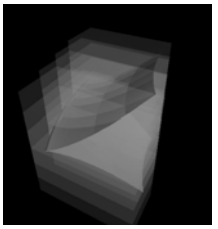
Linear Interval Estimations - A New Kind of Bounding Volumes for Implicit and Parametric Objects

Katja Bühler

VRVis – Research Center for Virtual Reality and Visualization

Vienna, Austria



Seminar on Numerical Software with Result Verification, Dagstuhl 19.-24.1.2003

Bounding Volumes

„ A geometric bound (also called bounding volume) is a simple solid (box, sphere), that contains a given shape S.

Such bounds have been considered to speed up interference detection, rendering and swept volume generation.

Two objects are disjoint if their bounds are.“

(Crosnier, Rossignac, 1999)

Linear Interval Estimations

„A Linear Interval Estimation (LIE) is a linear approximation of the parametric or implicit representation of an object combined with an interval estimation of the approximation error.

A LIE encloses the object.

Two objects are disjoint, if their LIEs are.“

(Bühler 2001)

Overview

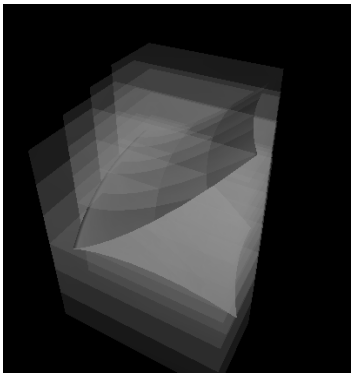
Motivation

Parametric LIEs
Parametric Problems,
Definition, Computation, Application

Implicit LIEs
Implicit Problems,
Definition, Computation, Application

Possible Extensions and Conclusions

Linear Interval Estimations for Parametric Objects



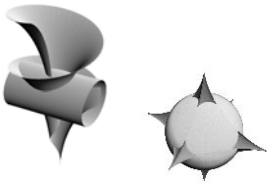
Intersection of two parametric surface patches

Analytical description

$$\mathbf{f}(u,v)=\begin{pmatrix} f_1(u,v) \\ f_2(u,v) \\ f_3(u,v) \end{pmatrix}, \quad (u,v)\in I_u \times I_v \subset \mathbf{R}^2 \quad \text{and}$$

$$\mathbf{g}(s,t)=\begin{pmatrix} g_1(s,t) \\ g_2(s,t) \\ g_3(s,t) \end{pmatrix}, \quad (s,t)\in J_s \times J_t \subset \mathbf{R}^2$$

where $\mathbf{f}\in C^2(\mathbf{I})$, $\mathbf{g}\in C^2(\mathbf{J})$
and $\mathbf{I} := I_u \times I_v$, $\mathbf{J} := J_s \times J_t$



Analytical Solution

Solve the underdetermined constrained equation system

$$f_i(u,v)=g_i(s,t), \quad i=1,\ldots,3$$

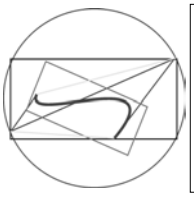
One Application Area for Bounding Volumes of Parametric Objects:

- Simplification of any kind of **intersection detection /computation**
  (surface/surface, ray/surface intersection, collision detection,...)

Most simple solution: Subdivision

- Computation of bounding volumes
- Interference test and intersection of bounding volumes
- Subdivision if possible intersection detected
- Stop subdivision if a certain termination criterion is fulfilled

Common Bounding Volumes for Parametric Objects



- Axes-Aligned Bounding Boxes
- Oriented Bounding Boxes / Parallelepipeds or Slabs
- Polyhedra
- Spheres

Solids!

Computation: Range analysis methods based on

- sampling
- convex hull properties of the control points of special object representations.
- evaluation of derivatives or
- interval or affine arithmetic,

A critical view on existing types of bounding volumes

1. They are either **overestimating** or **difficult to compute and/or to intersect.**
2. Techniques using interval or affine arithmetics to compute bounding volumes ***do not take functional dependencies into account*** or the additional information gets partly lost by conversion.
3. They are **solid**, i.e. they provide only information on the location of the whole object.
4. The ***intersection*** of two bounding volumes gives only information about the location of the intersection of the enclosed objects in object space but not about corresponding values in parameter space!

Ideal Bounding Volumes for Parametric Objects

- Tight
- Reliable

- Easy to compute
- Easy to intersect

AND: The ***intersection of bounding volumes*** should give information about the location of a possible intersection
- in object space *AND*
- in parameter space

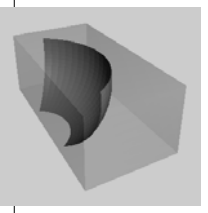
WHY?

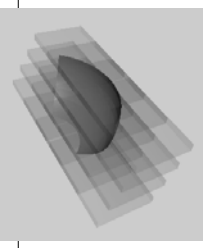
Combination of intersection test and parameter domain reduction

- Acceleration of subdivision processes
- Finding fast good starting points for iterative numerical solutions for the computation of intersection points

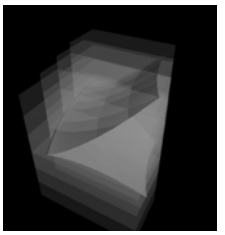
Creating a new type of bounding volume

- **Get tight bounding volumes:**
  Take advantage from **additional information e.g.** provided by **Taylor Models** or the intrinsic structure of **Affine Forms**.
- **Make intersections easy:**
  Choose a **linear structure**
- **Resolve compactness: Parameterize the bounding volume**
  - Provide information about the location of each surface point.
  - Directly connect the parametric representations of the object and its bounding volume.





Linear Interval Estimations (LIEs) for Parametric Objects



Definition :

A linear map $\mathbf{L} : \mathbf{I} \in \mathbf{IR}^m \rightarrow \mathbf{IR}^n$

$$\mathbf{L}(\mathbf{x}^*) = \mathbf{P} + \sum_{i=1}^{m} x_i^* \mathbf{v}_i \; ; \quad \mathbf{x}^* = (x_1^*, ..., x_m^*) \in \mathbf{I}, \quad x_i \in \mathbf{R}$$

with interval vector $\mathbf{P} \in \mathbf{IR}^n$ and $\mathbf{v}_i \in \mathbf{R}^n, \mathrm{i} = 1, ..., \mathrm{m}$

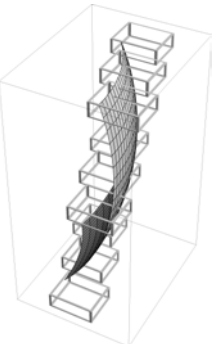
is called a *linear interval estimation* of the parametric object

$\mathbf{f}(\mathbf{x}), \quad \mathbf{x} \in \mathbf{J} \in \mathbf{IR}^m$

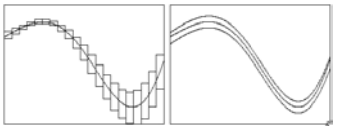
iff there exists a valid reparametrization

$$\Phi : \begin{cases} \mathbf{J} & \rightarrow & \mathbf{I} \\ \mathbf{x} & \mapsto & \Phi(\mathbf{x}) = \mathbf{x}^* \end{cases}$$

so that for all $\mathbf{x} \in \mathbf{J}$ holds $\mathbf{f}(\mathbf{x}) \in \mathbf{L}(\Phi(\mathbf{x})) = \mathbf{L}(\mathbf{x}^*)$.



Computations of LIEs: Taylor Models



Taylor models (Berz 1998) **:**

Let $\mathbf{f} : \mathbf{R} \to \mathbf{R}$ be $C^{n+1}(\mathbf{I})$, $u_0 \in \mathbf{I}$ and $\mathbf{I} \subset \mathbf{R}$ an interval.

Let $\mathbf{T}_{u_0}$ be the Taylor Polynom of $\mathbf{f}$ of order n around $u_0 \in \mathbf{I}$.

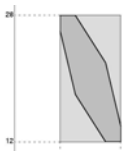
The interval $\mathbf{J} \subset \mathbf{R}$ is called an ***n - th order remainder bound*** of $\mathbf{f}$ on $\mathbf{I}$, iff

$$\mathbf{f}(u) - \mathbf{T}_{u_0}(u) \in \mathbf{J} \qquad \text{for all } u \in \mathbf{I}$$

The pair $(\mathbf{T}_{u_0}, \mathbf{J})$ is called an ***n - th order Taylor model*** of $\mathbf{f}$.

A linear Taylor Model of a parametric representation of an objects is a LIE of this object.

Affine Arithmetic



Arithmetic that operates on *ranges* defined as *affine forms*

Affine Forms

$$\tilde{x} = x_0 + \sum_{i=0}^{n} x_i \varepsilon_i; \qquad \text{Error symbols } \varepsilon_i \in [-1,1], \quad i = 0,\ldots,n$$

Conversion Interval /Affine Form

$$x \in [a,b] \;\rightarrow\; \tilde{x} = x_0 + x_1\varepsilon_1 \quad \text{with} \quad x_0 = \frac{a+b}{2},\; x_1 = \frac{|b-a|}{2},\; \varepsilon_1 \in [-1,1]$$

Operations

- *affine*
- *non-affine*

Addition: $\tilde{x} + \tilde{y} = x_0 + y_0 + \sum_{i=1}^{\max(m,n)} (x_i + y_i)\varepsilon_i$

Multiplication : $\tilde{x} \cdot \tilde{y} = x_0 \cdot y_0 + \sum_{i=0}^{\max(m,n)} (x_0 y_i + y_0 x_i)\varepsilon_i + z_k \varepsilon_k$

Computation of LIEs: Modified Affine Forms

Let be $\mathbf{f}(\mathbf{x})$, $\mathbf{x} \in \prod_{k=1}^{m} I_k \in \mathbf{IR}^m$ a parametric object,

$\tilde{x}_k := x_0^k + x_1^k \varepsilon_k$

the affine forms corresponding to I_k, $k = 1,..., m$ and $\tilde{\mathbf{x}} := (\tilde{x}_1,...., \tilde{x}_m)$.

Then

$$\mathbf{f}(\tilde{\mathbf{x}}) = \tilde{\mathbf{f}}(\varepsilon_1,..., \varepsilon_m, \gamma_1,..., \gamma_n)$$
$$= \mathbf{f}^0 + \sum_{k=1}^{m} \mathbf{f}^k \varepsilon_k + \sum_{i=1}^{n} \mathbf{g}^i \gamma_i; \quad \varepsilon_k, \gamma_i \in [-1,1]$$

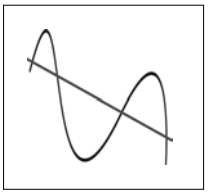
and

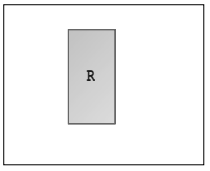
$$\mathbf{L}(\varepsilon_1,..., \varepsilon_m) := \mathbf{P} + \sum_{k=1}^{m} \mathbf{f}^k \varepsilon_k; \quad \varepsilon_k \in [-1,1]$$

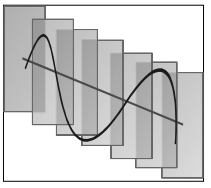
is a *linear interval estimation of* $\mathbf{f(x)}$ where $\mathbf{P} := \mathbf{f}^0 + \left[-\sum_{i=1}^{n} |\mathbf{g}^i|, \sum_{i=1}^{n} |\mathbf{g}^i| \right]$.

General Concept for Constructing LIEs

- Linearization
  + Interval estimation of approximation error
- Parameterization of the LIE has to correspond to parameterization of object.







Intersection of general LIEs in R^n

LIEs are linear objects

Thus,

Intersecting two LIEs or LIEs with lines or (hyper-) planes

=

Solving a constraint system of linear equations

Two LIEs....

$$\mathbf{F}(\mathbf{x}) = \mathbf{F}_0 + \sum_{i=1}^{m} x_i \mathbf{f}_i, \quad \mathbf{x} \in \mathbf{I} \in \mathbf{IR}^m$$

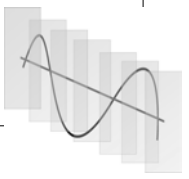
$$\mathbf{G}(\mathbf{y}) = \mathbf{G}_0 + \sum_{j=1}^{k} y_i \mathbf{g}_i, \quad \mathbf{y} \in \mathbf{J} \in \mathbf{IR}^k$$

...and the corresponding intersection equation

$$\mathbf{F}(\mathbf{x}) = \mathbf{G}(\mathbf{x}), \quad \mathbf{x} \in \mathbf{I}, \mathbf{y} \in \mathbf{J}$$

Characterization of LIEs

- **Intuitive and natural** bounding volumes: Linearization of the parametric representation of the patch.
- Each point of the patch is enclosed by an „interval point“ of the LIE with the same parameters.
- Tight and reliable enclosure of the patch.
- Easy to compute.



Furthermore:

- The diameter of the interval part of the LIE gives information about the flatness of the patch: **Good termination criterion**
- Allow a simple intersection test, that gives in addition an enclosure of the solution in the parameter space of the patches.

Example: Ray/Surface intersection

SurfacePatchLIE:

$$\mathbf{L}(u,v) = \mathbf{P} + u\mathbf{y}_1 + v\mathbf{y}_2; \quad (u,v) \in I_u \times I_v$$

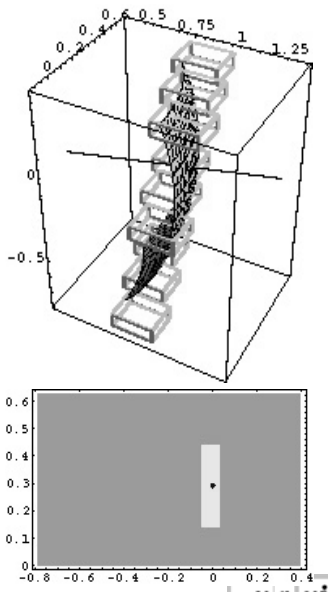
Ray:

$$\mathbf{r}(s) = \mathbf{q} + s\mathbf{w}, \quad s \in I_s$$

Systemof Equations:

$$\begin{pmatrix} \mathbf{y}_1 & \mathbf{y}_2 & -\mathbf{w} \end{pmatrix} \begin{pmatrix} u \\ v \\ s \end{pmatrix} = \mathbf{q} - \mathbf{P}$$

where $u \in I_u, v \in I_v, s \in I_s$

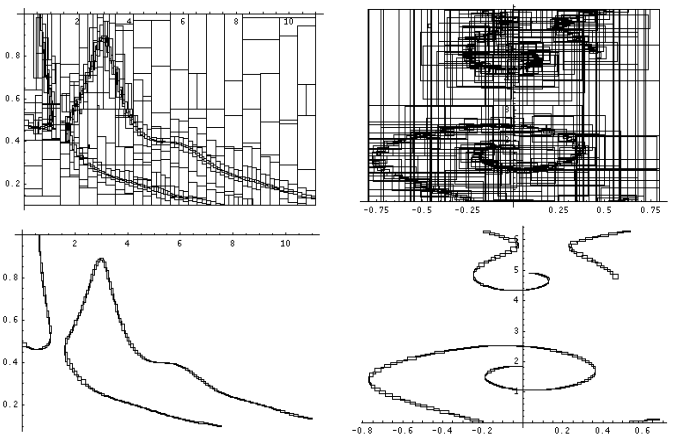


Example: Subdivision Algorithm for Surface Patch Intersection

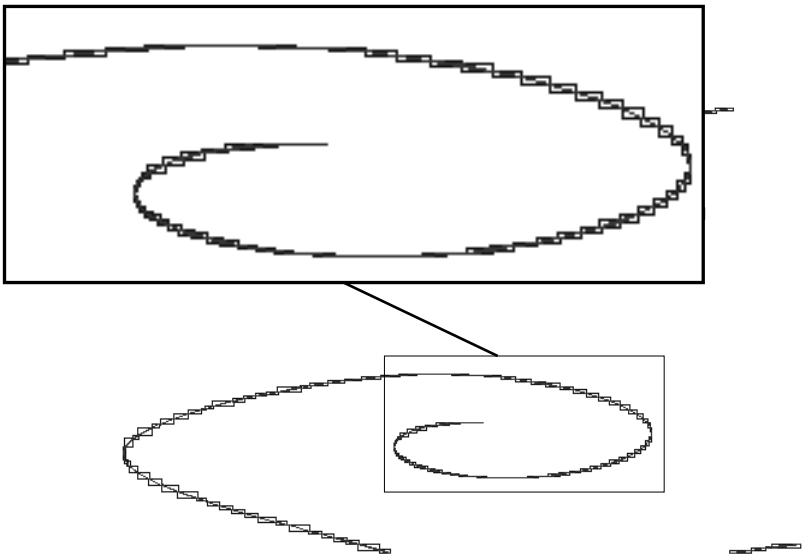
The use of LIEs

- allows to optimize the subdivision algorithm in almost all steps
- reduces the number of subdivisions and intersection tests compared to an subdivision that uses axes aligned bounding boxes dramatically (in a simple example about 1:100)
- reduces the computation time
- reduces the amount of data

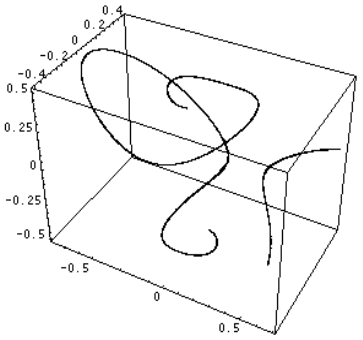
Experimental Results

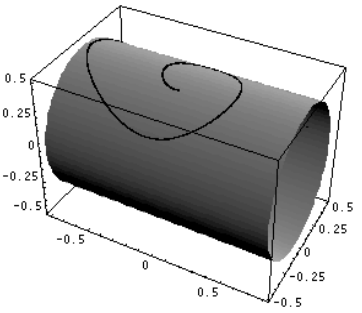


vrvis



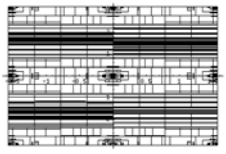
vrvis

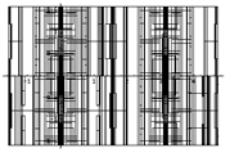


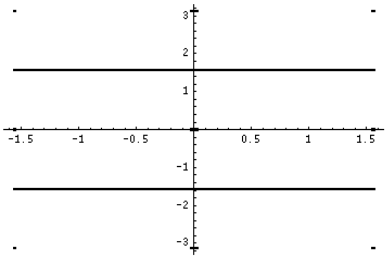


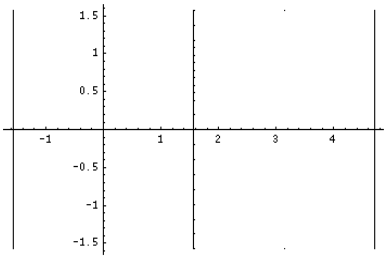
vrvis

Experimental Results 2



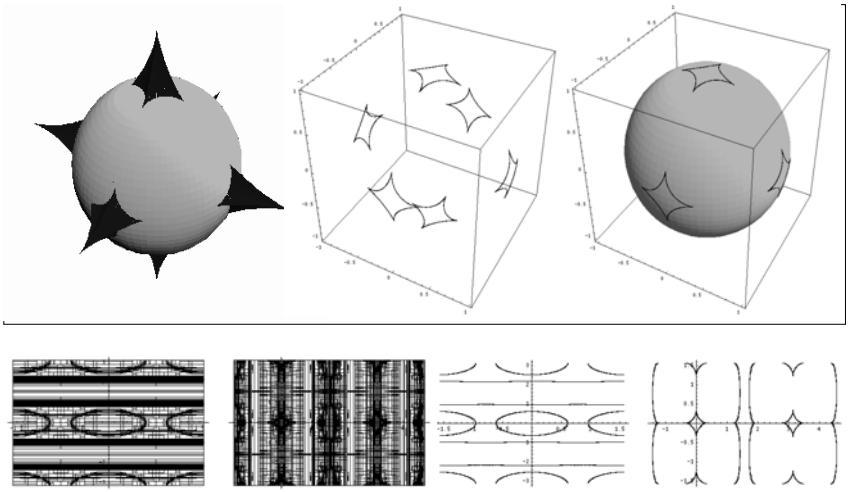




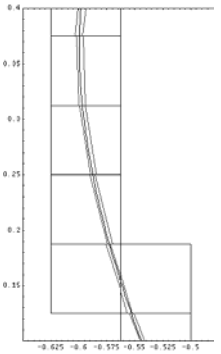


vrvis

Experimental Results 3



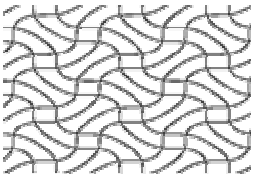
Extension of the Idea to the Implicit Case: Implicit LIEs for Implicit Objects



Implicit Objects - Curves, Surfaces

Definition of an implicit object:

$$F:\ f(\mathbf{x}) = 0, \quad \mathbf{x} \in \mathbf{R}^n$$



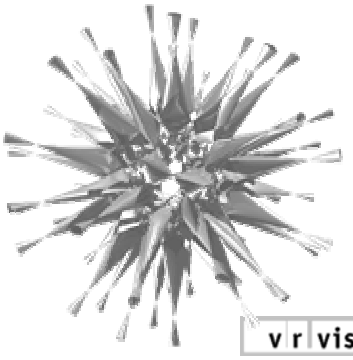
Advantage: Simple location of a point relative to the object:

Let be $\mathbf{p} \in \mathbf{R}^n$:

$f(\mathbf{p}) > 0$, $\mathbf{p}$ outside F

$f(\mathbf{p}) = 0$, $\mathbf{p}$ part of F

$f(\mathbf{p}) < 0$, $\mathbf{p}$ inside F



Problem:

Localization of the object for visualization and collision detection

Why?

- Resolving the implicit equation $f(x_1,...,x_n) = 0$ is in most cases not possible.
- Thus, determining those points in space belonging to the object is non-trivial.

Cell- or ray-object incidence test is an important operation in algorithms related with implicit objects.

Existing Solutions

- *Uniform/Adapted space subdivision*
  - Pixel / Voxel size
  - Basis for polygonization,....
- *Polygonization*
  - Marching squares/cubes,....
- *Ray tracing*
- *Particle systems*
- *Stochastic differential equations*

Possible Application Areas for ILIEs

- *Uniform/Adapted space subdivision (enumeration)*
  - Pixel / Voxel size
  - Basis for polygonization,....
- *Polygonization*
  - Marching squares/cubes,....
- *Ray tracing*
- *Particle systems*
- *Stochastic differential systems*

Possible Application Areas for ILIEs

- *Uniform/Adapted space subdivision (enumeration)*
  - Pixel / Voxel size
  - Basis for polygonization,....
- *Polygonization*
  - Marching squares/cubes,....
- *Ray tracing*
- *Particle systems*
- *Stochastic differential systems*

Example

Linear Interval Estimations (ILIEs) for Implicit Objects

Definition :

The interval (hyper-)plane

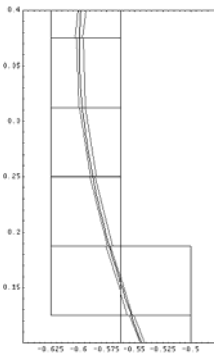
$$0 \in L(\mathbf{x}) := J + \sum_{i=1}^{n} a_i x_i \quad , \mathbf{x} := (x_1,...,x_n) \in \mathbf{R}^n$$

with $J \in \mathbf{IR}$, $a_i \in \mathbf{R}, i = 1,..,n$ is called an *implicit linear interval estimation* of

$F : f(\mathbf{x}) = 0$ on $\mathbf{I} \in \mathbf{IR}^n$

iff for all $\mathbf{x} \in F$ holds

$0 \in L(\mathbf{x})$.



The picture shows an implicit curve piecewise enclosed by ILIEs *(0 ∈ 2x + 6y + [-0.1,0.2])* defined on the blue boxes.

Computation of ILIEs

General Recipe:

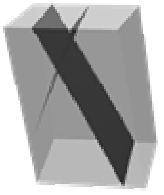
1. Compute any linear approximation $l_f(\boldsymbol{x})$ of $f(\boldsymbol{x})$ on a cell $\boldsymbol{I}$.
2. Estimate the approximation error with an interval J.
3. Combine both to get an ILIE of F:
   $L_F: \ 0 \ \ in \ \ l_f(\boldsymbol{x}) + J$

Here:
Computation of ILIEs using modified affine arithmetics

Other possibilities: Taylor Models,....

Characteristics

- *Fat linearization*
- Diameter can be used as *criterion for flatness*.
- *Low additional computational costs* compared to a cell/surface evaluation if affine arithmetic is used.
- *No Problems with singularities* if computed with affine arithmetic.
- *Tight enclosure* due to Tchebycheff approximation



Cell pruning

- The ILIE encloses the object in many cases much tighter than the corresponding cell.
- The axes aligned cell can be reduced easily to those parts containing the ILIE / object using interval arithmetics.

Iterative pruning:

- A new ILIE can be computed with respect to the pruned cell that can be pruned again with respect to the new cell,....

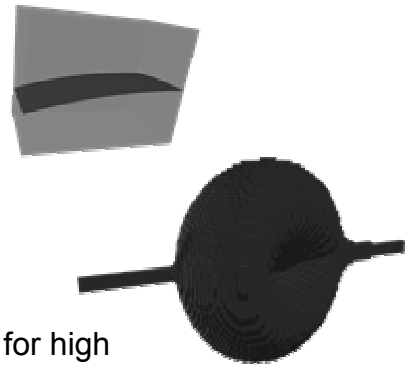
The Classical Enumeration Algorithm Based on Interval Arithmetic

Input: Implicit object *obj* defined by *obj.f(x) = 0*, an initial cell I.
Output: Set of cells that may contain the object.

```
Algorithm Enumerate(ImplObj c, Cell I)
    if  (0 not in obj.f(I))
       return;                          // Test, if box may hit the curve.
    if (termination criterion fulfilled)
       return;                          // Termination criterion fulfilled?
    subdivide I into subboxes b_i, i=1,..,2^n;
    for i=1,..., 2^n
          I := I_i ;
          Enumerate(c, I_i );           // Perform algorithm for new parts
```

Start algorithm with Enumerate(obj, I);

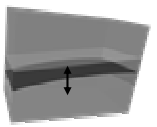
Analysis

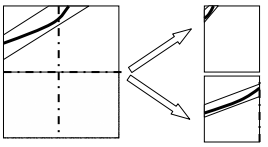


- Incidence test
  0 in obj.f(I) ?
  is expensive.
- No cell pruning.
- Axes aligned enclosures.
- High number of subdivisions for high precision.
- High number of cells representing the result.

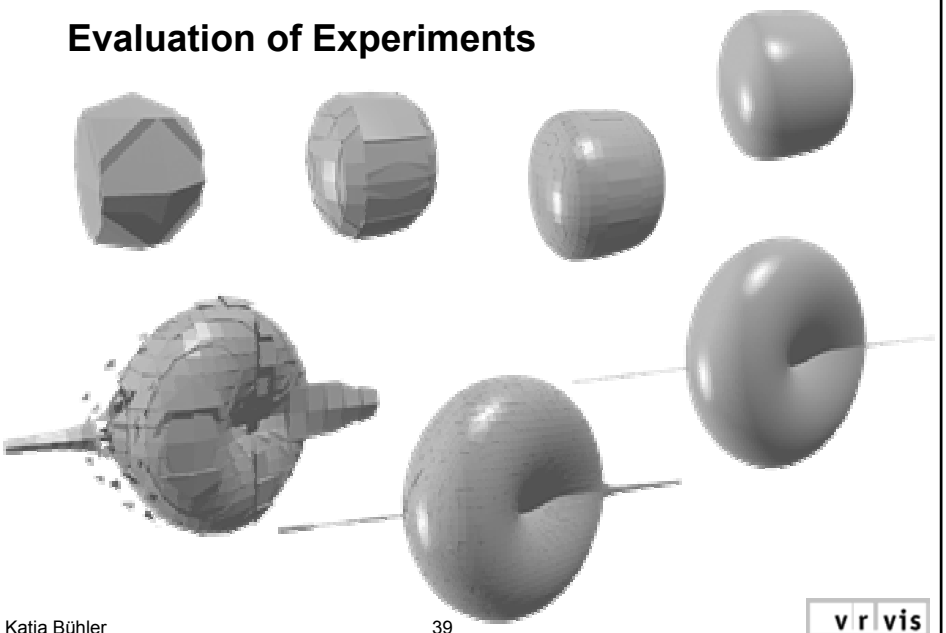
Possible Improvements Using ILIEs

1. Perform cell–object incidence test and computation ILIE in (almost) one step.
2. Use diameter of ILIE as termination criterion.
3. Perform (iterated) cell pruning for each computed sub-cell.
4. Reduce unnecessary cell–object tests doing a pre-test with the ILIE of the mother cell.
5. Apply adapted subdivision strategies.

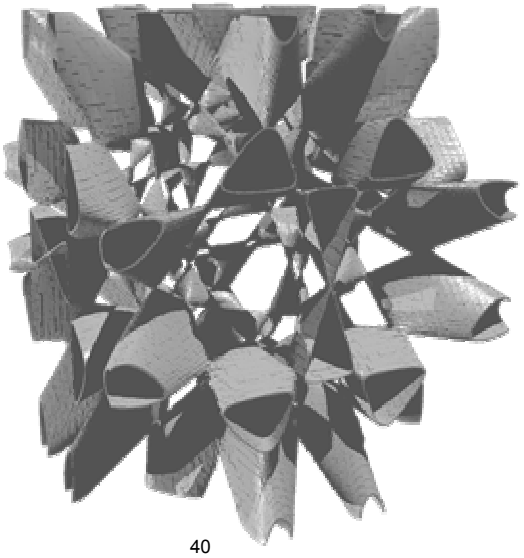




Evaluation of Experiments



Evaluation of Experiments

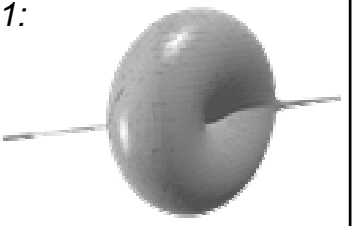


Comparing ILIE-Based Algorithm With Axes-Aligned Cell-Based Algorithm

Much less subdivisions and ILIEs are necessary to represent an enclosure of certain precision than using axes-aligned cells.

Example: *Cross Cap, Precision 0.01:*

140 times less subdivisions
160 times less ILIEs than cells
(70 times faster)



So……

- Introduction of ILIEs allows a redefinition of classical enumeration algorithms.
- ILIEs provide in many cases better enclosures, than axes-aligned cells.
- Results are much better adapted to the topology of the object.
- The number of necessary subdivisions decreased dramatically.
- The number of necessary ILIEs to represent a result with a certain precision is dramatically less using ILIEs than axes-aligned cells
  - As a basis for polygonization: much less polygons are necessary.
  - As a basis for collision detection: much less interference tests are necessary.
  - As a basis for ray tracing: Ray/plane test very fast with unique result.

Conclusions and Future Work

Conclusions

- LIEs implement a new philosophy of bounding volumes for geometric objects
  - natural and intuitive
  - directly connected to the representation of the enclosed object
  - tight and reliable
  - easy to compute and to intersect
  - allow development of adaptive algorithms, reduction of the number of necessary steps, and the size of output
  - accelerate basic algorithms

Future Work

- What basics are needed ?
  - Implementation of a better Affine Arithmetic ↘ Alternatives?
- Possible applications to implement:
  - Direct ray tracing of parametric / implicit surfaces
    → Nate Hayes, Sunfish Studios
  - Natural Bounding Volumes for Parametric Volumes
  - Collision Detection
- Extensions:
  - LIEs for triangular parametric surface patches
  - Application of higher order Taylor Models?
  - Exploration of other linearization techniques?

Contact:

Katja Bühler
VRVis- Research Center for Virtual Reality and Visualization

Vienna, Austria

katja@cg.tuwien.ac.at

http://www.vrvis.at