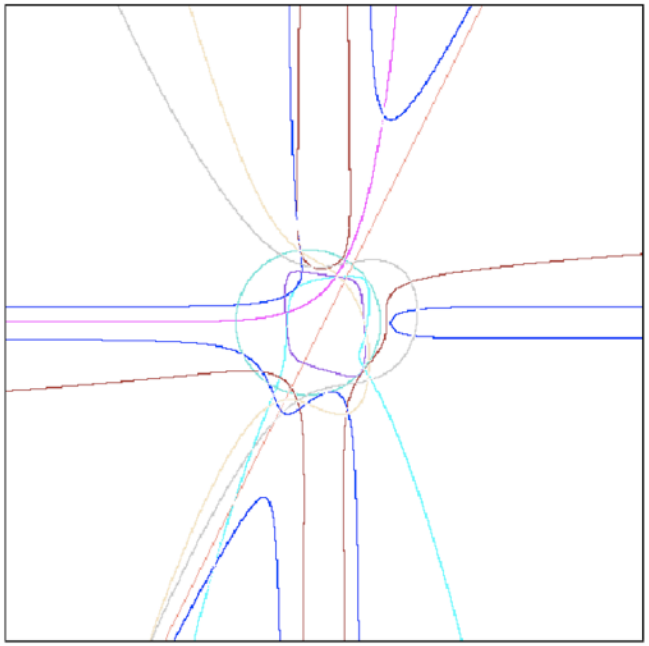
$$\hat{x} \times \hat{y} = (x_0 y_0) + \sum_{i=1}^{n} (x_0 y_i + y_0 x_i) e_i + (|x_0 y_{n+1}| + |y_0 x_{n+1}| + rad(\hat{A}) rad(\hat{B})) e_{n+1}$$

Outline

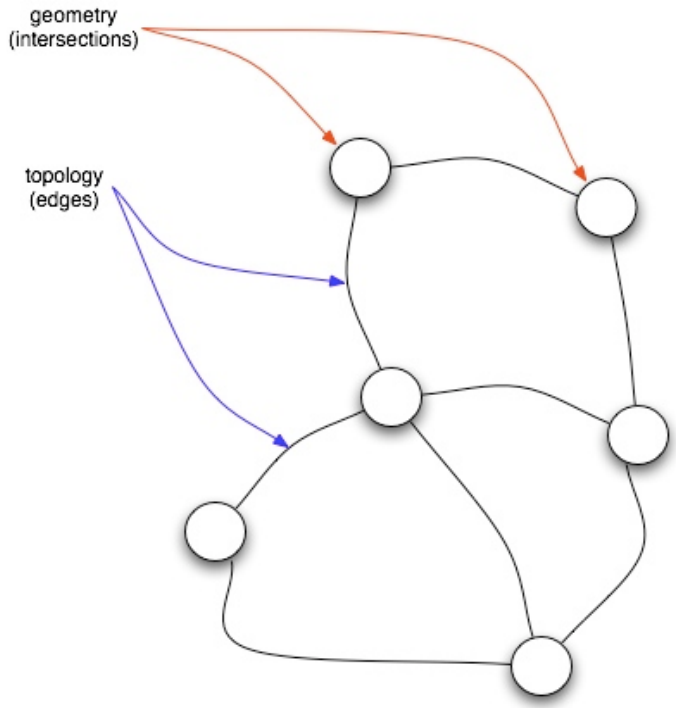
- Motivation
- Interval Arithmetic (IA)
- Arrangement of arbitrary implicit planar curves
- Ray casting arbitrary implicits
- Conclusions & Future Work

Arrangement of curves (1)

The goal



input:
collection of implicit curves

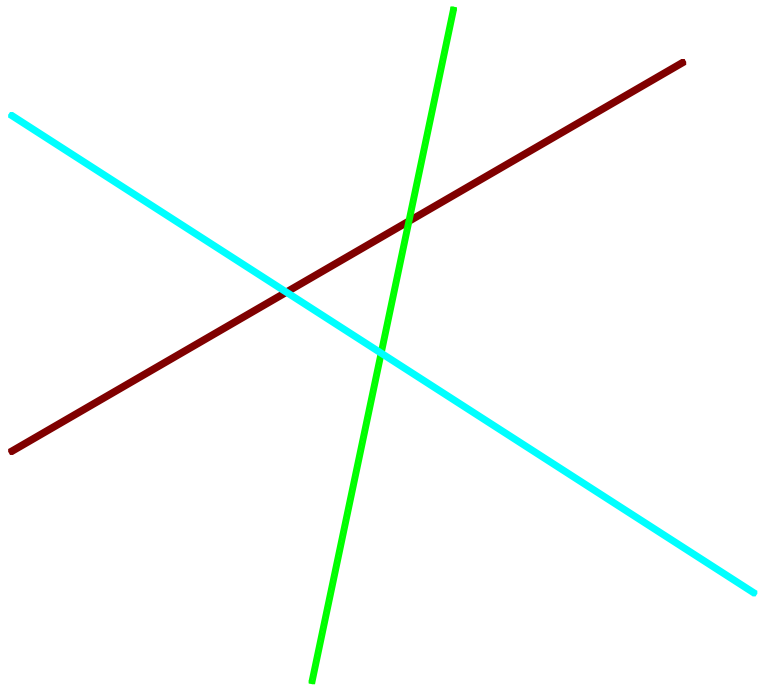


desired output:
arrangement graph

An implicit $2D$ curve is the zero-set of a function $f : \Omega \subseteq \mathbb{R}^2 \rightarrow \mathbb{R}$
Example: $y - ax - b = 0$

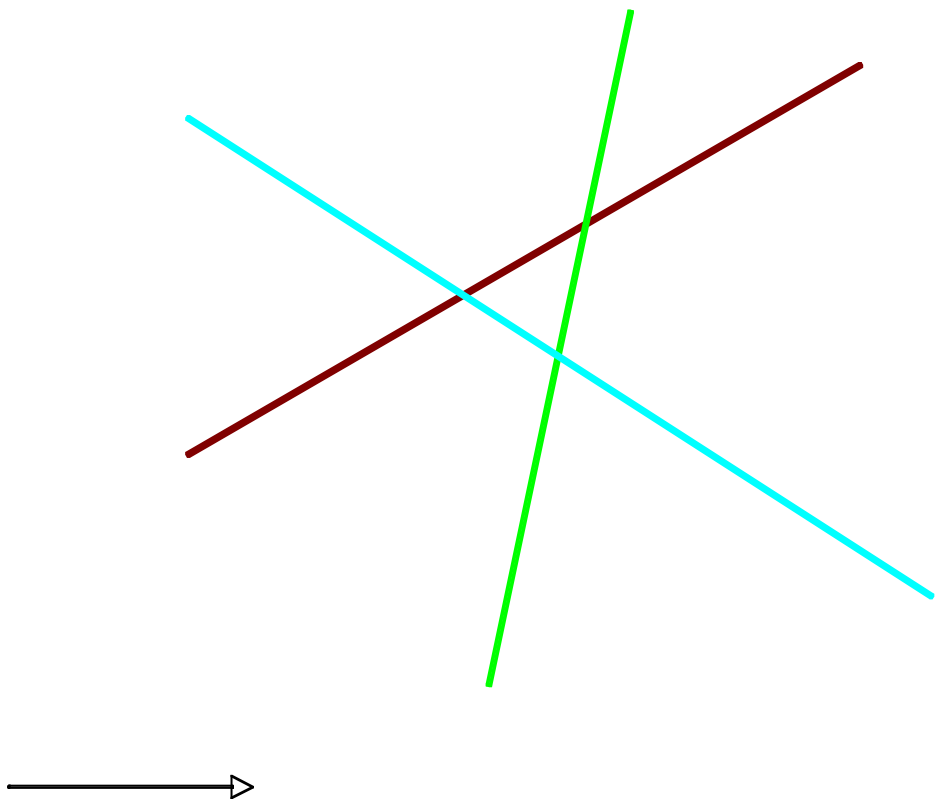
Arrangement of curves (2)

- Classical method: line sweep (e.g. Bentley-Ottmann)



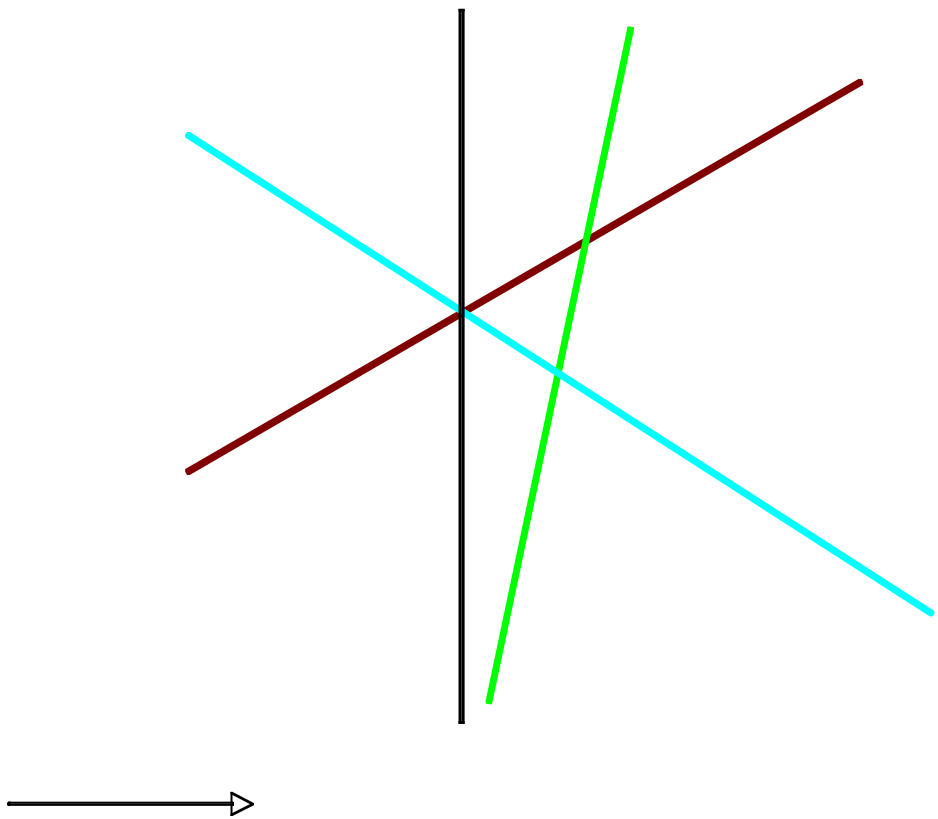
Arrangement of curves (2)

- Classical method: line sweep (e.g. Bentley-Ottmann)



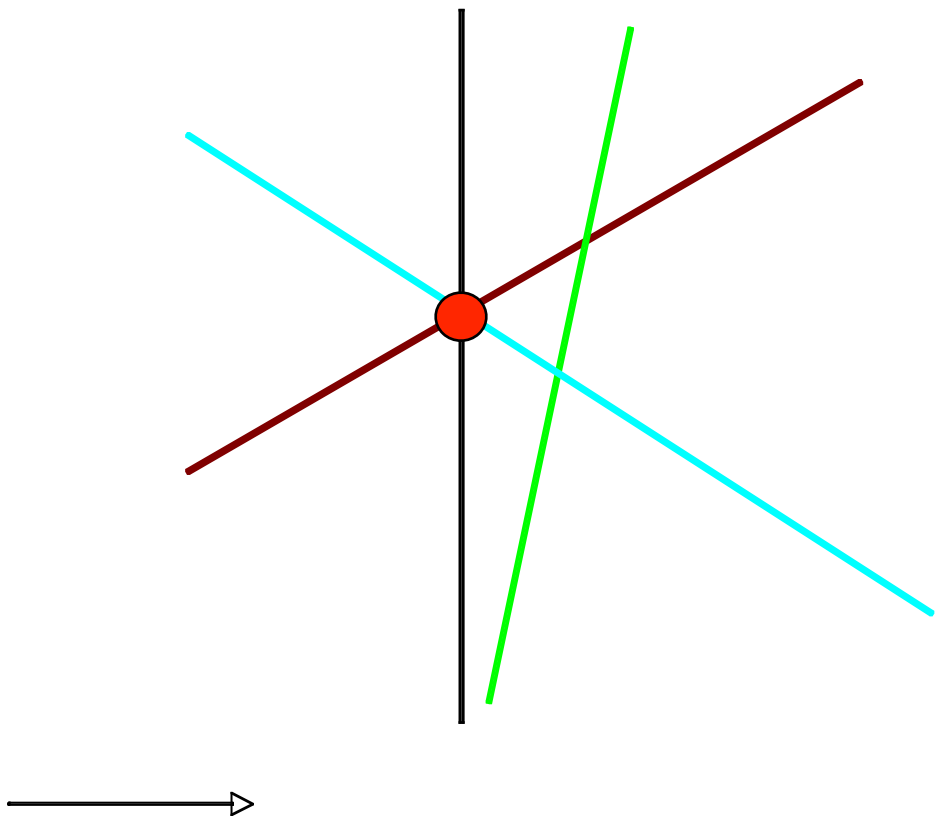
Arrangement of curves (2)

- Classical method: line sweep (e.g. Bentley-Ottmann)



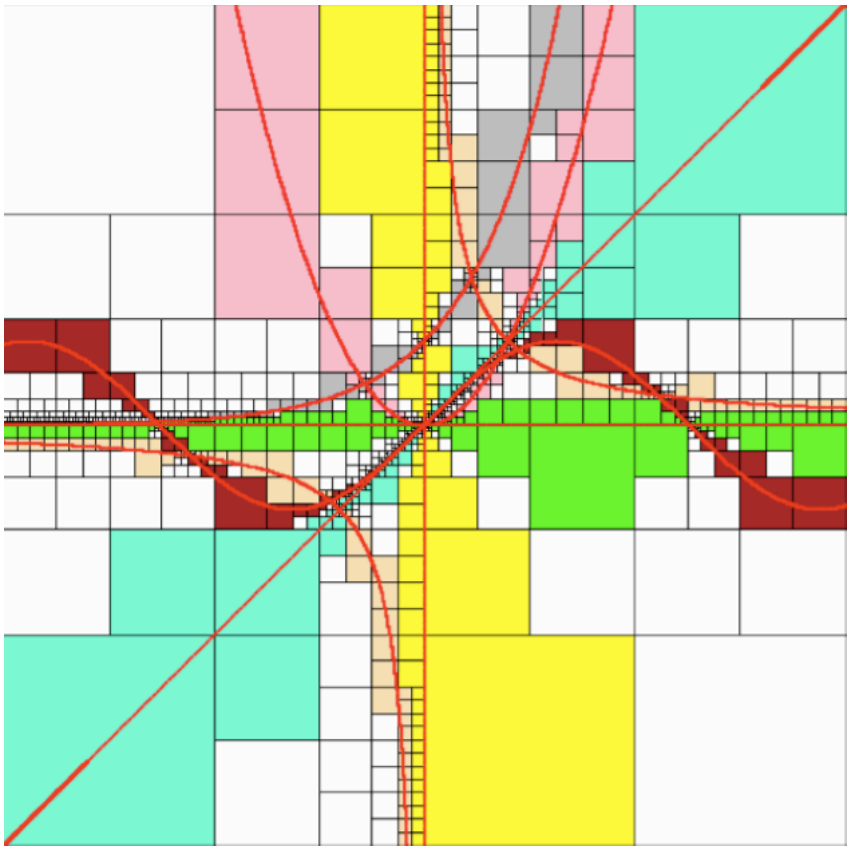
Arrangement of curves (2)

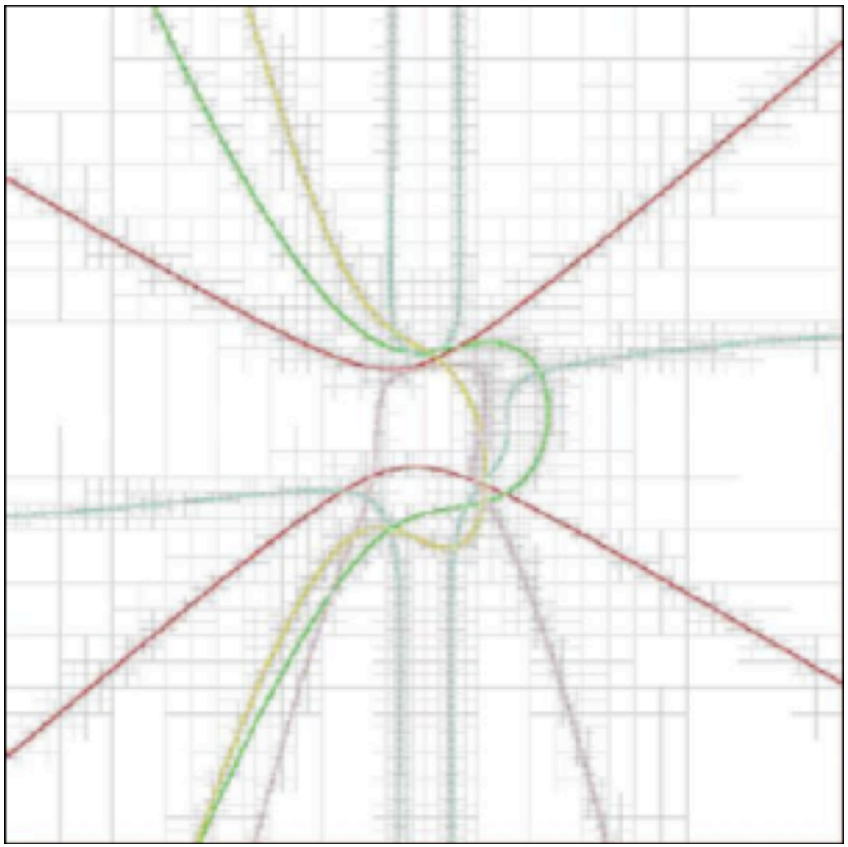
- Classical method: line sweep (e.g. Bentley-Ottmann)



Arrangement of curves (3)

- Our method: subdivision + IA





Emerging subdivision approaches: Yap06, Wintz06

Arrangement of curves (4)

Collaboration with Thomas Breuel (DFKI Kaiserslautern)

CAPS (Curves Arrangements by Planar Subdivisions) algorithm:

- Input: n implicit curves + $2D$ box
- Output: quadtree
  - root node = initial bounding box
  - leaf labeled as containing zero curve, one curve or indeterminate
- Recursively examine several properties

Arrangement of curves (4)

Collaboration with Thomas Breuel (DFKI Kaiserslautern)

CAPS (Curves Arrangements by Planar Subdivisions) algorithm:

- Input: n implicit curves + $2D$ box
- Output: quadtree
  - root node = initial bounding box
  - leaf labeled as containing zero curve, one curve or indeterminate
- Recursively examine several properties

```
Create a node associated to a box
if the box is empty
  mark node as 0
  return node
if the box contains at most one curve
  if it contains exactly one curve(index)
    mark node as 1
    set the index
    return node
  else
    mark node as 0
    return node
if the box is below the threshold
  return node
Call CAPS on each child box
return node
```

Arrangement of curves (4)

Collaboration with Thomas Breuel (DFKI Kaiserslautern)

CAPS (Curves Arrangements by Planar Subdivisions) algorithm:

- Input: n implicit curves + $2D$ box
- Output: quadtree
  - root node = initial bounding box
  - leaf labeled as containing zero curve, one curve or indeterminate
- Recursively examine several properties

```
Create a node associated to a box
if the box is empty
  mark node as 0
  return node
if the box contains at most one curve
  if it contains exactly one curve(index)
    mark node as 1
    set the index
    return node
  else
    mark node as 0
    return node
if the box is below the threshold
  return node
Call CAPS on each child box
return node
```

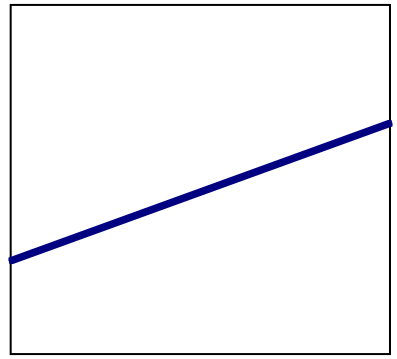
Arrangement of curves (5)

The recursive test:

When a box contains **at most one curve**, i.e. function evaluation returns interval containing zero

Question: is the box empty or does it contain one curve?

Solution: **breadth-first recursive function** which checks whether the current box is strictly positive or strictly negative. If not, the box splits into four and the children are enqueued to be checked.



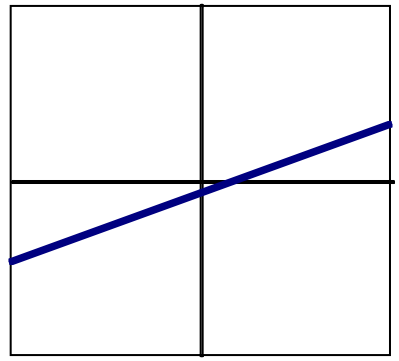
Arrangement of curves (5)

The recursive test:

When a box contains **at most one curve**, i.e. function evaluation returns interval containing zero

Question: is the box empty or does it contain one curve?

Solution: **breadth-first recursive function** which checks whether the current box is strictly positive or strictly negative. If not, the box splits into four and the children are enqueued to be checked.



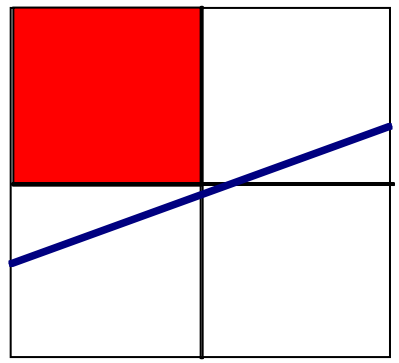
Arrangement of curves (5)

The recursive test:

When a box contains **at most one curve**, i.e. function evaluation returns interval containing zero

Question: is the box empty or does it contain one curve?

Solution: **breadth-first recursive function** which checks whether the current box is strictly positive or strictly negative. If not, the box splits into four and the children are enqueued to be checked.



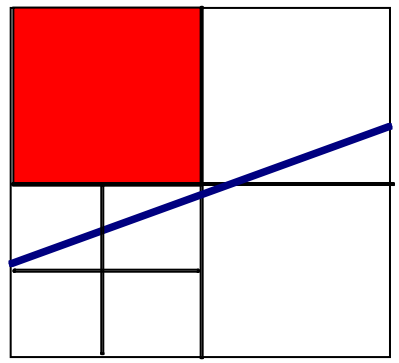
Arrangement of curves (5)

The recursive test:

When a box contains **at most one curve**, i.e. function evaluation returns interval containing zero

Question: is the box empty or does it contain one curve?

Solution: **breadth-first recursive function** which checks whether the current box is strictly positive or strictly negative. If not, the box splits into four and the children are enqueued to be checked.



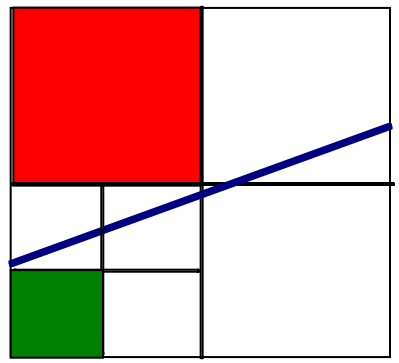
Arrangement of curves (5)

The recursive test:

When a box contains **at most one curve**, i.e. function evaluation returns interval containing zero

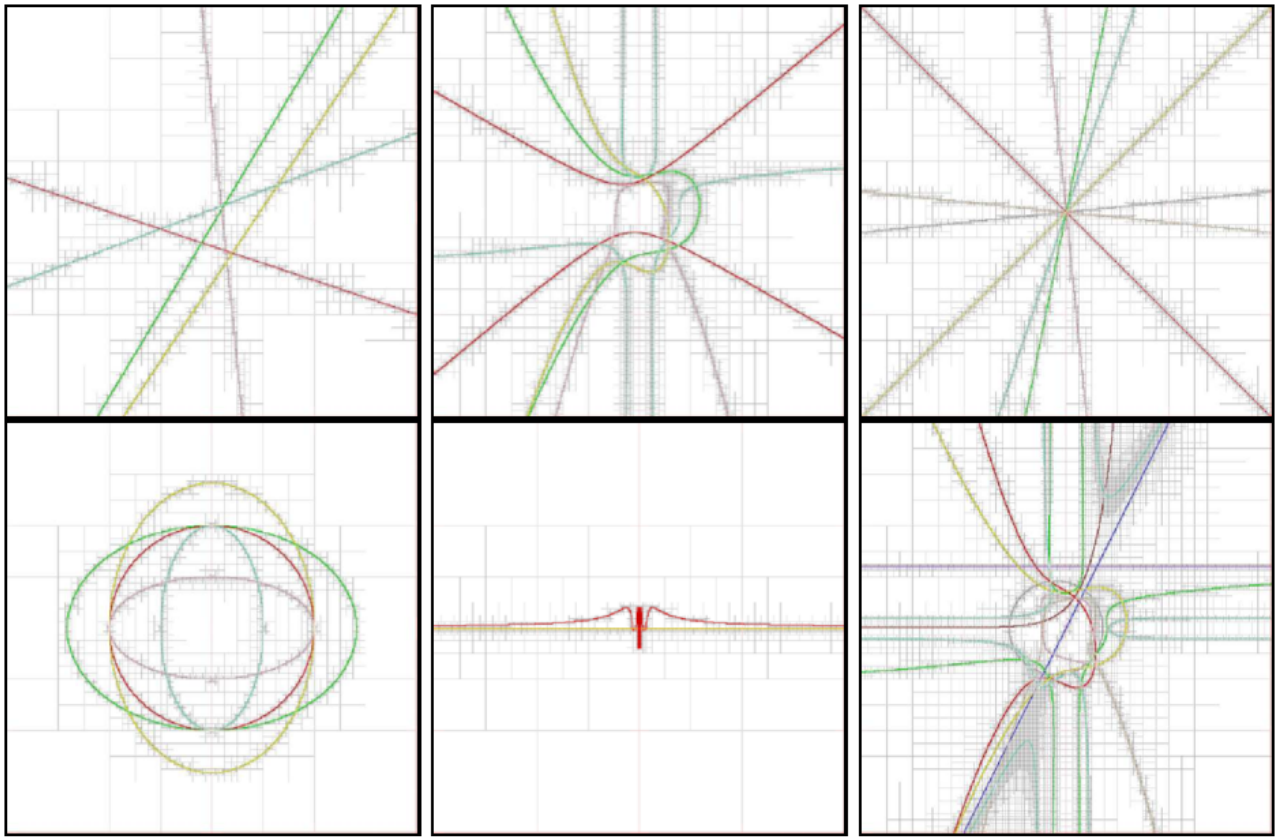
Question: is the box empty or does it contain one curve?

Solution: **breadth-first recursive function** which checks whether the current box is strictly positive or strictly negative. If not, the box splits into four and the children are enqueued to be checked.



Arrangement of curves (6)

- Results



Arrangement of curves (7)

Class of curves	Quadtree size	Runtime (s)
Lines (5)	12097	0.8
Algebraic curves (5)	32875	1.3
Degenerate (7 lines)	19546	1.9
Degenerate (5 polynomials)	251440	2.8
Trigonometric functions (3)	12271	0.2
Inverse Sine (2)	20800	12.6
Arbitrary curves (10)	78736	7.5

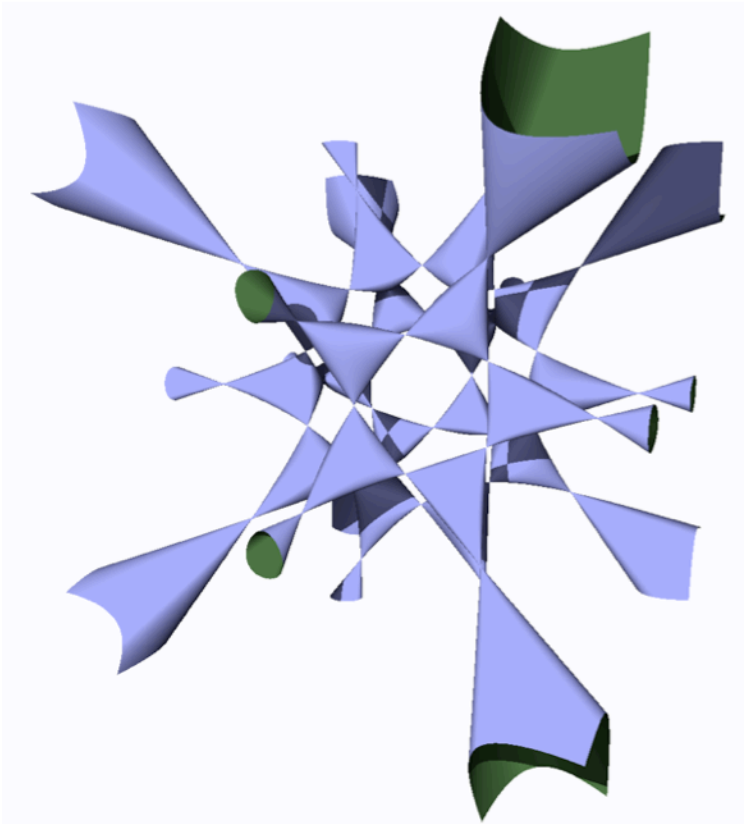
The CAPS algorithm has been used to compute the quadtree structure.
The implementation is in C++ and benchmarking was carried out on a Pentium Xeon 3.6Ghz 64-bit processor with 2GB memory running Linux.

Outline

- Motivation
- Interval Arithmetic (IA)
- Arrangement of arbitrary implicit planar curves
- Ray casting arbitrary implicits
- Conclusions & Future Work

Ray casting implicits (1)

The goal



Barth Sextic:

$$4(\tau^2x^2 - y^2)(\tau^2y^2 - z^2)(\tau^2z^2 - x^2) - (1 + 2\tau)(x^2 + y^2 + z^2 - 1)^2 = 0$$

where $\tau = \frac{1+\sqrt{5}}{2}$ is the Golden Ratio

Given any implicit function, render it correctly and interactively.

Ray casting implicits (2)

Motivation

- Why use implicits?
  - reconstruction of discrete data (e.g. MPU implicits)
  - of visual interest in their own right (e.g. in math)

- Why arbitrary implicits?
  - because we can!
  - one step towards arbitrary primitives
  - many openings: CSG modeling, iso-surfaces, dynamic scenes, etc.

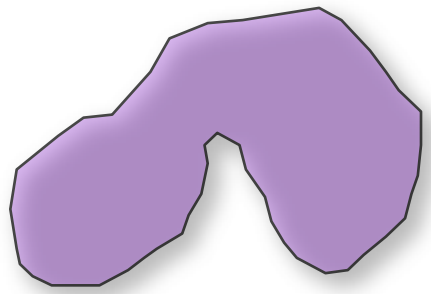
Ray casting implicits (3)

Background: ray casting implicits, a root-finding problem

- We have an implicit function and a ray equation:

$$f(\vec{P}) = f(x, y, z) = 0$$

$$\vec{P}(t) = \vec{O} + t\vec{D}$$



- By substitution we get:

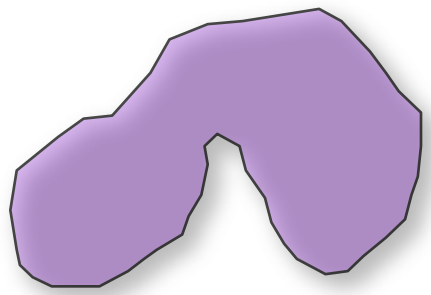
Ray casting implicits (3)

Background: ray casting implicits, a root-finding problem

- We have an implicit function and a ray equation:

$$f(\vec{P}) = f(x, y, z) = 0$$

$$\vec{P}(t) = \vec{O} + t\vec{D}$$



- By substitution we get:

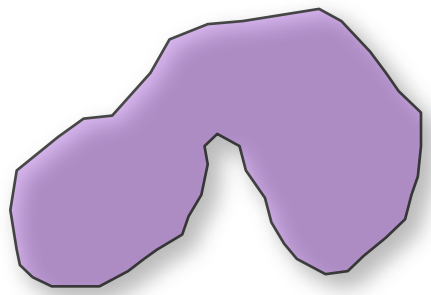
Ray casting implicits (3)

Background: ray casting implicits, a root-finding problem

- We have an implicit function and a ray equation:

$$f(\vec{P}) = f(x, y, z) = 0$$

$$\vec{P}(t) = \vec{O} + t\vec{D}$$



- By substitution we get:

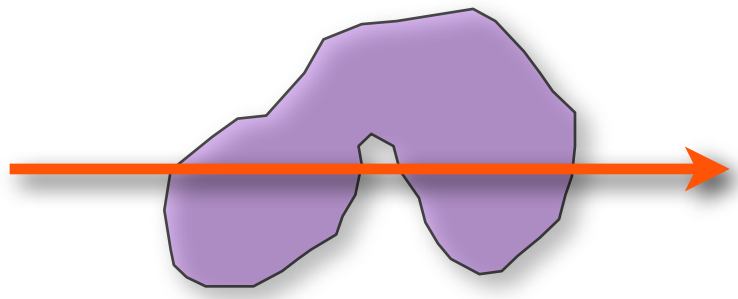
Ray casting implicits (3)

Background: ray casting implicits, a root-finding problem

- We have an implicit function and a ray equation:

$$f(\vec{P}) = f(x, y, z) = 0$$

$$\vec{P}(t) = \vec{O} + t\vec{D}$$



- By substitution we get:

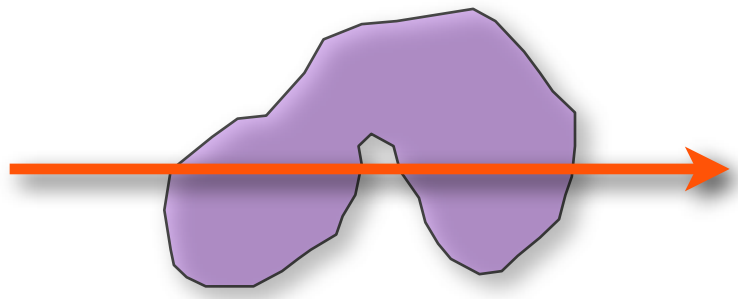
Ray casting implicits (3)

Background: ray casting implicits, a root-finding problem

- We have an implicit function and a ray equation:

$$f(\vec{P}) = f(x, y, z) = 0$$

$$\vec{P}(t) = \vec{O} + t\vec{D}$$



- By substitution we get:

$$f_t(t) = f(O_x + tD_x, O_y + tD_y, O_z + tD_z) = 0$$

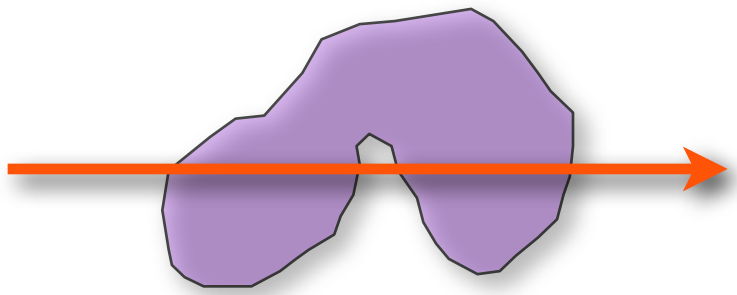
Ray casting implicits (3)

Background: ray casting implicits, a root-finding problem

- We have an implicit function and a ray equation:

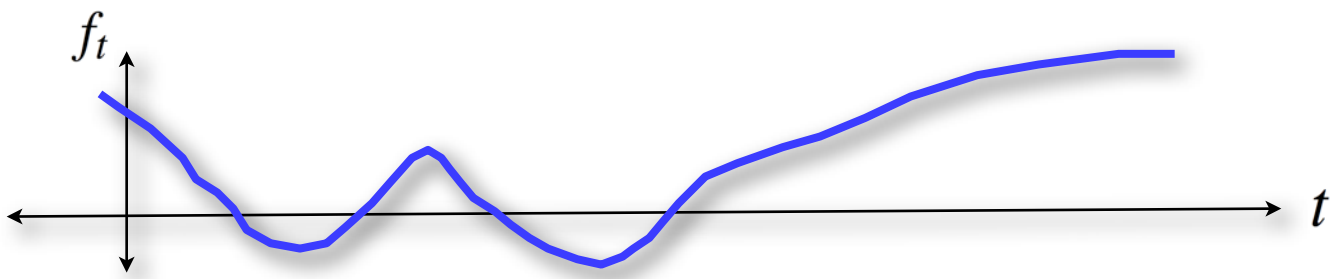
$$f(\vec{P}) = f(x, y, z) = 0$$

$$\vec{P}(t) = \vec{O} + t\vec{D}$$



- By substitution we get:

$$f_t(t) = f(O_x + tD_x, O_y + tD_y, O_z + tD_z) = 0$$



Ray casting implicits (4)

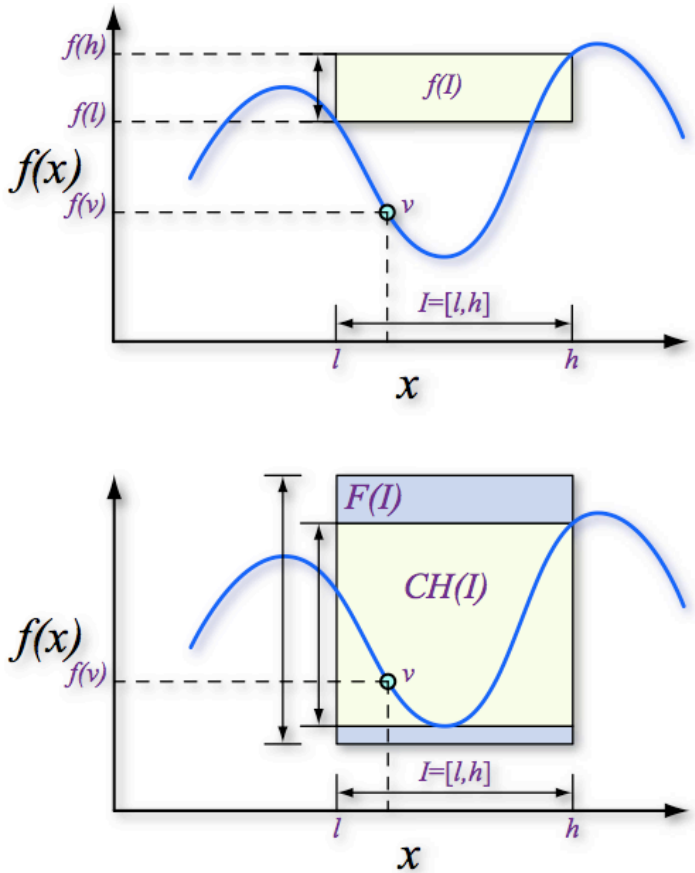
Ray casting implicits using IA

- Given a box $B = X \times Y \times Z \subseteq \Omega$ and a function $f : \Omega \subseteq \mathbb{R}^3 \rightarrow \mathbb{R}$ the corresponding interval extension $F : B \rightarrow F(B)$ is an inclusion function of f, i.e:

$$F(B) \supseteq f(B) = \{f(x, y, z) \mid (x, y, z) \in B\}$$

- Reliable rejection test:

$$0 \notin F(B) \Rightarrow 0 \notin f(B)$$



Ray casting implicits (5)

Collaboration with A. Knoll, C. Hansen, I. Wald (SCI Utah) and H. Hagen

Ray casting implicits interactively on the CPU

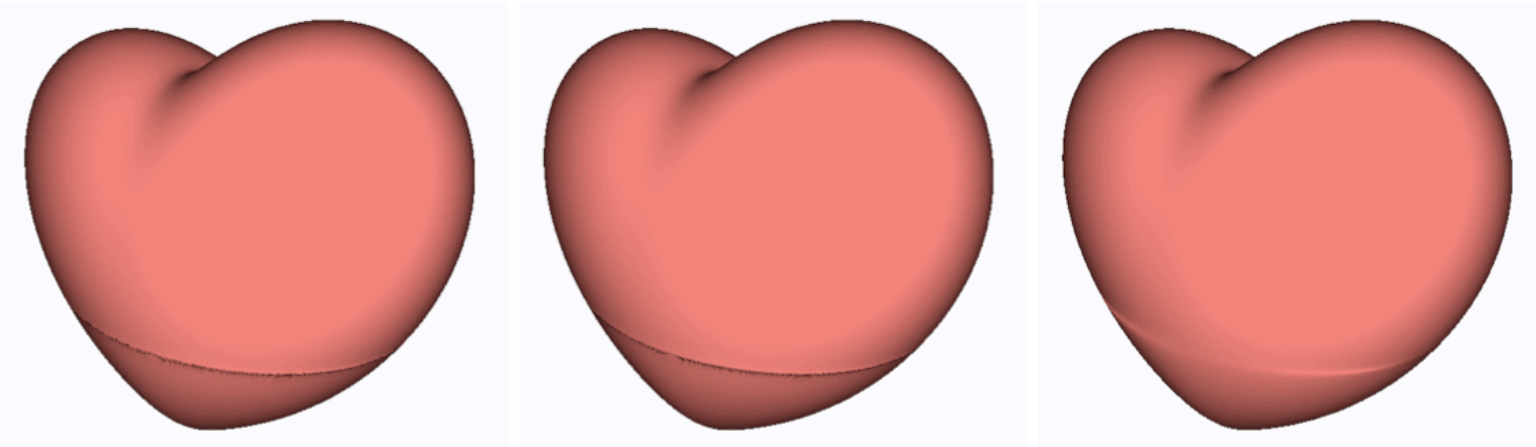
General idea:

- use subdivision + IA (elimination method)
- efficient IA computations through SIMD vectors
- efficient coherent bisection/traversal algorithm
- visually accurate precision, not numerically precise results
  - coarse interval bisection is sufficient
  - no refinement methods

Ray casting implicits (6)

Ray tracing effects

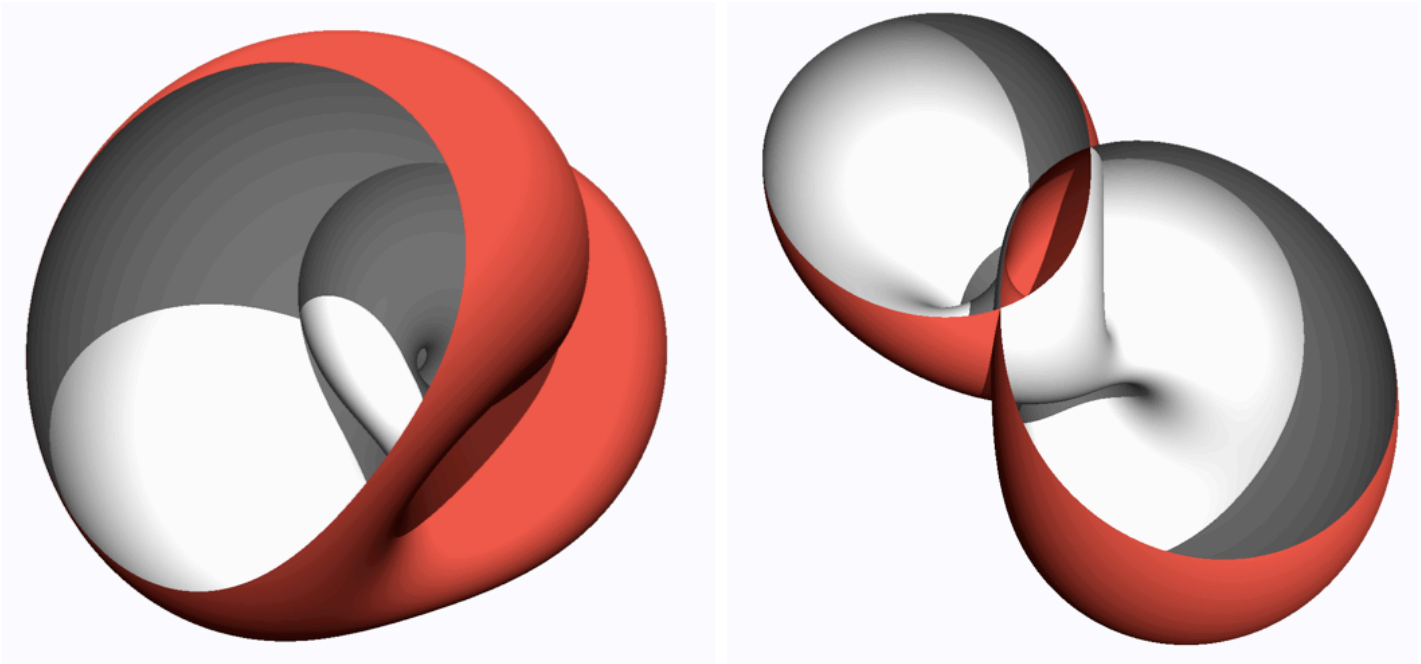
- Lambert/Phong shading
  - gradient given either analytically by user, automatic differentiation or by central differences



Left: analytical gradient; center and right: central differences

Ray casting implicits (7)

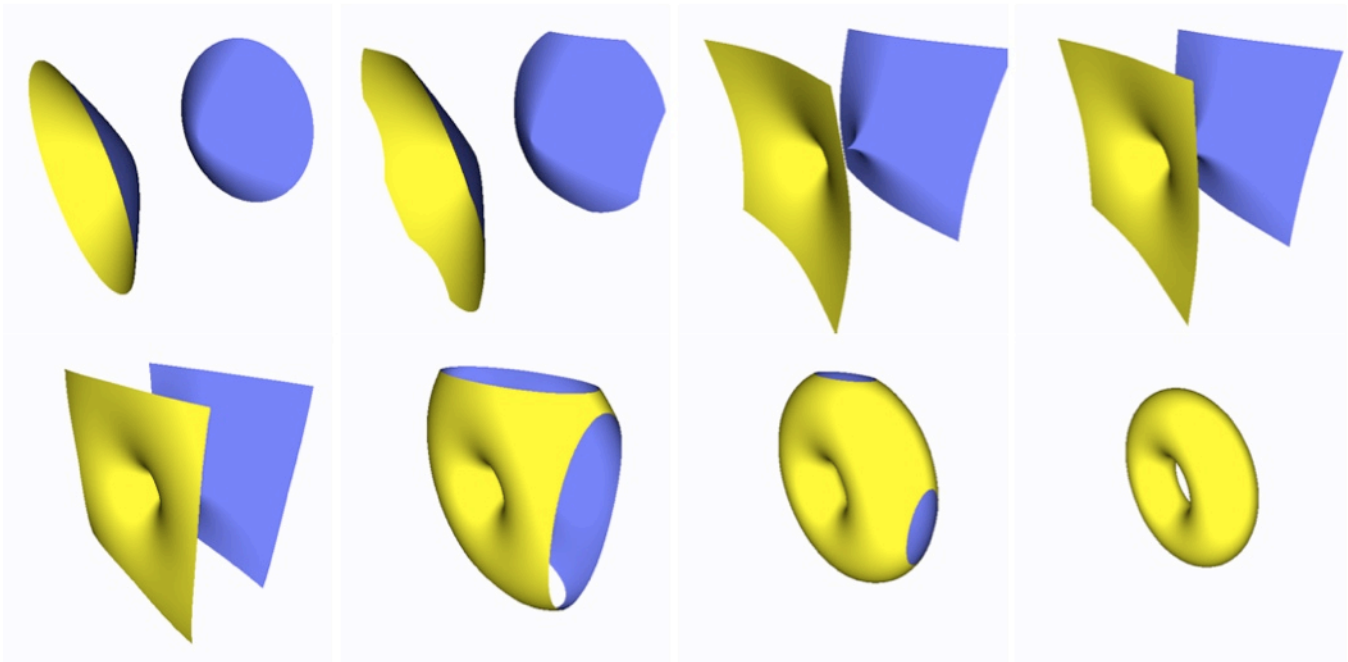
- Shadows
  - help visualization
  - cost 30-50 % in frame rate



Dynamic shadows on a cut of Klein Bottle's $3D$ immersion (3-4 fps)

Ray casting implicits (8)

- Dynamic scenes with $4D$ implicits
  - performed “on the fly” at interactive rates
  - no acceleration structure needed

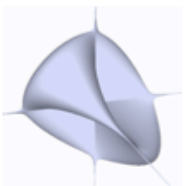


Morphing between a hyperboloid and a torus (9-20 fps)

Ray casting implicits (9)

- Results

Torus	Sphere	Parabolic Cylinder	Hyperboloid
$(r_i - \sqrt{x^2+y^2})^2 + z^2 - r_o^2$	$x^2+y^2+z^2-r^2$	$x^2 - y - r^2$	$-\frac{x^2}{a^2} - \frac{y^2}{a^2} + \frac{z^2}{c^2} - 1$
22.9 fps	20.5 fps	34.1 fps	19.2 fps



Steiner 1	Steiner 2	Mitchell
$x^2y^2+y^2z^2+x^2z^2+xyz$	$(x^2y^2+y^2z^2+x^2z^2)^2+xyz$	$4(x^4+(y^2+z^2)^2)+17x^2(y^2+z^2)$ $-20(x^2+y^2+z^2)+17$
6.1 fps	17.2 fps	5.9 fps

Tangle	Blobby	Absolute value	Inverse function
$x^4-5x^2+y^4-5y^2+z^4-5z^2+11.8$	$\Sigma_{i=1}^{N} \frac{r_i^2}{\|x-p_i\|^2} - 1$	$\|x\|+\|y\|-z$	$\frac{1}{x-y^2} - z$
3.8 fps	4.7 fps	30.8 fps	20.8 fps

Ray casting implicits (9)

- Results

Klein Bottle	Intersecting planes	Teardrop
$(x^2+y^2+z^2+2y-1)((x^2+y^2+z^2-2y-1)^2-8z^2)$ $+16xz(x^2+y^2+z^2-2y-1)$	xy	$0.5x^5+0.5x^4-y^2-z^2$
6.0 fps	33.0 fps	15.3 fps

Linked tori	Barth-sextic
$g(10x,10y-2,10z,13)g(10z,10y+2,10x,13)+1000$ $g(x,y,z,c)=(x^2+y^2+z^2+c)^2-53(x^2+y^2)$	$4(\tau^2x^2-y^2)(\tau^2y^2-z^2)(\tau^2z^2-x^2)$ $-(1+2\tau)(x^2+y^2+z^2-1)^2$
4.0 fps	4.9 fps

Ray casting implicits (10)

In short, our method is

- robust: thanks to bisection and IA
- general: handles any implicit function
- efficient: generally interactive

Ray casting implicits (11)

Collaboration with A. Knoll, A. Kensler, M. Schott, C. Hansen (SCI Utah) and H. Hagen

Ray casting implicits real-time on the GPU

General idea:

- port previous implementation on recent GPU (nvidia g80)
  - same general approach as previously
  - need of a new traversal algorithm, efficient on the GPU
- use more elaborated arithmetic
  - reduced affine arithmetic
  - implementation for shader languages
- shader meta-programming
  - full flexibility, e.g. dynamic surfaces performed “on the fly”
  - multi-bounce effects (transparency, shadows...)

Ray casting implicits (12)

Results

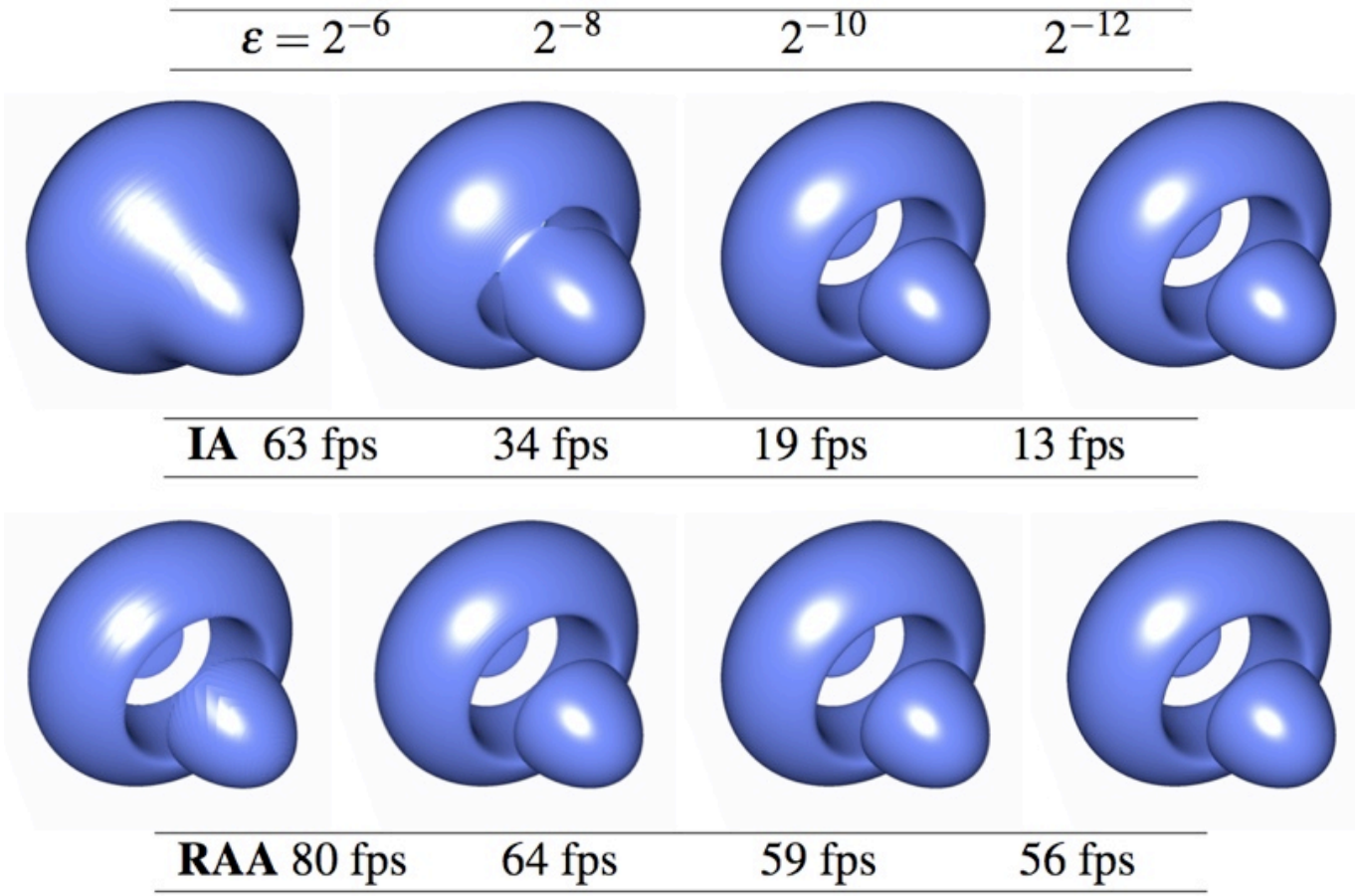
- performed on a 4-core Xeon 2.33 GHz CPU workstation with NVIDIA 8800 GTX at 1024x1024 frame buffer resolution
- precision used: $\varepsilon = 2^{-11}$

FPS

		CPU	GPU		
		$\varepsilon = 2^{-11}$			correct
function (fig)	**degree**	IA		RAA	IA/RAA
sphere	2	15	75	147	165
steiner (7.3)	4	7.5	34	40	38
mitchell (7.2)	8	5.2	16	58	60
teardrop (7.4a)	4	5.5	102	115	121
4-bretzel (7.5a)	12	13	78	48	90
klein b. (7.4b)	6	11	30	110	101
tangle (7.5b)	4	3.2	15	68	71
decocube (7.7)	4	5.5	28	27	28
barth sex. (7.6l)	6	7.4	31	76	88
barth dec. (7.6r)	10	0.9	4.9	15.6	15.6
superquadric	200	18	119	8.3	108
icos.csg (7.8l)	na	-	13.3	-	13.3
sq.csg (7.8r)	na	-	8.9	-	7.2
sin.blob (7.1)	na	-	6.0	-	6.0
cloth (7.9l)	na	-	38	-	44
water (7.9r)	na	-	37	-	44

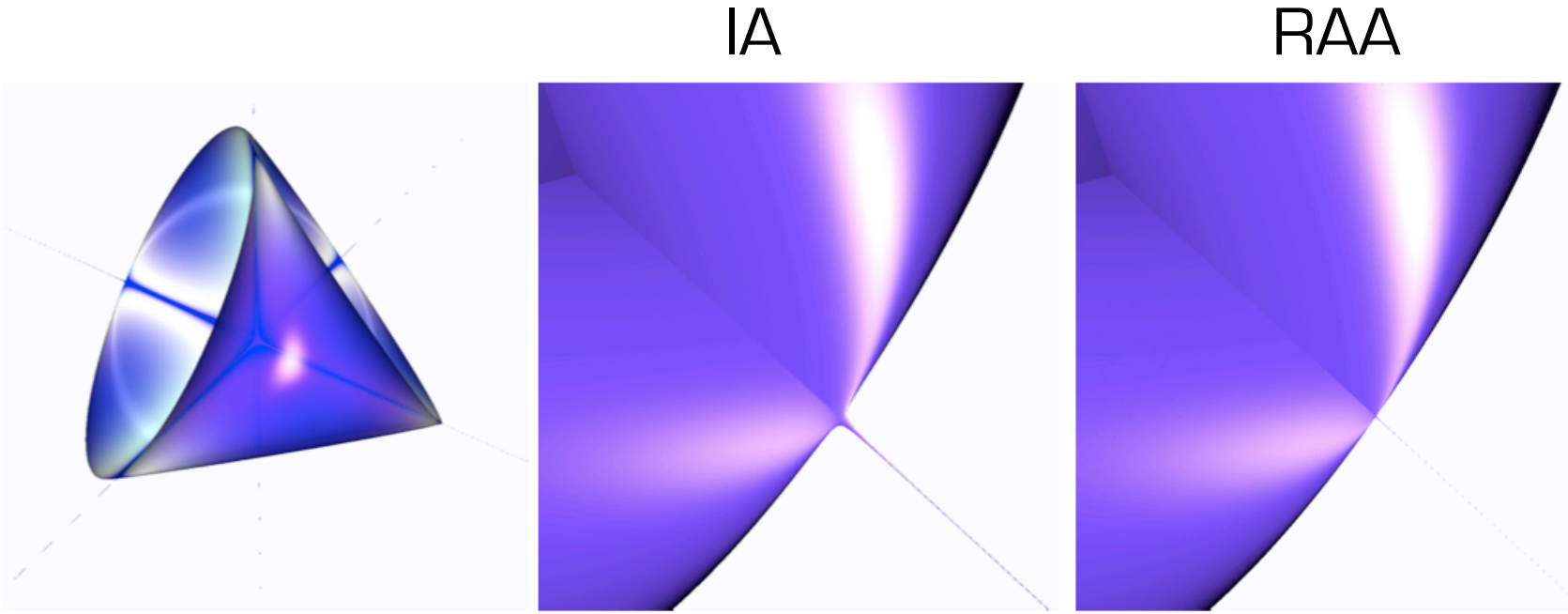
Ray casting implicits (13)

IA vs RAA



Ray casting implicits (14)

Correctness and robustness

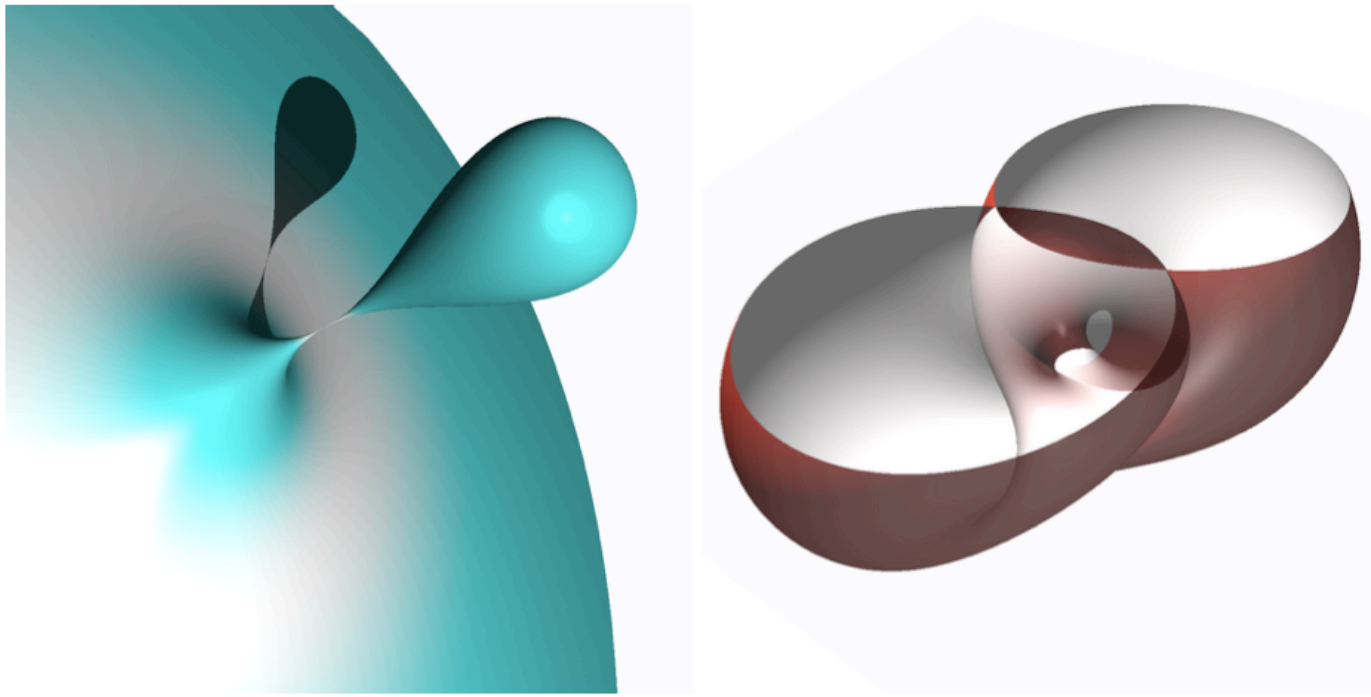


Fine feature visualization in the Steiner surface

Left to right: shading with depth peeling and gradient magnitude coloration; close-up on a singularity with IA at $\varepsilon = 2^{-18}$; and with RAA at the same depth.

Ray casting implicits (15)

Shading effects

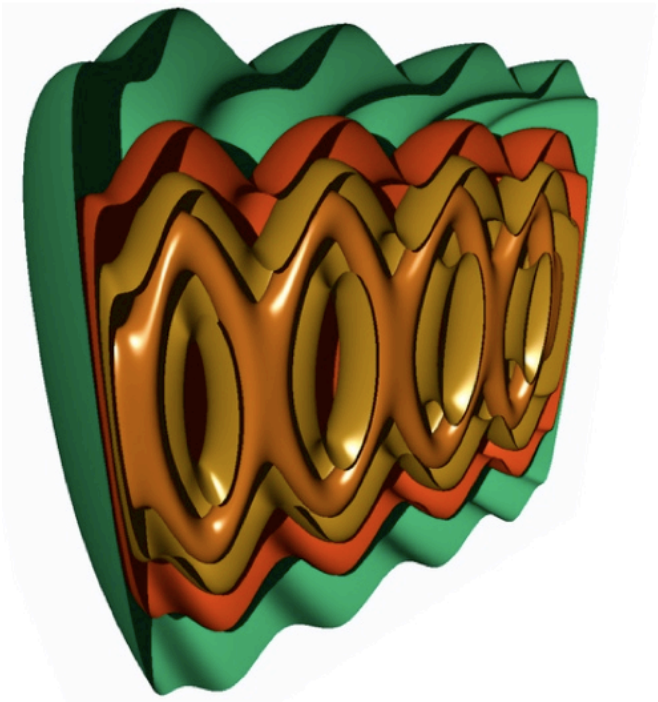


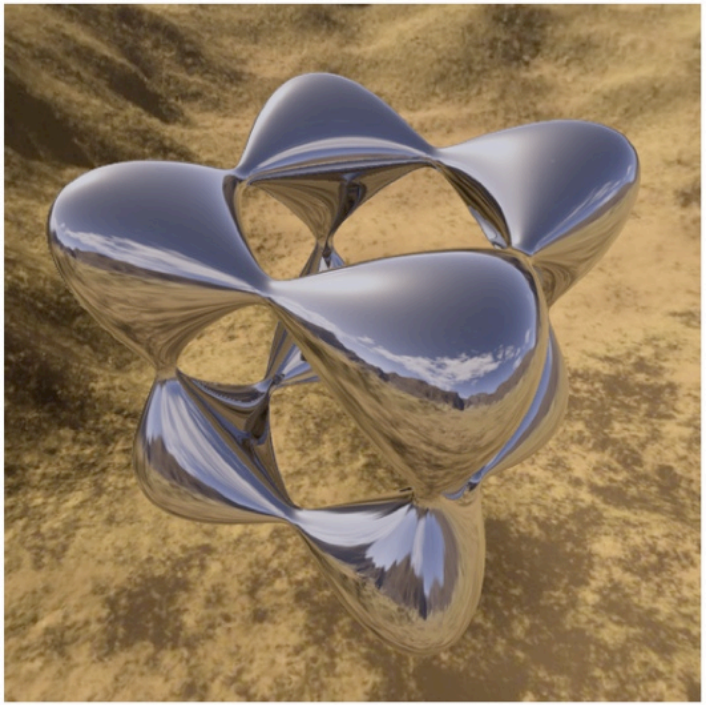
Shading effects: shadows and transparency

Left: shadows on the teardrop (40 fps)
Right: transparency on the Klein Bottle (41 fps)

Ray casting implicits (16)

Shading effects (continued)



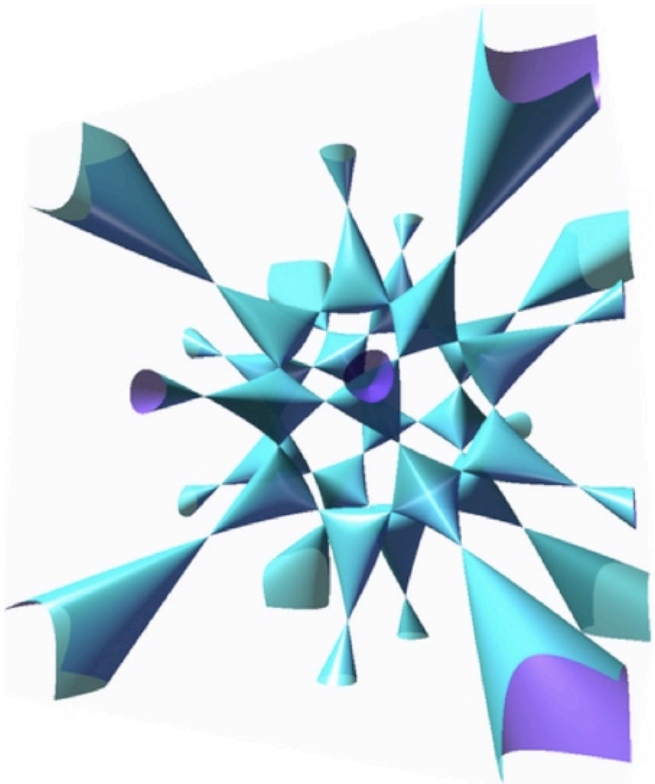


Shading effects: multiple iso-values and reflections

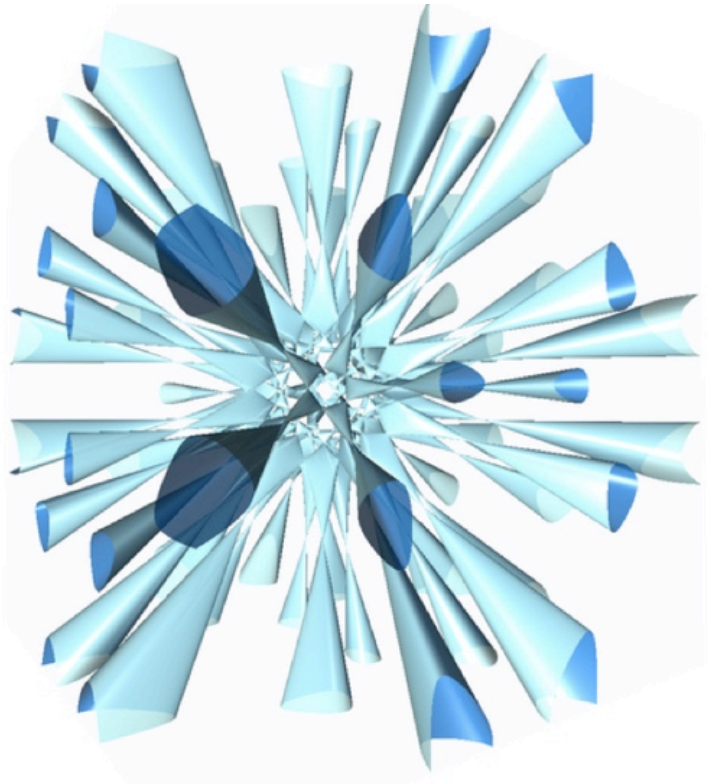
Left: shadows and multiple isovalues of the 4-Bretzel (18 fps)
Right: the tangle with up to six reflection rays (44 fps)

Ray casting implicits (17)

Applications I: mathematical visualization



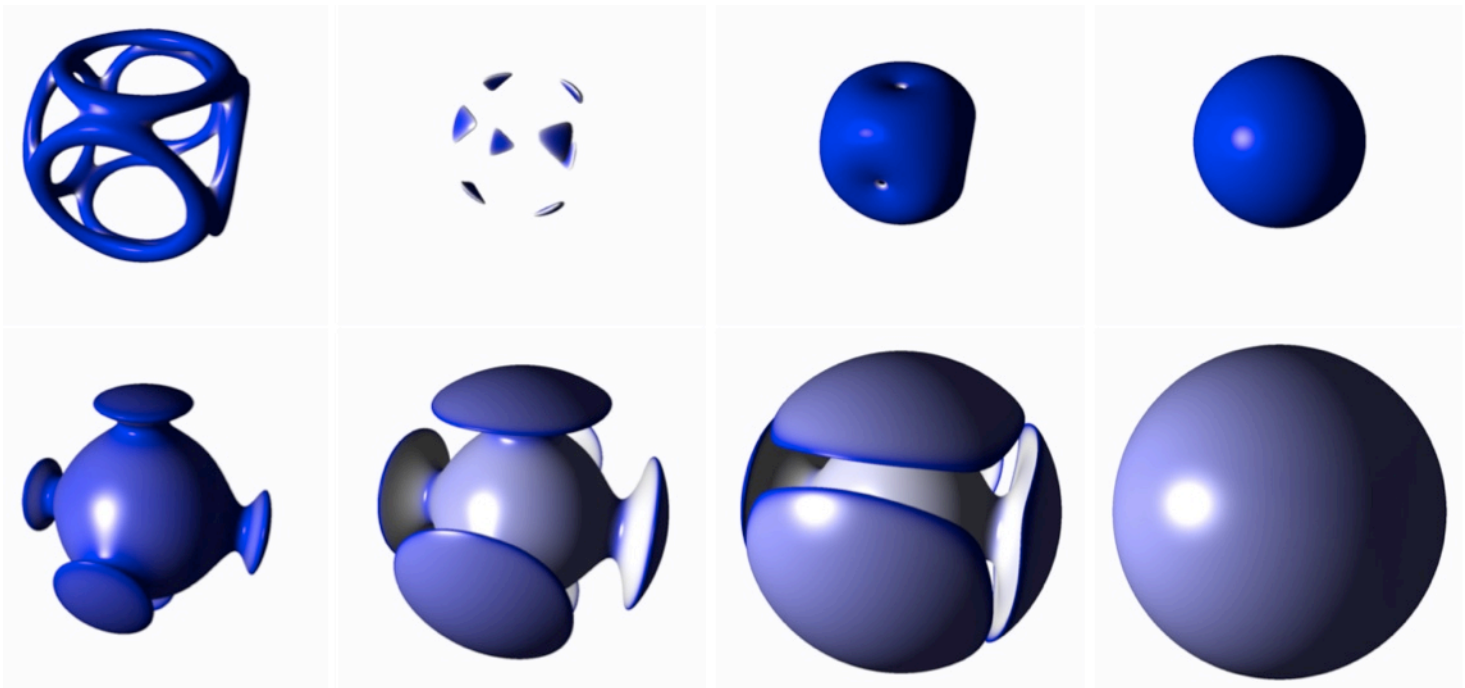
The Barth Sextic surface (88 fps)



The Barth Decic surface (16 fps)

Ray casting implicits (18)

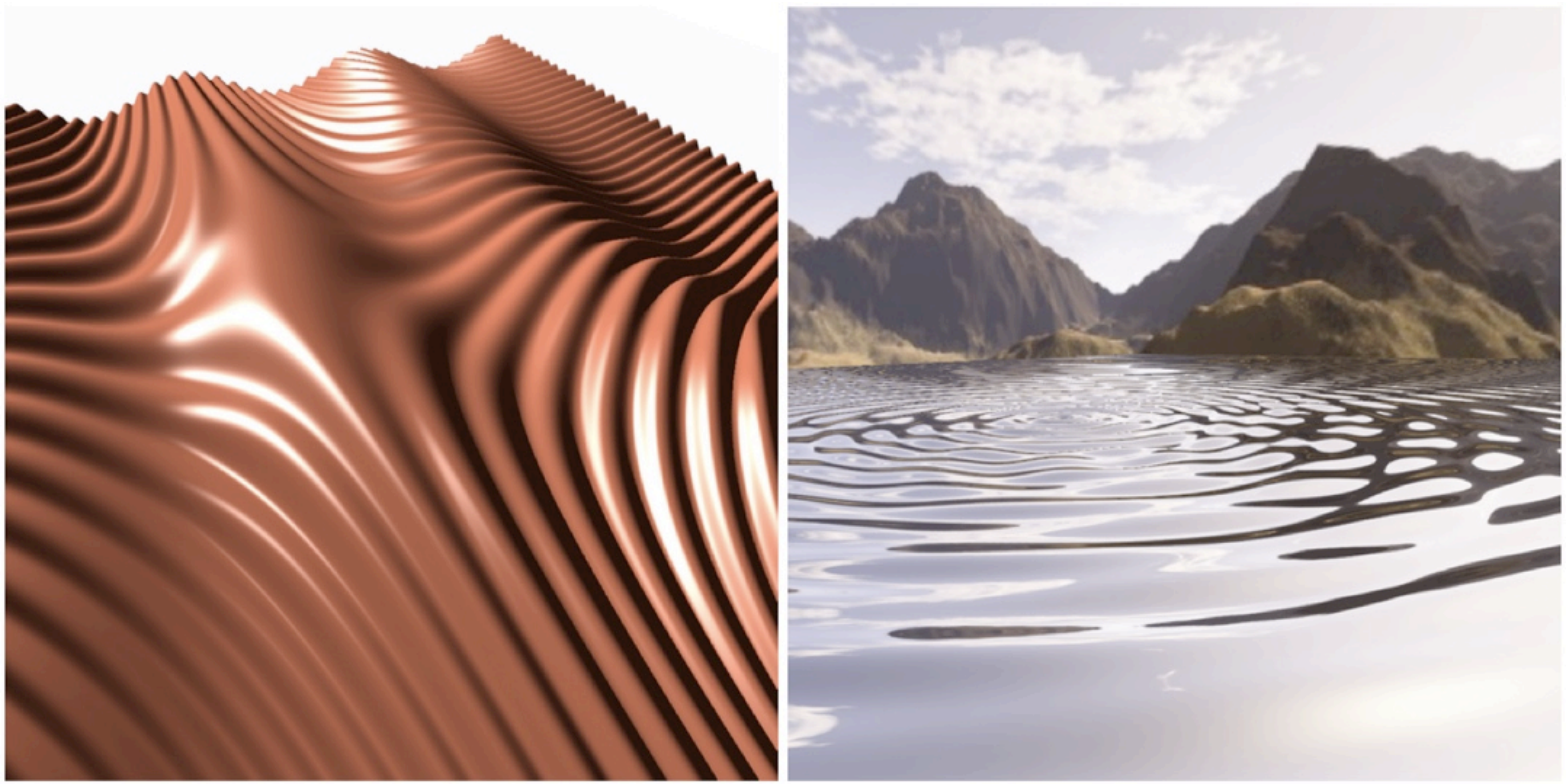
Applications II: interpolation, morphing and blending



$4D$ sigmoid blending of the decocube and a sphere
with interpolation and extrapolation phases, running at 33–50 fps

Ray casting implicits (20)

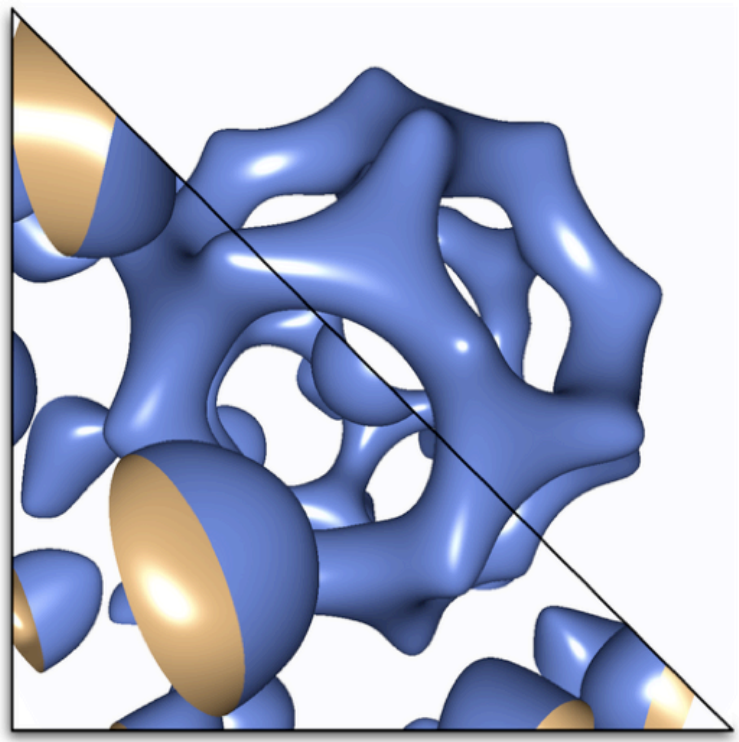
Applications III: procedural geometry

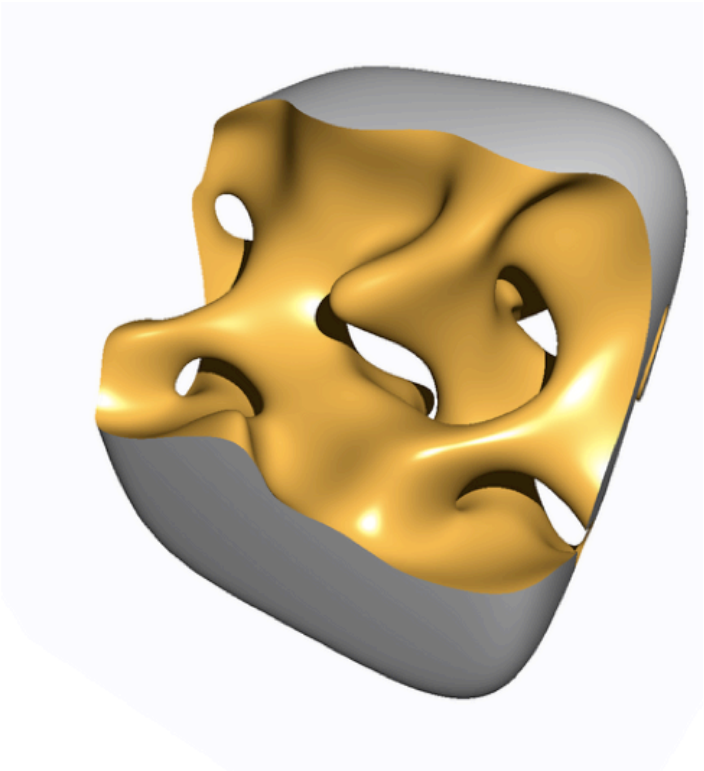


Sinusoid procedural geometry for dynamic simulation of cloth and water.
With IA, these surfaces render at 38 and 37 fps respectively

Ray casting implicits (19)

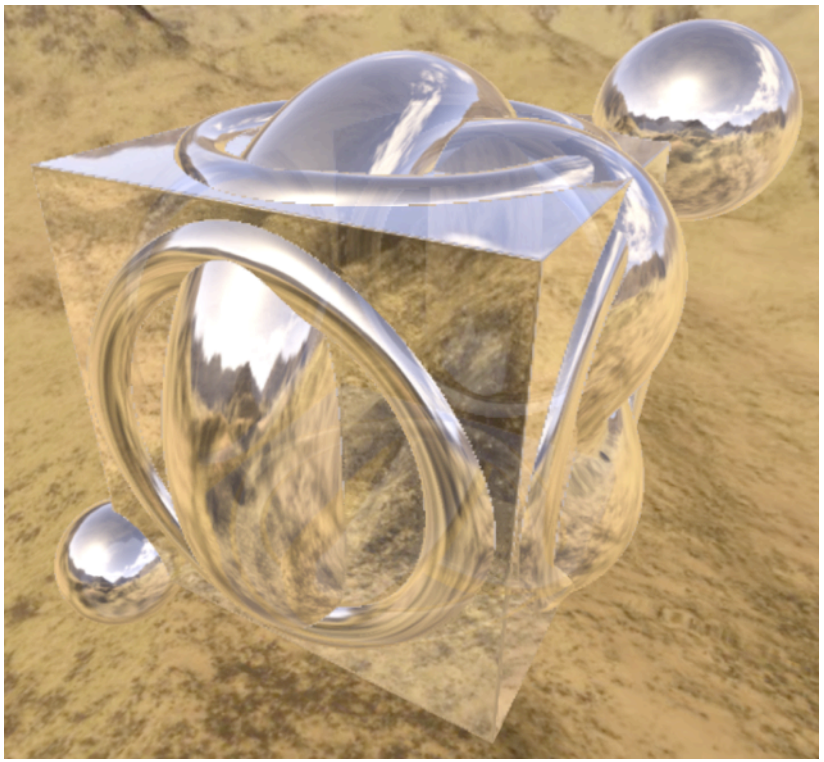
Applications IV: Constructive Solid Geometry





CSG surfaces using level-set conditions (20 fps)

Ray casting implicits (20)



Depth-peeling and reflections on a complicated CSG object (3 fps)

Shadows on 4-bretzel U torus (15 fps)

Ray casting implicits (21)

Conclusion

We have demonstrated a fast, general and robust algorithm for rendering even most difficult implicit functions with good visual quality in real-time on the GPU.

Outline

- Motivation
- Interval Arithmetic (IA)
- Arrangement of arbitrary implicit planar curves
- Ray casting arbitrary implicits
- Conclusions & Future Work

Conclusions

IA-based techniques together with subdivision can be very useful in computer graphics and visualization

- Generality
- Inherent adaptivity, robustness
- Interactive or even real-time

Future Work

Differential geometry and topology for scientific visualization

- mathematical tools:
  - differential/spinorial geometry
  - topology
- computer science tools:
  - extend algorithms to other input (parametric, subdivision)
  - apply to real data sets (volumes, iso-surfaces)

Deutsche Forschungsgemeinschaft DFG

Publications

Younis Hijazi, Aaron Knoll, Mathias Schott, Andrew Kensler, Charles Hansen and Hans Hagen
CSG Operations of Arbitrary Primitives with Interval Arithmetic and Real-Time Ray Tracing
Special Issue of Dagstuhl Proceedings, 2010 (to appear)

Aaron Knoll, Younis Hijazi, Andrew Kensler, Mathias Schott, Charles Hansen and Hans Hagen
Fast Ray Tracing of Arbitrary Implicit Surfaces with Interval and Affine Arithmetic
Computer Graphics Forum, vol. 28(1), pp. 26-40, 2009

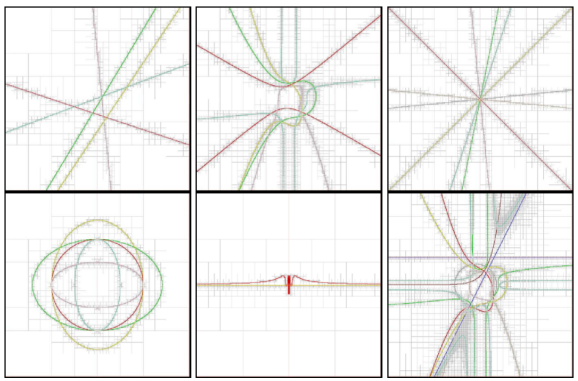
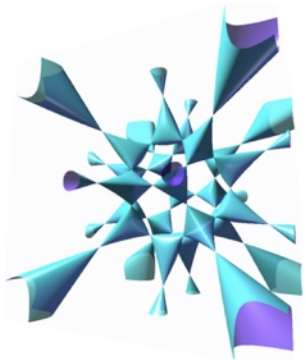
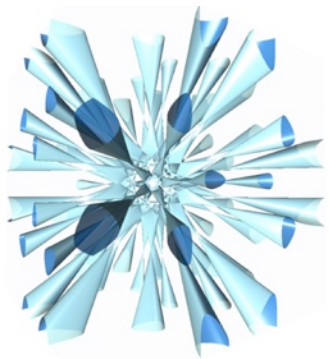
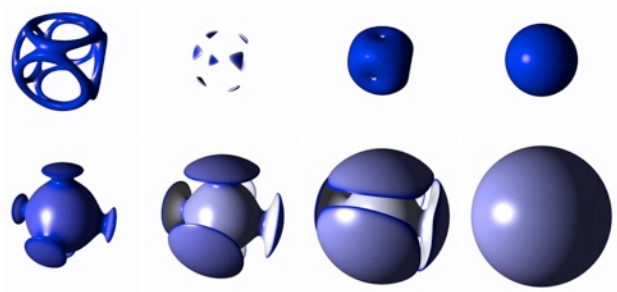
Younis Hijazi
Feature Based Visualization
Ph.D. thesis, TU Kaiserslautern, 2008

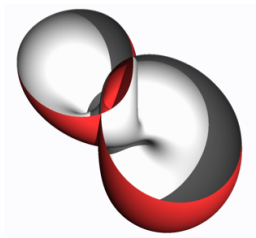
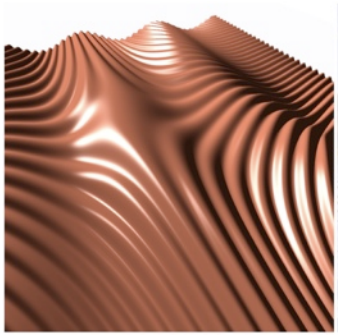
Younis Hijazi, Hans Hagen, Charles Hansen and Kenneth I. Joy
Why Interval Arithmetic is so useful
In H. Hagen, M. Hering-Bertram, Ch. Garth (Eds.) : Visualization of Large and Unstructured Data Sets, pp. 36-53, 2007

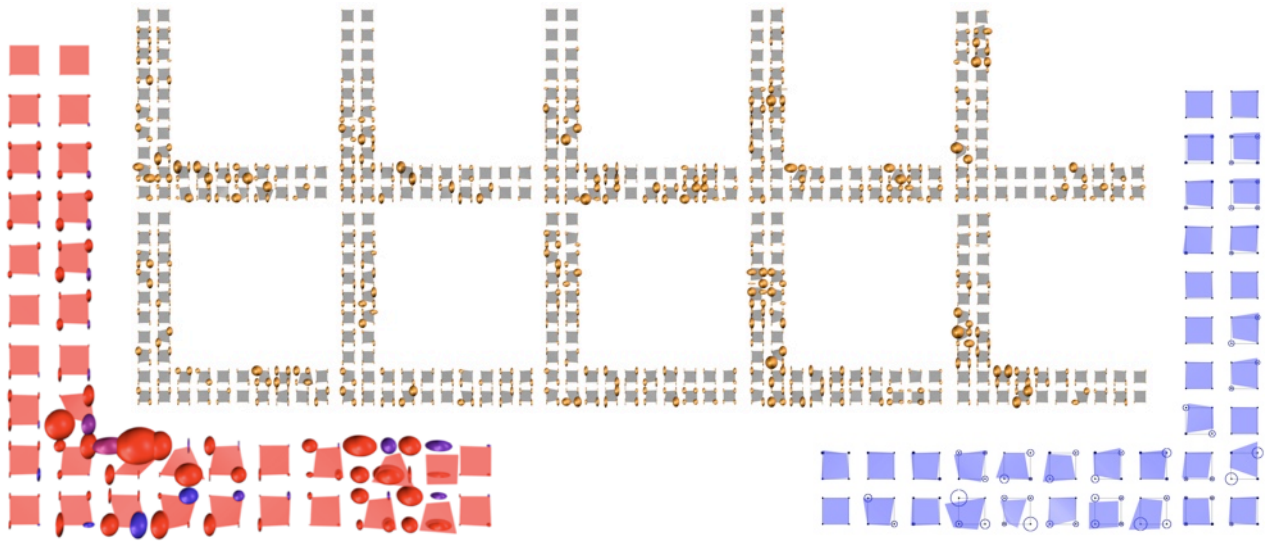
Aaron Knoll, Younis Hijazi, Charles Hansen, Ingo Wald and Hans Hagen
Interactive Ray Tracing of Arbitrary Implicits with SIMD Interval Arithmetic
Proceedings of the 2nd IEEE/EG Symposium on Interactive Ray Tracing, pp. 11-18, 2007

Younis Hijazi and Thomas Breuel
Computing Arrangements using Subdivision and Interval Arithmetic
In P. Chenin, T. Lyche and L.L. Schumaker (Eds.),
Proceedings of the Sixth International Conference on Curves and Surfaces, June 29-July 5, 2006, Avignon, France,
Curve and Surface Design: Avignon 2006, Nashboro Press, pp. 173-182, 2007

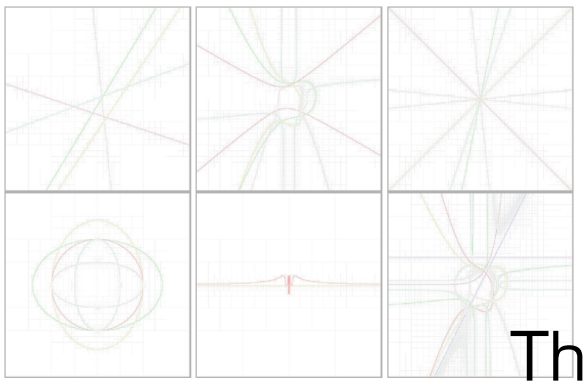
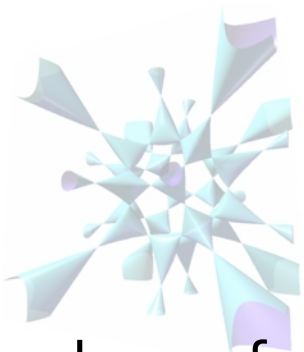
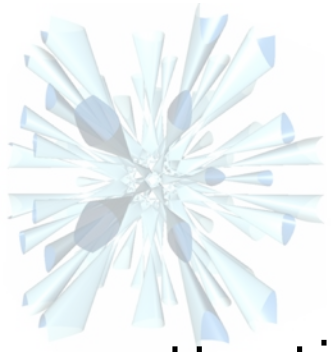
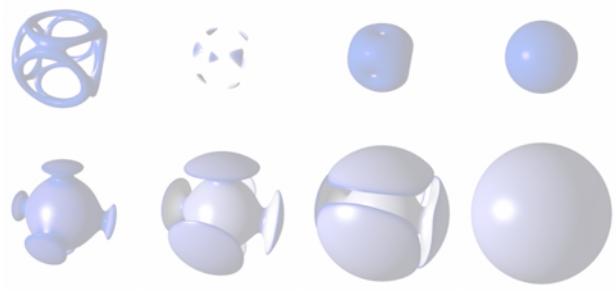
The End

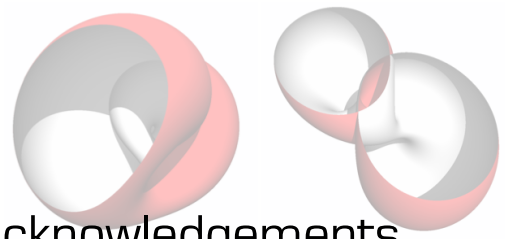





The End

Thank you for your attention.



Acknowledgements

- DFG - IRTG 1131 & University of Kaiserslautern
- UC Davis & University of Utah
- Commission Européenne, PASSPORT-STREP & Université de Strasbourg