

Implicit Linear Interval Estimations

Katja Bühler

Institute of Computer Graphics and Algorithms, Vienna University of Technology

Abstract

Visualization and collision detection are two of the most important problems connected with implicit objects. Enumeration algorithms can be used either directly or as preprocessing step for many algorithms solving these problems. In general, enumeration algorithms based on recursive space subdivision are reliable tools to encounter those parts in space, where the object might be located. But the bad performance and the huge number of computed enclosing cells, if high precision is required, are grave drawbacks. Implicit Linear Interval Estimations (ILIEs) introduced in this paper are implicit interval (hyper-)planes providing oriented tight bounds of the object within given cells. It turns out that the use of ILIEs highly improves the performance of the classical enumeration algorithm and the quality of the results. The theoretical background as well as a fast and simple technique to compute ILIEs are presented. The applicability of ILIEs is demonstrated by means of a modified enumeration algorithm that has been implemented and tested for implicit surfaces.

Keywords: implicit surfaces; implicit curves; enumeration; rendering; collision detection; affine arithmetic; interval arithmetic; linear interval estimation;

1 Introduction

Using implicit equations to describe a geometric object builds a powerful tool for the representation of curves, surfaces, and volumes in computer graphics. Besides of the description of mathematical, physical, geological, and other scientific phenomena, implicit surfaces and volumes are mainly used in CSG-Systems to design complex objects by adding, subtracting, and inverting several smooth surfaces. Implicit surfaces used in CSG-modellers are mainly planes and quadrics. The SvLis geometric modeller [5] includes general algebraic surfaces. Recently skeletal and blobby surfaces have been added to the CSG-tree by Wyvill, Guy and Galin [32]. These types of implicit surfaces allow a very compact description of complex objects. Smooth blending, warping, and deformation of objects are proposed as supplementary operations.

An implicit defined object in \mathbf{E}^n is given by an equation of the form

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

where f can be either a polynomial or any other real

valued function. The implicit representation has the advantage that it allows a fast test whether a point lies inside ($f(\mathbf{x}) > 0$), outside ($f(\mathbf{x}) < 0$) or on ($f(\mathbf{x}) = 0$) the object. Besides of many other positive features of implicit descriptions of objects, there is one main drawback: The points building the object are defined as zero-set of f - an equation which is in general not explicitly resolvable. The computation of an approximation of this zero-set for visualization and collision detection is topic of many publications of the last two decades, finding fast and guaranteed reliable solutions is one of the subjects of recent research [22, 23, 31].

The most expensive way to visualize an implicit object is direct ray tracing [18] - especially for objects with high algebraic degree. Other solutions are based on enumeration algorithms [3, 6, 7, 11, 12, 22, 30, 31], marching squares/cubes/triangles [1, 14, 27], particle systems [10, 17], or stochastic differential equations [25]. These methods can be used directly for visualization, as preprocessing step for further rendering, or polygonization of the object. A wide variety of solutions exists also for the collision detection problem of implicit objects. Solving an equation system is an expensive possibility to handle this problem [20], polygonizing the objects before the collision detection process is not reliable due to the error introduced by the approximation. Using axes-aligned boxes as piecewise enclosures of the object produced by the classical enumeration algorithm as bounding volumes presents an often used alternative to the two solutions mentioned before [28].

All visualization and collision detection algorithms based on a discretization of the object suffer from uncertainties. Depending on the precision, important features like singularities may be missed. Enumeration algorithms are based on recursive adaptive space subdivision combined with an incidence test of axes-aligned boxes (cells) and the object. If the reliability of the object-cell incidence test is guaranteed, the result is in general a reliable enclosure of the object.

Related work. The most simple enumeration algorithms are based on recursive *uniform space subdivision*. These algorithms do not take the topology of the object to be detected into account. Some algorithms subdivide until pixel size is reached to voxelize the object [6, 12, 21, 31], others follow a hybrid approach and compute for each de-

tected cell a linear approximation of the object.

Adaptive space subdivision techniques take the curvature of the object during the subdivision process into account, followed by a linearization like in the hybrid uniform case. Adaptive techniques are faster, but in the case of implicit surfaces cracks could appear in the polygonal approximation due to different levels of subdivision of neighbouring cells. Bloomenthal [4] solved the problem for cracks caused by neighbouring cells differing one level of depth in subdivision. Recently Balsys and Suffern [3] presented an adaptive algorithm solving the problem for an arbitrary level of subdivision.

As mentioned before, the heart of each subdivision algorithm is the *reliable test* if the actual box to be examined hits the object or not. The reliability of the incidence test is an important requirement, because it is the condition for the reliability of further computations. Taubin [26] developed for his “accurate algorithm for rastering algebraic curves” an approximation of the distance of the midpoint of a cell and the curve to determine whether the cell hits the curve or not. The majority of recent published algorithms perform the test in a reliable way using interval arithmetics as a tool for range analysis [6, 12, 19, 24]. $f(\mathbf{x})$ is simply evaluated with respect to the interval vector corresponding to the axes-aligned box to be tested. If the resulting interval contains zero, the box may contain a part of the object and further subdivision is performed. To reduce overestimations caused by interval arithmetic, De Figueiredo and Stolfi [11] replace in their algorithm interval arithmetic by affine arithmetic. Voiculescu et al. [31] introduce two further methods to improve the results in the case of algebraic curves and surfaces: They show that a reformulation of the equation into Bernstein-Bézier form and/or the use of a modified affine arithmetic [16] improves the result and the performance of the subdivision algorithm.

The algorithms based on interval or affine arithmetic mentioned above perform the incidence test with the original curve or surface that might be of high algebraic degree or have a non-trivial non-algebraic description. It is clear that as higher the degree and as more complicate the description of the object is, as more expensive becomes the validation of the incidence test.

Furthermore, pure subdivision algorithms suffer from the high number of necessary subdivisions to reach pixel size. An adaptive subdivision algorithm loses its reliability due to the approximation implied in the linearisations representing the final results.

Aims and scopes of this paper. The purpose of the work presented here is to improve the classical enumeration algorithm in a way that it computes fast an optimized number of tight, oriented, piecewise linear, and reliable enclosures for an implicit object in \mathbf{R}^n .

To do so, Implicit Linear Interval Estimations (ILIEs)

for general implicit objects are introduced following the same construction principles like proposed by the author for the parametric case [9]. ILIEs are implicitly defined interval (hyper-)planes providing for each cell a piecewise linear enclosure of the object adapted to its topology. They allow to reduce the cost for cell/object incidence tests during the subdivision process and to decrease the number of subdivisions by reducing the cell to parts containing the corresponding ILIE. Finally, the diameter of an ILIE informs about the curvature of the object inside the corresponding cell. The resulting ILIEs could be used for polygonization and the acceleration of a rendering algorithm.

ILIEs for curves based on affine arithmetics have been already introduced to provide a reliable and fast plotting tool [8]. This paper is an extension of the described techniques to general implicit objects and shows examples for implicit surfaces in 3-space.

Structure. A definition of ILIEs for general implicit objects in \mathbf{R}^n is given in section 2. In the same section a method to compute ILIEs based on affine arithmetic is developed and a characterization is presented. The theoretical ideas of the previous section are applied in section 3 to improve the classical enumeration algorithm followed by a discussion of experimental results in section 4. The paper is closed with conclusions and ideas for future work. Furthermore, appendix A contains a short introduction into interval and affine arithmetic, appendix B the formulas for the examples in section 4.

Notations. In the following, \mathbf{R} denotes the set of real numbers and \mathbf{IR} the set of intervals. Furthermore, if there is no other declaration, thin small letters ($u \in \mathbf{R}$) denote real scalars, thin capital letters ($I_u \in \mathbf{IR}$) intervals, bold letters ($\mathbf{x} \in \mathbf{R}^n$) real vectors, and hollow letters ($\mathbb{I} \in \mathbf{IR}^n$) interval vectors representing axes-aligned boxes, also often referred to as *cell*. Affine forms and vectors of affine forms are marked with a hat ($\hat{u}, \hat{\mathbf{f}}$).

2 Implicit Linear Interval Estimations (ILIEs) for general implicit objects in \mathbf{R}^n

This section requires some basic knowledge about interval and affine arithmetic. A short introduction can be found in appendix A.

Definition Let be $\mathcal{F} : f(\mathbf{x}) = 0$, $\mathbf{x} = (x_1, \dots, x_n)^T \in \mathbf{R}^n$ the implicit definition of an object in \mathbf{R}^n and

$$L(\mathbf{x}) := \sum_{i=1}^n a_i x_i + J \quad (1)$$

with $J \in \mathbf{IR}$ and $a_i \in \mathbf{R}, i = 1, \dots, n$.

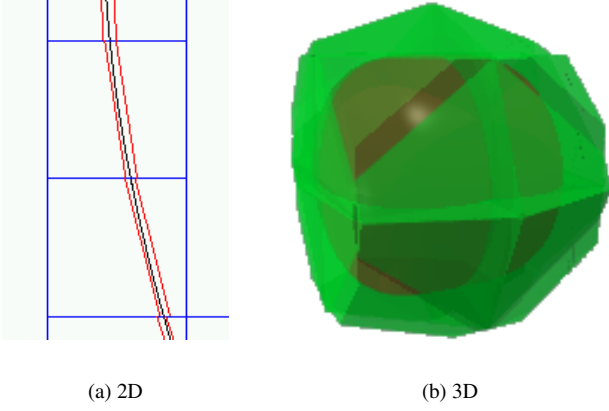


Figure 1: Examples for ILIEs.

The interval hyperplane segment inside the axes aligned box $\mathbb{I} \in \mathbb{IR}^n$

$$\mathcal{L} := \{\mathbf{x} \in \mathbb{I} \mid 0 \in L(\mathbf{x})\}$$

is called Implicit Linear Interval Estimation (ILIE) of \mathcal{F} on \mathbb{I} , iff for all $\mathbf{x} \in (\mathcal{F} \cap \mathbb{I})$ holds

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

For example, an ILIE of a curve on a square $\mathbb{I} \subset \mathbb{R}^2$ is a fat line segment enclosing inside \mathbb{I} all points of the curve: Figure 1(a) shows the enclosing ILIEs of a curve corresponding to some given boxes, figure 1(b) shows a surface of degree four enclosed by 32 ILIEs.

Computation of ILIEs exploiting the intrinsic structure of affine forms One way to compute ILIEs in an effective way is to use affine arithmetic: Like in the straight forward use of affine arithmetic for cell/object incidence tests [7, 11, 31], $f(\mathbf{x})$ is evaluated with respect to the vector of affine forms $\hat{\mathbf{I}}$ corresponding to \mathbb{I} . In the algorithms of de Figueiredo et al. [11] and Bowyer et al. [7, 31], the resulting affine form \hat{f} is converted back into an interval to determine whether zero is contained in the resulting interval. The conversion of the result from affine form to interval destroys all additional information included in the affine form. In this section this additional information is used to compute ILIEs with almost no additional cost. The following theorem gives the theoretical background for the computation of ILIEs for implicit objects in n-space.

Theorem *Let be*

- $\mathcal{F} : f(\mathbf{x}) = 0$ an implicit object in \mathbb{E}^n
- $\mathbb{I} = \prod_{i=1}^n I_i \subset \mathbb{R}^n$ a non-degenerated interval box
- $\hat{x}_i = \hat{x}_i(\epsilon_i) = x_i^0 + x_i^1 \epsilon_i$, $\epsilon_i \in [-1, 1]$, the corresponding affine form to interval I_i ; $i = 1, \dots, n$.

Furthermore let be

- for $\gamma_j \in [-1, 1]$, $j = 1, \dots, m$

$$\begin{aligned} \hat{f}(\epsilon_1, \dots, \epsilon_n, \gamma_1, \dots, \gamma_m) &:= \hat{f}(\hat{\mathbf{x}}) \\ &= f^0 + \sum_{i=1}^n f^i \epsilon_i + \sum_{j=1}^m g^j \gamma_j \end{aligned}$$

Define

- for $\mathbf{x} \in \mathbb{I}$ and $\epsilon_i = \epsilon_i(x_i) := \frac{1}{x_i^1}(x_i - x_i^0)$, $i = 1, \dots, n$

$$\begin{aligned} L(\mathbf{x}) &:= \hat{f}(\epsilon_1(x_1), \dots, \epsilon_n(x_n), [-1, 1], \dots, [-1, 1]) \\ &= J + \sum_{i=1}^n f^i \frac{1}{x_i^1} x_i \end{aligned}$$

with

$$J := f^0 - \sum_{i=1}^n \frac{x_i^0}{x_i^1} f^i + \left[-\sum_{j=1}^m |g^j|, \sum_{j=1}^m |g^j| \right].$$

Then

$$\mathcal{L} := \{\mathbf{x} \in \mathbb{I} \mid 0 \in L(\mathbf{x})\}$$

is a linear interval estimation of \mathcal{F} on \mathbb{I} .

Proof: All conditions and definitions of the theorem are presumed.

It follows from the definition of affine forms, that the relation of \mathbf{x} and $(\epsilon_1, \dots, \epsilon_n)$

$$\begin{aligned} \mathbb{I} \in \mathbb{IR}^n &\rightarrow [-1, 1]^n \\ \mathbf{x}^T &\mapsto \left(\frac{1}{x_1^1}(x_1 - x_1^0), \dots, \frac{1}{x_n^1}(x_n - x_n^0) \right)^T \\ &=: (\epsilon_1, \dots, \epsilon_n)^T \end{aligned}$$

is a bijection iff \mathbb{I} is not degenerated.

The definition of affine arithmetic guarantees that for every point $\mathbf{x}^T \in \mathbb{I}$ there exist $\epsilon'_i, \gamma'_j \in [-1, 1]$, $i = 1, \dots, n, j = 1, \dots, m$, so that

$$f(\mathbf{x}') \equiv \hat{f}(\epsilon'_1, \dots, \epsilon'_n, \gamma'_1, \dots, \gamma'_m)$$

Furthermore, it follows from the inclusion property of interval arithmetics that

$$\begin{aligned} f(\mathbf{x}') \in L_\epsilon(\epsilon'_1, \dots, \epsilon'_n) \\ := \hat{f}(\epsilon'_1, \dots, \epsilon'_n, [-1, 1], \dots, [-1, 1]) \end{aligned} \quad (2)$$

The bijective relation of affine forms and intervals allows to rewrite L_ϵ with respect to the coordinates \mathbf{x}

$$\begin{aligned} L(\mathbf{x}) &:= L_\epsilon\left(\frac{1}{x_1^1}(x_1 - x_1^0), \dots, \frac{1}{x_n^1}(x_n - x_n^0)\right) \\ &= f^0 + \sum_{i=1}^n f^i \frac{1}{x_i^1} (x_i - x_i^0) \\ &\quad + \left[-\sum_{j=1}^m |g^j|, \sum_{j=1}^m |g^j| \right] \\ &= J + \sum_{i=1}^n f^i \frac{1}{x_i^1} x_i \end{aligned}$$

with $J := f^0 - \sum_{i=1}^n \frac{x_i^0}{x_i} f^i + [-\sum_{j=1}^m |g^j|, \sum_{j=1}^m |g^j|]$.

Now, (2) can be rewritten as

$$f(\mathbf{x}') \in L(\mathbf{x}') \quad (3)$$

$L(\mathbf{x})$ is of the form $\sum_{i=1}^n a_i x_i + J$, $a_i \in \mathbf{R}$, $J \in \mathbf{IR}$ and from equation (3) follows for all $\mathbf{x} \in \mathbb{I}$ with $f(\mathbf{x}) = 0$ that $0 \in L(\mathbf{x})$.

Thus, by definition, $\mathcal{L} = \{\mathbf{x} \in \mathbb{I} \mid 0 \in L(\mathbf{x})\}$ is a linear interval estimation of $\mathcal{F} : f(\mathbf{x}) = 0$, $\mathbf{x} \in \mathbb{I}$. \square

Modus operandi for the computation of ILIEs:

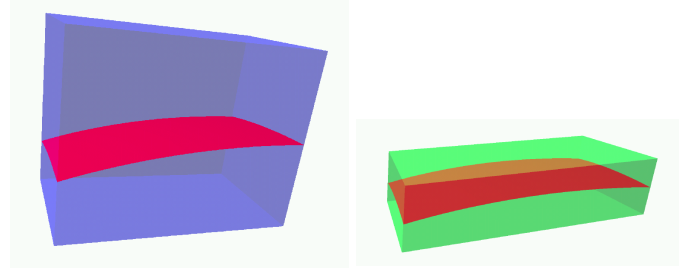
Given an implicit object $\mathcal{F} : f(\mathbf{x}) = 0$ and a cell $\mathbb{I} = \prod_{i=1}^n I_i$. The following steps give an algorithm to compute $J \in \mathbf{IR}$ and $a_i \in \mathbf{R}$, $i = 1, \dots, n$ determining the corresponding ILIE

$$\mathcal{L} := \{\mathbf{x} \in \mathbb{I} \mid 0 \in J + \sum_{i=0}^n a_i x_i\}$$

1. Compute the n affine forms \hat{x}_i , $i = 1, \dots, n$ corresponding to I_i , $i = 1, \dots, n$
2. Record the set of indices $\mathcal{I}_k := \{k_i \in \mathbf{N} \mid k_i \neq k_j, i, j = 1, \dots, n\}$ corresponding to the errors symbols of $\hat{x}_i = x_i^0 + x_i^1 \epsilon_{k_i}$, $i = 1, \dots, n$.
3. Compute $f(\hat{\mathbf{x}}) = \hat{f}(\epsilon_{k_1}, \dots, \epsilon_{k_n}, \epsilon_{l_1}, \dots, \epsilon_{l_m}) = f^0 + \sum_{i=1}^n f^{k_i} \epsilon_{k_i} + \sum_{j=1}^m f^{l_j} \epsilon_{l_j}$, where $\mathcal{I}_l := \{l_j \in \mathbf{N} \mid l_j \neq l_i, i, j = 1, \dots, m\}$ is the set of indices of error symbols generated during the evaluation of f . Note that \mathcal{I}_k and \mathcal{I}_l are disjoint.
4. Set $J := [-\sum_{j=1}^m |f^{l_j}|, \sum_{j=1}^m |f^{l_j}|] + f^0 - \sum_{i=1}^n \frac{x_i^0}{x_i} f^{k_i}$.
5. Set $a_i := \frac{1}{x_i} f^{k_i}$, $i = 1, \dots, n$.

Characterizations ILIEs provide linear, tight, simple, and oriented enclosures for implicit objects inside a given cell. Figures 2 (a) and (b) illustrate the difference of an enclosure by a cell and the corresponding ILIE.

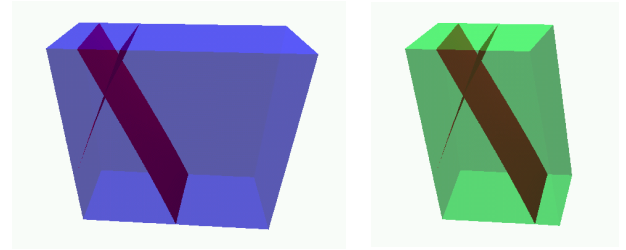
1. An ILIE can be interpreted as “fat” linearisation of the object inside a certain range that encloses the object. Thus, an ILIE is a bounding volume of the object inside the corresponding cell.
2. Furthermore, the interval part of the ILIE gives reliable information about the deviation of the enclosed object: The diameter $d := \frac{1}{\|(a_1, \dots, a_n)^T\|} \text{diam}(J)$ of the ILIE $\mathcal{L} : \sum_{i=1}^n a_i x_i + J = 0$, $J \in \mathbf{IR}$ can be used as measure for the curvature.
3. Using affine arithmetic, an ILIE can be computed with almost no additional cost.



(a) Enclosing cell

(b) Corresponding ILIE

Figure 2: Comparison of enclosures of an implicit surface patch in a given cell.



(a) Enclosing cell

(b) Corresponding ILIE

Figure 3: An implicit surface patch of degree 4 with singularities.

4. The described method to compute ILIEs works well even if the surface has singularities inside the given cell. (See figure 3.)

Pruning the corresponding cell to relevant parts

The ILIE provides in most cases a tighter enclosure of the surface inside its corresponding cell, than the cell itself. Having in mind the aim to improve the enumeration procedure, pruning the cell to relevant parts, i.e. parts containing the ILIE, could be useful with respect to the reduction of necessary subdivisions (Compare figures 2 (a) and figure 4 (a)). Again interval arithmetic turns out to be a handy tool to develop a simple algorithm for this purpose:

Given an ILIE \mathcal{L} on $\mathbb{I} \in \mathbf{IR}^n$ like described in equation (1), the intervals I_i^* defining the pruned cell $\mathbb{I}^* = \prod_{i=1}^n I_i^*$ can be computed using the following equations:

$$I_i^* = \left(-\frac{1}{a_i} \left(\sum_{\substack{k=1 \\ k \neq i}}^n a_k I_k + J\right)\right) \cap I_i; \quad i = 1, \dots, n$$

Remark: An iterative pruning of the cell might be reasonable in cases where big parts of the cell have been

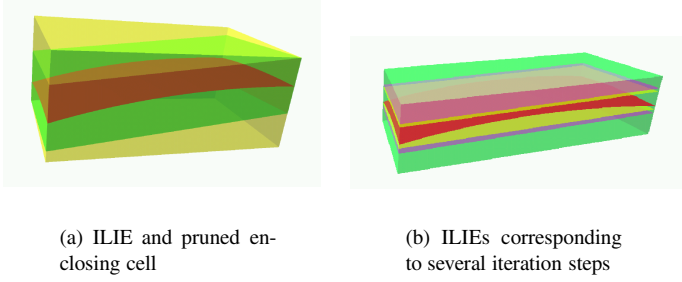


Figure 4: (a) shows the ILIE already presented in figure 2 inside the pruned cell. Figure (b) shows three nested ILIEs that are results of iterated cell pruning.

pruned off: A recalculation of the ILIE with respect to the new cell will result in a tighter and better fitting ILIE for the enclosed part of the surface. Figure 4 (b) visualizes such an iteration process producing in each step a tighter ILIE. As affine arithmetic uses min-max or Chebycheff approximation to approximate non-affine operations, the iteration described above leads to a well oriented and tight ILIE.

3 Using ILIEs to improve the classical enumeration algorithm

A classical enumeration algorithm based on adaptive subdivision and interval arithmetic has the following form:

Input: An implicit object obj defined by the equation $obj.f(\mathbf{x}) = 0$ inside the cell I (an axes-aligned box).
Output: Set of cells enclosing the object.

```

Algorithm Enumerate ( ImplObj obj, Cell I )
  if ( 0 not in obj.f(I) )
    return; //Stop if cell does not hit the object
  if (termination criterion fulfilled)
    write results; return;
  subdivide I into  $2^n$  sub-cells,  $I_k, k = 1, \dots, 2^n$ ;
  for  $k = 1, \dots, 2^n$ 
    Enumerate( ImplObj( obj.f, I_k ));
    // Perform test for new parts

```

Analysis of the algorithm The algorithm performs in each step a cell-object incidence test evaluating the expression $f(\mathbb{I})$ using interval or affine arithmetic. If zero lies inside the resulting interval, the cell might hit the object and further subdivision is performed until a certain precision is reached. This algorithm has several weak points, that could be used as motivation for improvements.

1. The incidence test with the original object is a very expensive operation due to overestimations caused by interval and affine arithmetics, especially if the object is of high algebraic degree.
2. There is no mechanism to prune the cells in a way that only the relevant part of the cell is used for further subdivision. This could reduce the overall number of necessary subdivisions and enlarge the adaptive effect of the algorithm.
3. The result of the algorithm is a set of axes-aligned cells, that are not adapted to the topology of the enclosed part of the object. An axes aligned box contains no information about the position of the enclosed surface which may lead to ambiguities if the results are used for the polygonization of implicit surfaces.
4. The number of necessary subdivisions and cells representing the result increases dramatically if high precision results are required.

Improving the algorithm The classical enumeration algorithm mentioned above can be improved significantly with respect to the number of necessary subdivisions, time consumption, and the quality of results using ILIEs.

The algorithm is modified in the following way: Let be $f(\mathbf{x}) = 0$ the expression describing the implicit object \mathcal{F} , \mathbb{I} the cell to be examined and $\hat{\mathbf{x}}$ the corresponding vector of affine forms.

- Each time $f(\mathbb{I})$ is evaluated using affine arithmetic to determine if cell \mathbb{I} may hit the object or not, the description for the ILIE corresponding to \mathbb{I} and \mathcal{F} is derived from the resulting affine form $\hat{f}(\epsilon_1, \dots, \epsilon_n, \gamma_1, \dots, \gamma_m) = \hat{f}(\hat{\mathbf{x}})$ following the given theorem in section 2
- The diameter of the ILIE is used as termination criterion as it represents a good estimation for the grade of flatness of the implicit patch.
- To avoid unnecessary direct cell-object test, each of the sub-cells resulting from a subdivision of the initial cell is first tested, if it hits the ILIE corresponding to its “mother cell”. Only cells passing this test in a positive way will be used for further computations. (See figure 5.)
- The result of the whole subdivision process is, besides of the set of computed cells that may contain a part of the implicit surface, a set of ILIEs, that provides a tight piecewise linear interval enclosure of the surface.
- The application of ILIEs combined with cell pruning allows to implement new subdivision strategies that also reduce the amount of necessary subdivisions:

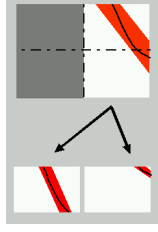


Figure 5: Avoiding unnecessary object-cell tests using ILIEs.

Most known enumeration algorithms perform a uniform space subdivision technique. In general each cell is subdivided into eight equal-sized sub-cells regardless of the topology of the (maybe) enclosed object. Cell pruning based on ILIEs allows to implement for example an adapted *binary* subdivision: The pruned cell is subdivided only once along its longest edge.

A modified algorithm following the propositions above is listed below. The two lines marked with stars can be repeated to perform iterated cell pruning.

Input: Implicit object obj including its implicit description $obj.f$, an initial cell I .

Output: Set of linear interval estimations and axes aligned boxes enclosing the object.

```

Algorithm Enumerate ( ImplObj obj, Cell I )
  evaluate  $f(I)$  with affine arithmetic;
  if ( 0 not in  $obj.f(I)$  )
    return; // Stop if box does not hit the object.
  compute  $ilie$ ;
  * prune  $I$ ;
  * recompute  $ilie$ ;
  if (diameter of  $ilie$  is small enough)
    write results;
    return; // Stop if ILIE tight enough
  subdivide  $I$  into  $m$  sub-boxes,  $I_k$ ,  $k = 1, \dots, m$ ;
  for  $k = 1, \dots, m$ 
    if (0 in  $ilie(I_k)$ ) // Test if  $I_k$  may hit the object
      Enumerate( ImplObj(  $obj.f, I_k$  ));
      // Perform test for new parts

```

4 Experimental Results

The modified enumeration algorithm has been implemented in C++ using the affine arithmetic package of van Iwaarden [29] and the Profile interval package of Knüppel [13]. The affine arithmetic package has been adapted and extended to allow the extraction of “direct dependencies”, the factors of those error symbols corresponding to the input intervals describing the cell \mathbb{I} .

Three surfaces, a stretched sphere (\mathcal{F}_1), an algebraic surface of degree 4 called “Cross Cap” (\mathcal{F}_2), and a Barth

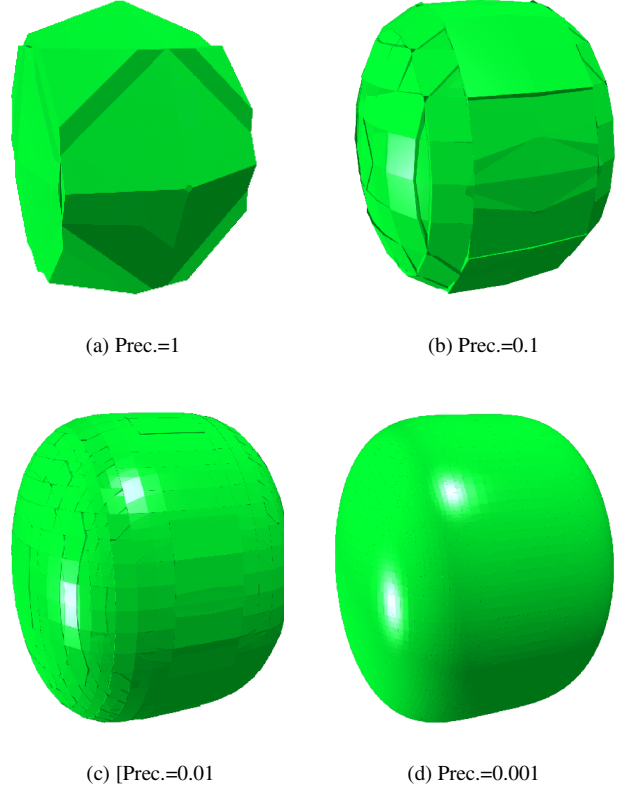
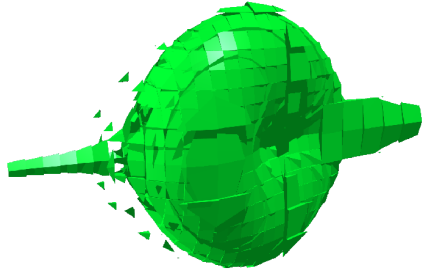


Figure 6: Surface \mathcal{F}_1 enclosed by ILIEs with different precision.

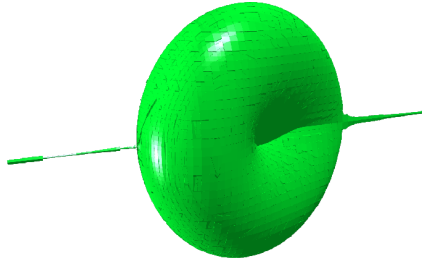
Decic (\mathcal{F}_3) which is algebraic of degree 10, have been chosen to demonstrate the applicability of ILIEs on the enumeration of implicit surface in \mathbb{E}^3 . Although only algebraic surfaces have been chosen for demonstration, the algorithm can be applied on any implicitly defined surface. The formulas of the surfaces $\mathcal{F}_1 - \mathcal{F}_3$ are listed in appendix B. Figures 6 and 7 show results rendered using POVray. Tables 1 – 3 compare the performance of the different algorithms. For each surface the computed precision, subdivision method, number of subdivisions, number of ILIEs/cells representing the result, and computation time are compared. The methods are denoted in the following way: ‘b’ stands for binary subdivision using ILIEs, ‘o’ for octree subdivision with ILIEs and ‘a’ for the classical enumeration algorithm using axes-aligned cells, an incidence test based on affine arithmetic, the diameter of the cells as termination criterion, and uniform adapted octree subdivision. The computation time¹ is noted in seconds and includes a file output of the results in POVray format.

Comparison of axes-aligned enumeration and piecewise enclosures with ILIEs As mentioned before, the computation of ILIEs using affine arithmetic can be done with almost no additional cost compared to an evaluation of the implicit object description with affine

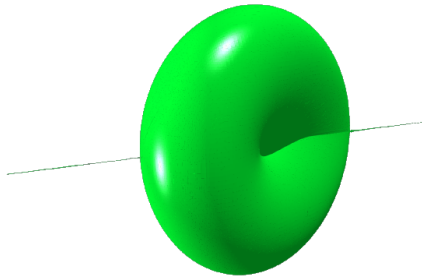
¹PC, 800 MHz AMD Athlon processor



(a) Prec.=0.1



(b) [Prec.=0.01

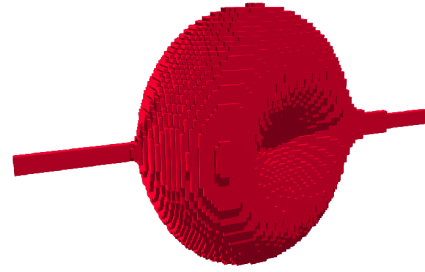


(c) Prec.=0.001

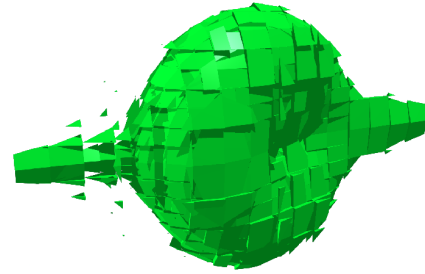
Figure 7: Surface \mathcal{F}_2 enclosed by ILIEs with different precision. Subdivision technique: Octree.

prec.	meth.	#subdiv.	time (s)	#ILIES/cells
1	b	127	0.05	32
	o	73	0.02	32
	a	585	0.08	152
0.1	b	559	0.16	208
	o	521	0.15	248
	a	58569	7.77	21016
0.01	b	2167	0.67	968
	o	2881	1.1	1980
	a	3558793	482.39	1330256
0.001	b	18647	6.49	9176
	o	24449	9.04	18848

Table 1: Results for surface \mathcal{F}_1



(a) Classical Subdivision



(b) Adapted binary subdivision

Figure 8: Cross Cap enclosure of precision 0.1 using axis-aligned cell (a) and adapted binary subdivision with ILIEs (b).

prec.	meth.	#subdiv.	time (s)	#ILIES/cells
0.1	b	2799	1.05	964
	o	3413	1.19	1232
	a	55049	10.20	14956
0.01	b	14635	6.28	5920
	o	18557	7.41	9392
	a	2327561	445.21	840164
0.001	b	95431	42.58	45316
	o	128857	57.23	87768

Table 2: Results for surface \mathcal{F}_2

prec.	meth.	#subdiv.	time (s)
0.1	b	182463	132.40
	o	263457	188.97
0.01	b	1398047	1066.10

Table 3: Results for surface \mathcal{F}_3

arithmetic. The test whether a cell hits the ILIE corresponding to its “mother cell” before the real object-cell incidence test is performed, costs time on one hand, but saves many unnecessary subdivisions and incidence tests on the other hand.

A comparison of numbers in table 2 shows: To get comparable tight enclosure for the Cross Cap using just axes-aligned cells almost 20 times more subdivisions are necessary. The result consists of more than 15 times more elements as if 0.1 is chosen as precision. Just 964 ILIEs are necessary to enclose the surface with the given precision, where 14946 axes-aligned boxes are necessary to get a comparable tight enclosure. Choosing 0.01 as precision, the results are even more convincing: Almost 160 times more subdivisions are required using the classical algorithm and more than 140 times more axes-aligned boxes than ILIEs to represent the results.

This demonstrates not only a remarkable improvement with respect to the performance but also with respect to the quality of the results: The amount of necessary elements to enclose an object is as well a criterion to measure the quality of an enclosure as its reliability and tightness. See figure 8 for a visual comparison of the results.

Comparison of different subdivision techniques

Comparing ILIE based octree subdivision with adapted binary subdivision (see tables 1 – 3) shows that the latter leads especially for high precision to better results with respect to the necessary subdivisions, the number of computed enclosing ILIEs as well as the performance of the algorithm. In case of the Cross Cap (\mathcal{F}_2), binary adapted subdivision requires on the average 24% less subdivisions than octree subdivision. Comparing figure 7 (a) with figure 8 (b) it strikes out that ILIEs computed with octree subdivision fit the surface a bit smoother than the ones computed with binary subdivision.

Reliability of the algorithm Figure 7 (c) shows, that elements not detected by a normal ray tracer do not disappear using ILIEs: the z-axis is part of the surface.

Using iterated cell pruning Iterated cell pruning can be used to improve the result with respect to a further reduction of enclosing ILIEs in the result. An iterative pruning might increase the tightness of the enclosure without decreasing the number of enclosing elements.

5 Conclusions and Future Work

Implicit Linear Interval Estimations have been introduced to improve the classical enumeration algorithm. ILIEs allow to compute piecewise linear, implicit, tight, reliable, and oriented bounds for implicit objects in \mathbf{R}^n . The experimental results presented in section 4 allow the following conclusions:

- The use of ILIEs combined with a simple cell pruning step reduces the amount of necessary subdivisions compared to the classical algorithm dramatically.
- The ILIEs provide a much better enclosure of the object than axes aligned cells in the sense that much less elements are needed to enclose the object.
- The result is much better adapted to the topology of the surface and represents a much better approximation.

The modified enumeration algorithm and the enclosing ILIEs can be used as basis to develop new algorithms for rendering, polygonization, and collision detection. The general definition of ILIEs allows to take other methods to compute ILIEs based on approximation methods like e.g. Taylor Models into consideration.

A Interval and Affine Arithmetics

Interval Arithmetic Intervals can be used to describe fuzzy data, or data that becomes fuzzy during it is processed: The wish to record errors caused by the finite precision of floating point operations gave the initial impulse to use interval arithmetic for numerical calculations. Only a short introduction into some basic ideas of interval arithmetic can be given in this context. The books of Neumaier [15] and Alefeld et al. [2] are recommended for further reading.

Interval arithmetic operates on the set of compact intervals \mathbf{IR} , where a compact interval $I = [a, b] \in \mathbf{IR}$ is defined as

$$[a, b] := \{x \in \mathbf{R} \mid a \leq x \leq b\}$$

For $I = [a, b] \in \mathbf{IR}$, $\inf(I) := a$ denotes the *infimum*, $\sup(I) := b$ the *supremum*, $\text{rad}(I) := (b - a)/2$ the *radius*, $\text{mid}(I) := (a + b)/2$ the *midpoint* and $|I| := \max\{a, b\}$ the *absolute value* of I . An interval is called *thin* if $\inf(I) = \sup(I)$.

For $I = [a, b]$, $J = [c, d]$, the basic arithmetic operations $+$, $-$, \cdot , $/$ are defined as $I + J := [a + c, b + d]$, $I - J := [a - d, b - c]$, $I \cdot J := [\min\{ac, ad, bc, bd\}, \max\{ac, ad, bc, bd\}]$. If $0 \notin J$, division is defined as $I / J := [a, b] \cdot [\frac{1}{d}, \frac{1}{c}]$.

The order relations $\sim \in \{<, \leq, \geq, >\}$ for intervals have the definition $I \sim J \Leftrightarrow x \sim y \quad \forall x \in I, y \in J$.

If *interval arithmetics with directed roundings* is used, the result of a direct interval evaluation $\phi(I)$ of a function $\phi(x)$, $x \in I \in \mathbf{IR}$ is always an enclosure of the range $R_\phi := \{\phi(x) \mid x \in I\}$ of ϕ , that is overestimated in most of the cases and only optimal for some special functions.

For *interval vectors* $x \in \mathbf{IR}^n$ the terms infimum, supremum, midpoint, radius and absolute value, as well as the comparison and inclusion relations are used component wise. Interval vectors are often referred to as *axes aligned boxes* to emphasise the geometric interpretation.

Reliable range analysis is an important application of interval analysis and the overestimations caused by direct interval evaluations is an often criticised drawback. Recent research in the field of reliable arithmetics tries to reduce the effect of overestimation allowing a flexible refinement of the computation or taking more information about occurring errors into account. One of these approaches is *affine arithmetic*, that has been introduced by Stolfi and de Figueiredo will be shortly presented in the next paragraph.

Affine Arithmetic [21] Affine arithmetic reduces the uncontrollable blow up of intervals during the evaluation of arithmetic expressions taking dependencies of uncertainty factors of input values, approximation and rounding errors into account.

Definition [21]: A partially unknown quantity x is represented by an affine form

$$\hat{x} := x_0 + x_1\epsilon_1 + x_2\epsilon_2 + \dots + x_n\epsilon_n$$

in the following shortly denoted by the vector (x_0, \dots, x_n) . The x_i are known real coefficients, the $\epsilon_i \in [-1, 1]$ are symbolic variables, standing for an independent source of error or uncertainty.

x_0 is called the central value of the affine form, the x_i are the partial deviations and the ϵ_i the noise symbols.

Each interval can be expressed as affine form but an affine form can only be approximated by an interval, as it carries much more information. An interval describes only the general uncertainty of the data, whereas affine arithmetic splits this uncertainty into specific parts. Thus, a conversion from affine forms to intervals implies in most cases a loss of information.

Let $\hat{x} := x_0 + x_1\epsilon_1 + x_2\epsilon_2 + \dots + x_n\epsilon_n$ be the affine form of the fuzzy quantity x . x lies in the interval

$$[\hat{x}] := [x_0 - \xi, x_0 + \xi]; \quad \xi := \sum_{i=1}^n |x_i|$$

$[\hat{x}]$ is the smallest interval enclosing all possible values of x .

Let $X = [a, b]$ be an interval representing the value x . Then x can be represented as affine form

$$\hat{x} = x_0 + x_k\epsilon_k$$

with $x_0 := (b + a)/2$; $x_k := (b - a)/2$.

Addition and scalar multiplication are so-called affine operations and follow simple rules applied to their evaluation with affine forms:

Let $\hat{x} = (x_0, \dots, x_n)$ and $\hat{y} = (y_0, \dots, y_n)$ be two affine forms with respect to the same noise symbols $\epsilon_0, \dots, \epsilon_n$ and $\alpha \in \mathbf{R}$. Then

$$\begin{aligned} \hat{x} \pm \hat{y} &:= (x_0 \pm y_0, \dots, x_n \pm y_n) \\ \alpha \hat{x} &:= \alpha (x_0, \dots, x_n) \\ \hat{x} + \alpha &:= (x_0 \pm \alpha, x_1, \dots, x_n) \end{aligned}$$

Non-affine operations are more difficult to determine. Stolfi and de Figueiredo [21] propose the following general strategy for the implementation: Split the operation (if possible) into an affine part and a non-affine part. Calculate the affine part as described in the previous section. For all non-affine parts calculate an affine (best) approximation (e.g. Tchebycheff approximation). The approximation error has to be multiplied with a new noise symbol and has to be added to the affine form to get the affine form of the final result.

A new noise symbol has to be introduced for round-off errors. The upper bounds of all occurring round-off errors have to be added to the partial deviation of the new symbol.

B The tested surfaces

- “Stretched Sphere”

$$\mathcal{F}_1 : x^2 + y^2 + z^4 - 1 = 0$$

- “Cross Cap”

$$\mathcal{F}_2 : 4x^2(x^2 + y^2 + z^2 + z) + y^2(y^2 + z^2 - 1) = 0$$

- “Barth Decic” ($t = \frac{(1+\sqrt{5})}{2}$)

$$\begin{aligned} \mathcal{F}_3 : &8(x^2 - t^4 y^2)(y^2 - t^4 z^2)(z^2 - t^4 x^2) \\ &(x^4 + y^4 + z^4 - 2x^2 y^2 - 2x^2 z^2 - 2y^2 z^2) + \\ &+(3 + 5t)(x^2 + y^2 + z^2 - 1)^2(x^2 + y^2 + z^2 - (2 - t))^2 \\ &= 0 \end{aligned}$$

References

- [1] S. Akkouche and E. Galin. Adaptive implicit surface polygonization using marching triangles. *Computer Graphics Forum*, 20(2):67–80, 2001.
- [2] G. Alefeld and J. Herzberger. *Introduction to Interval Computations*. Academic Press, New York, 1983.
- [3] R. J. Balsys and K. G. Suffern. Visualisation of implicit surfaces. *Computers and Graphics*, 25(1):89–107, 2001.
- [4] J. Bloomenthal. Polygonisation of implicit surfaces. *Computer Aided Geometric Design*, 5:341–355, 1988.
- [5] A. Bowyer. *SvLis: Introduction and User Manual*. Information Geometers, Winchester, UK, 1994.
- [6] A. Bowyer, J. Berchtold, D. Eisenthal, I. Voiculescu, and K. Wise. Interval methods in geometric modeling. In R. Martin and W. Wang, editors, *Proceedings of the Conference on Geometric Modeling and Processing (GMP-00)*, pages 321–327, Los Alamitos, April 10–12 2000. IEEE.

- [7] A. Bowyer, R. Martin, H. Shou, and I. Voiculescu. Affine intervals in a csg geometric modeller. In *Proceedings of the Workshop on Uncertainty in Geometric Computations*. Sheffield University, UK, July 2001. to be published in 2002.
- [8] K. Bühler. Fast and reliable plotting of implicit curves. In *Proceedings of the Workshop on Uncertainty in Geometric Computations*. Sheffield University, UK, July 2001. to be published in 2002.
- [9] K. Bühler. Linear interval estimations for parametric objects. *Computer Graphics Forum*, 20(3), 2001.
- [10] L. H. De Figueiredo and J. Gomes. Sampling implicit objects with physically-based particle systems. *Computers and Graphics*, 20(3):365–375, 1996.
- [11] L. H. de Figueiredo and J. Stolfi. Adaptive enumeration of implicit surfaces with affine arithmetic. *Computer Graphics Forum*, 15(5):287–296, 1996.
- [12] T. Duff. Interval arithmetic and recursive subdivision for implicit functions and constructive solid geometry. *Computer Graphics*, 26(2):131–138, 1992.
- [13] W. Knüppel. Profil - programmer's runtime optimized fast interval library. Technical Report 93.4, TU Hamburg Harburg, Technische Informatik II, 1993.
- [14] L. P. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel. Feature-sensitive surface extraction from volume data. In E. Fiume, editor, *SIGGRAPH 2001, Computer Graphics Proceedings, Annual Conference Series*, pages 57–66. ACM Press, 2001.
- [15] A. Neumaier. *Interval Methods for Systems of Equations*, volume 37 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 1990.
- [16] R.R. Martin Q. Zhang. Polynomial evaluation using affine arithmetic for curve drawing. In *Eurographics UK 2000 Conference Proceedings*, pages 49–56, Abingdon, 2000. Eurographics UK.
- [17] A. Rösch, M. Ruhl, and D. Saupe. Interactive visualization of implicit surfaces with singularities. *Computer Graphics Forum*, 16(5):295–306, 1997.
- [18] A. Sherstyuk. Fast ray tracing of implicit surfaces. *Computer Graphics Forum*, 18(2):139–148, 1999.
- [19] J. M. Snyder. Interval analysis for computer graphics. *Computer Graphics*, 26(2):121–130, 1992.
- [20] J. M. Snyder, A. R. Woodbury, K. Fleischer, B. Currin, and A. H. Barr. Interval method for multi-point collision between time-dependent curved surfaces. In J. T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 321–334, 1993.
- [21] J. Stolfi and L. H. de Figueiredo. Self-validated numerical methods and applications, 1997. Course Notes for the 21th Brazilian Mathematics Colloquium held at IMAP, July 1997.
- [22] N. Stolte and A. Kaufman. Parallel spatial enumeration of implicit surfaces using interval arithmetic for octree generation and its direct visualization. In *Implicit Surfaces '98*, pages 81–88, 1998.
- [23] N. Stolte and A. Kaufman. Robust hierarchical voxel models for representation and interactive visualization for implicit surfaces in spherical coordinates. In *Implicit Surfaces '98*, pages 81–88, 1998.
- [24] K. G. Suffern and E. D. Fackerell. Interval methods in computer graphics. *Computers and Graphics*, 15(3):331–340, 1991.
- [25] S. Tanaka, A. Morisaki, S. Nakata, Y. Fukuda, and H. Yamamoto. Sampling implicit surfaces based on stochastic differential equations with converging constraint. *Computers and Graphics*, 24(3):419–431, 2000.
- [26] G. Taubin. An accurate algorithm for rasterizing algebraic curves. *IEEE Computer Graphics and Applications*, 14(2):14–23, 1994.
- [27] F. Triquet, P. Meseure, and C. Chaillou. Fast polygonization of implicit surfaces. In V. Skala, editor, *WSCG 2001 Conference Proceedings*, 2001.
- [28] G. van den Bergen. Efficient collision detection of complex deformable models using AABB trees. *Journal of Graphics Tools: JGT*, 2(4):1–14, 1997.
- [29] R. van Iwaarden and J. Stolfi. Affine arithmetic software. 1997.
- [30] L. Velho, L. H. de Figueiredo, and J. Gomes. A methodology for piecewise linear approximation of surfaces. *Journal of the Brazilian Computer Society*, 3(3):30–42, 1997.
- [31] I. Voiculescu, J. Berchtold, A. Bowyer, R. R. Martin, and Q. Zhang. Interval and affine arithmetic for surface location of power- and Bernstein-Form polynomials. In R. Cipolla and R. Martin, editors, *The Mathematics of Surfaces*, volume IX, pages 410–423. Springer, 2000.
- [32] B. Wyvill, A. Guy, and E. Galin. Extending the CSG tree. warping, blending and Boolean operations in an implicit surface modeling system. *Computer Graphics Forum*, 18(2):149–158, June 1999.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.