*Volume xxx, (1996) number yyy pp. 1–10*

# Adaptive Enumeration of Implicit Surfaces with Affine Arithmetic

Luiz Henrique de Figueiredo[†] and Jorge Stolfi[‡]

[†] TeC$_{Graf}$ : Computer Graphics Technology Group, Computer Science Department, PUC–Rio
Rua Marquês de São Vicente 225, 22453–900 Rio de Janeiro, RJ, Brazil (lhf@icad.puc-rio.br)
[‡] Computer Science Department (DCC/IMECC), Universidade Estadual de Campinas (UNICAMP)
Caixa Postal 6065, 13081–970 Campinas, SP, Brazil (stolfi@dcc.unicamp.br)

**Abstract**

*We discuss adaptive enumeration and rendering methods for implicit surfaces, using octrees computed with affine arithmetic, a new tool for range analysis. Affine arithmetic is similar to standard interval arithmetic, but takes into account correlations between operands and sub-formulas, generally providing much tighter bounds for the computed quantities. The resulting octrees are accordingly much smaller, and the rendering faster.*

**Keywords:** Cellular models; Interval arithmetic; Affine Arithmetic; Rendering; Implicit surfaces.

## 1. Introduction

Implicit surfaces have recently become popular in computer graphics and solid modeling. In order to exploit existing hardware and algorithms, it is often necessary to approximate such surfaces by models with simpler geometry, such as polygonal meshes or voxel arrays. Therefore, it is important to find efficient approximation methods for implicit surfaces.

Let $S$ be a surface defined implicitly by the equation $h(x, y, z) = 0$. A simple and general technique for computing an approximation of $S$ in a region $\Omega$ is:

1. decompose $\Omega$ into small cells;
2. identify which cells intersect $S$;
3. approximate $S$ within each intersecting cell.

The enumeration of the intersecting cells is usually the most expensive step in this method. In the simplest schema, the cells that intersect $S$ are identified by sampling. The function $h$ is evaluated at the vertices of each cell; if the signs of those values are not identical, then the cell necessarily intersects $S$ (assuming that $h$ is continuous). Obviously, the converse does not hold: when all the values of $h$ at the vertices have the same sign, we cannot conclude that the cell does not intersect $S$. This is a form of aliasing in the sampling related to the size of the cell. Therefore, cell sizes must be carefully chosen to avoid missing features due to undersampling.

Typically, the cellular decomposition of $\Omega$ is a regular grid of cubes. If there are $n$ cubes along each main direction, then there are $n^3$ cubes to be scanned in a full enumeration, but only $O(n^2)$ cubes will intersect $S$. Thus, choosing a smaller cell size to avoid aliasing in the sampling will greatly increase the number of cells to be scanned, and also increase the fraction of "useless" tests. Therefore, this approximation method is simple but not efficient.

Figure 1 shows a full enumeration of the curve defined implicitly by $y^2 - x^3 + x = 0$, in the square $\Omega = [-2 \,..\, 2] \times [-2 \,..\, 2]$, using a $16 \times 16$ grid. Intersecting cells were identified by sampling and appear in grey (note how few they are: only 44 out of 256). Black dots mark the points where the curve crosses cell edges; these points can be joined to form a polygonal approximation to the curve.

In Section 2, we review some general methods that try to find the cells that intersect $S$ without enumerating all cells in $\Omega$. The most reliable of those seems to be recursive subdivision of space based on *range analysis*, i.e., on estimates for the range of values taken by

© The Eurographics Association 1996. Published by Blackwell Publishers, 108 Cowley Road, Oxford OX4 1JF, UK and 238 Main Street, Cambridge, MA 02142, USA.

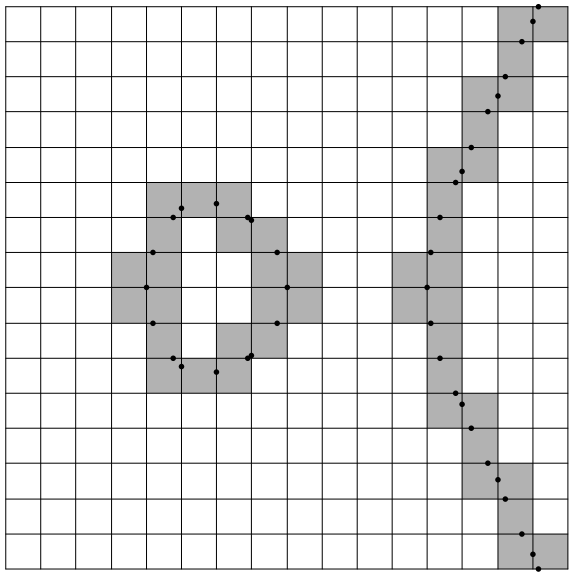

**Figure 1:** *Full enumeration of the curve given implicitly by* $y^2 - x^3 + x = 0$ *in the square* $[-2 \,..\, 2] \times [-2 \,..\, 2]$.

$h$ on subsets of $\Omega$. Interval arithmetic is the natural technique for range analysis[1]. However, as we point out in Section 3, the excessive conservatism of interval arithmetic may greatly reduce the efficiency of the search. In Section 4, we describe *affine arithmetic*, a computation technique that generally provides much tighter bounds than interval arithmetic. In Section 5, we describe some experiments that exploit the properties of affine arithmetic to efficiently enumerate implicit surfaces. In Section 6, we show how to render implicit surfaces directly, during enumeration, without explicitly creating a cellular model. Finally, Section 7 contains some conclusions and outlines directions for future work.

## 2. Efficient enumeration methods

Several methods exist for finding the cells intersecting a surface $S$ without visiting all cells in a cellular decomposition of a region $\Omega$. The goal is to avoid testing all $O(n^3)$ cells, since only $O(n^2)$ cells intersect $S$.

The most popular efficient enumeration methods are still based on sampling. However, hierarchical decomposition methods are more efficient and more robust. In this section, we review some of these methods.

### 2.1. Sampling-based methods

*Continuation methods*[2–5] sample the surface only in the immediate neighbourhood of known intersecting cells. These methods only work for cellular subdivisions of $\Omega$ whose topology and geometry are well understood[5], so that it is simple to advance from one intersecting cell to an adjacent one, using "pivoting" rules like reflection[4].

*Adaptive refinement techniques*[2, 6–8] start with a coarse cellular subdivision, and further subdivide only those cells that do intersect $S$. Such techniques frequently use estimates of the curvature of $S$ in a cell as an additional subdivision criterion, thus building a cellular subdivision that is adapted not only to the position of $S$ inside $\Omega$, but also to the intrinsic geometry of $S$. Polygonal approximations based on such subdivisions are thus quite efficient.

All these methods based on sampling have one fundamental difficulty, besides aliasing: finding a set of initial "seed" cells intersecting each connected component of $S$ in $\Omega$. (Unlike parametric surfaces, implicit surfaces can have several connected components; see Figure 1.) Obviously, if $S$ is a general implicit surface, and we are restricted to evaluating $h$ at individual points, then this problem has no satisfactory solution. Even if we assume that $S$ has no extended features thinner than the minimum cell size (so that there are no aliasing problems), we still must sample the whole region $\Omega$ uniformly at that resolution; otherwise, we may miss some component of $S$.

### 2.2. Hierarchical decomposition methods

In order to discard large portions of $\Omega$ quickly and reliably, we need a more powerful test than point sampling. The latter can only prove the *presence* of $S$ in some region of $\Omega$; to reduce the number of cells scanned, we need a test procedure that can also prove the *absence* of $S$ in a region.

The *hierarchical decomposition methods* rely on such a test to explore $\Omega$ recursively, starting with $\Omega$ itself as the initial cell. If a cell is proved to be empty, it is ignored; otherwise, it is subdivided into smaller cells, which are then explored recursively, until the cells are small enough to approximate $S$.[9–13]

The meaning of "small enough" depends on the application. For rendering, it might mean "smaller than a pixel". For other applications, such as modeling, it may depend on some other numerical criterion. For instance, testing how closely the surface can be approximated by a linear function inside the cell allows polygonal approximations to adapt to the curvature of the surface.

Note that the test procedure is not required to be complete, in the sense that it may fail to prove either the presence or the absence of $S$ in a given cell.

© The Eurographics Association 1996

In particular, a cell that is declared "small enough" may still have unknown status. Each application must decide what to do with those "indeterminate" cells: discard them, treat them just like the cells that do intersect $S$, or handle them in some special way. Point sampling may be useful at this stage to help identify some intersecting cells.

The subdivision of $\Omega$ resulting from a hierarchical decomposition is not a regular grid, but rather a tree of nested cells. The leaves in this tree form a cellular model for $S$. There are many variants of this method; they differ in the shape of cells and the method of subdivision. In particular, if the cells are cubical and divided into eight equal parts, then the resulting subdivision is called an *octree*[14]. Other choices give *binary space partitions*[15], 3-*d trees*[16], *hierarchical triangulations*[17], and many more.

## 3. Interval arithmetic

In order to prove that a given cell $C$ does not intersect $S$, we must prove that the function $h$ does not vanish inside $C$. For some classes of surfaces, this fact can usually be established by specific tests. For instance, if $h$ is a low-degree polynomial, then we can compute its Bézier-Bernstein coefficients in that cell, by writing $h$ as a linear combination of Bernstein polynomials. If all the coefficients are positive, then $h$ is positive in the cell, because Bernstein polynomials are always positive and hence the value of $h$ at any point in the cell can be expressed as a convex combination of those coefficients. Similarly, if all the coefficients are negative, then so is $h$ in the cell. Unfortunately, this method cannot be used for general (non-polynomial) functions; or even for polynomials of high degree that are not given in monomial form, but rather in semi-factored form, such as $(x^2 + y^3)^1 5 - (x^4 - z^2)^1 2$, due to the large number of coefficients that would have to be computed.

Range analysis can be used to provide general test procedures for hierarchical decompositions. In range analysis, an estimate is computed for the whole range of values taken by the function $h$ on the points of a cell $C$. This estimate is an interval *guaranteed* to contain $h(C)$. If this interval does not contain zero, then the function $h$ cannot vanish inside $C$ and the test has proved that the cell does not intersect $S$. However, if the interval does contains zero, we cannot conclude that the cell intersects $S$ because estimates are not required to be exact.

The classical technique of *interval arithmetic* (IA), also known as *interval analysis*, provides a natural tool for range analysis[1]. In IA, each quantity is represented by an interval of floating-point numbers. Those intervals are added, subtracted, multiplied, etc., in such a way that each computed interval is guaranteed to contain the (unknown) value of the quantity it represents[18].

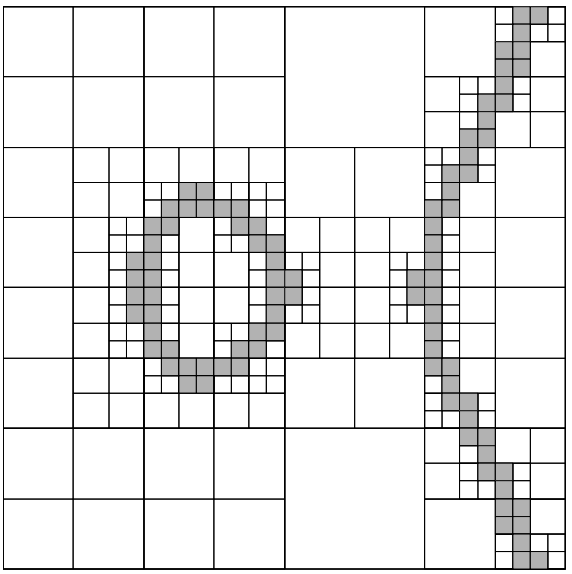

**Figure 2:** *Hierarchical enumeration of the cubic* $y^2 - x^3 + x = 0$ *in the square* $[-2 \,..\, 2] \times [-2 \,..\, 2]$ *using interval arithmetic.*

To test whether a cell $C$ intersects the surface $S$, we can evaluate $h(x, y, z)$ with IA, letting $x$, $y$ and $z$ be the projections of the cell onto the coordinate axes. The interval thus computed is guaranteed to contain all values of $h$ for points inside the cell. If this interval is entirely positive, or entirely negative, then we have proved that $C$ does not intersect $S$.

Hierarchical decomposition methods based on IA have recently been proposed for the enumeration of implicit surfaces in computer graphics applications[10–12]. Those methods have become quite popular, due to their ability to handle arbitrarily complex non-polynomial surfaces, and their immunity to round-off errors.

Figure 2 shows a hierarchical enumeration based on IA of the curve shown in Figure 1, but now using a $32 \times 32$ grid. Note that large portions of $\Omega$ were discarded at early stages. Section 5 contains other examples.

### 3.1. The error explosion problem

The main weakness of IA is that it tends to be too conservative: the computed interval for a quantity may be much wider than the exact range of that quantity, often to the point of uselessness. This over-conservatism

© The Eurographics Association 1996

is mainly due to the assumption that the (unknown) values of the arguments to primitive operations may vary *independently* over the given interval If there are any mathematical constraints between these arguments, then not all combinations of values in the corresponding intervals will be valid. In that case, the result interval computed by IA may be much wider than the exact range of the result quantity.

For example, consider evaluating $x(10 - x)$, where $x$ is known to lie in the interval $\bar{x} = [4 .. 6]$. Applying the IA formulas blindly, we get

$$\begin{aligned} 10 - \bar{x} &= [10 .. 10] - [4 .. 6] = [4 .. 6] \\ \bar{x}(10 - \bar{x}) &= [4 .. 6] \cdot [4 .. 6] = [16 .. 36], \end{aligned}$$

which is 20 times wider than the exact range of the expression $x(10 - x)$ over $[4 .. 6]$, namely $[24 .. 25]$. The large discrepancy between the two intervals is due to the inverse relation between the quantities $x$ and $10 - x$, which is not known to the IA multiplication algorithm.

The over-conservatism of IA is particularly bad in long computation chains, where the intervals computed by one stage of the chain are the inputs to the following stage. In such cases, one often observes an "error explosion": as the evaluation advances down the chain, the relative accuracy of the computed intervals decreases exponentially, and they soon become too wide to be useful, by many orders of magnitude. Unfortunately, long computations chains are not uncommon in computer graphics applications.

## 4. Affine arithmetic

To address the "error explosion" problem in IA, Comba and Stolfi[19] proposed a new model for numerical computation, called *affine arithmetic* (AA). Like standard IA, affine arithmetic keeps track automatically of the round-off and truncation errors affecting each computed quantity. Unlike IA, however, affine arithmetic keeps track of *correlations* between those quantities. Thanks to this extra information, AA is able to provide much tighter intervals than IA, especially in long computation chains.

The key feature of AA is an extended encoding of quantities from which one can determine, in addition to their ranges, also certain relationships to other quantities — such as the ones existing between $x$ and $10-x$ in the example. Specifically, a partially unknown quantity $x$ is represented in AA by an *affine form* $\hat{x}$, which is a first-degree polynomial:

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

Here, the $x_i$ are known real coefficients (stored as floating-point numbers), and the $\varepsilon_i$ are symbolic variables whose values are unknown but assumed to lie in the interval $\mathbf{U} = [-1 .. +1]$.

Each $\varepsilon_i$ is called a *noise symbol*: it stands for an independent source of error or uncertainty that contributes to the total uncertainty of the quantity $x$; the corresponding coefficient $x_i$ gives the magnitude of that contribution. The source of error may be external (due to original uncertainty in some input quantity) or internal (due to round-off and truncation errors committed in the computation of $\hat{x}$).

A somewhat similar approach has been proposed by Hansen[20], in which quantities are represented instead by affine combinations of a *fixed* number of *intervals*. In AA, new noise symbols are dynamically created during a long computation (Section 4.3).

As one may expect, affine arithmetic is more complex and expensive than ordinary interval arithmetic. However, its higher accuracy is worth the extra cost in many applications, including adaptive enumeration of implicit objects, as we show in Section 5.

### 4.1. Conversions between IA and AA

If $\hat{x} = x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n$ is an affine form for a quantity $x$, then the value of $x$ is guaranteed to be in the *range* of $\hat{x}$, which is the interval

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

Note that $[\hat{x}]$ is the smallest interval that contains all possible values of $\hat{x}$, assuming that each $\varepsilon_i$ ranges independently over the interval $\mathbf{U} = [-1 .. +1]$.

Conversely, given an interval $\bar{x} = [a .. b]$ representing some quantity $x$ in IA, an equivalent affine form for the same quantity is given by $\hat{x} = x_0 + x_k\varepsilon_k$, where

$$x_0 = \frac{b+a}{2} \quad \text{and} \quad x_k = \frac{b-a}{2} .$$

The noise symbol $\varepsilon_k$ symbolizes the uncertainty in the value of $x$. Since the interval $\bar{x}$ tells us nothing about possible constraints between the value of $x$ and that of other variables, $\varepsilon_k$ must be distinct from all other noise symbols previously used in the same computation.

### 4.2. Exploiting correlations in operands

The key feature of AA is that the same noise symbol $\varepsilon_i$ may contribute to the uncertainty of two or more quantities (inputs, outputs, or intermediate results) arising in the evaluation of an expression. The sharing of a noise symbol $\varepsilon_i$ by two affine forms $\hat{x}$, $\hat{y}$ indicates some partial dependency between the underlying quantities $x$, $y$. The magnitude and sign of

© The Eurographics Association 1996

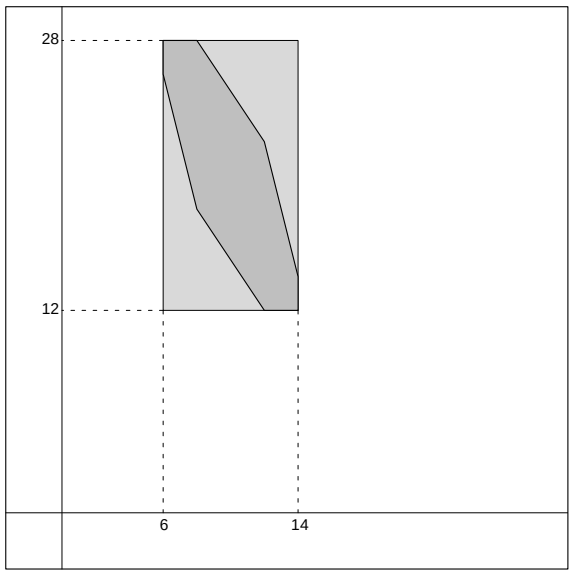


**Figure 3:** *Joint range of two quantities $x$ and $y$, where $\hat{x} = 10 + 2\varepsilon_1 + 1\varepsilon_2 - 1\varepsilon_4$ and $\hat{y} = 20 - 3\varepsilon_1 + 1\varepsilon_3 + 4\varepsilon_4$. Because $x$ and $y$ are not entirely independent of each other, the pair $(x, y)$ lies in a region that is substantially smaller than the rectangle* $[6\,..\,14] \times [12\,..\,28]$.

the dependency is determined by the corresponding coefficients $x_i, y_i$.

For example, suppose that the quantities $x$, $y$ are represented by the affine forms

$$\begin{aligned} \hat{x} &= 10 + 2\varepsilon_1 + 1\varepsilon_2 \qquad - 1\varepsilon_4 \\ \hat{y} &= 20 - 3\varepsilon_1 \qquad + 1\varepsilon_3 + 4\varepsilon_4. \end{aligned}$$

From this data, we can tell that $x$ lies in the interval $[6\,..\,14]$ and $y$ lies in $[12\,..\,28]$. However, since they both include the same noise variables $\varepsilon_1$ and $\varepsilon_4$ with non-zero coefficients, they are not entirely independent of each other. In fact, the pair $(x, y)$ is constrained to lie in the region of $\mathbf{R}^2$ depicted in Figure 3 (dark grey), which is substantially smaller than the rectangle $[6\,..\,14] \times [12\,..\,28]$ (light grey). Obviously, this dependency information would be lost if we were to replace $\hat{x}$ and $\hat{y}$ by the ordinary intervals $[\hat{x}]$ and $[\hat{y}]$, even though the latter encode precisely the same ranges of values as the former.

### 4.3. Computing with affine arithmetic

To evaluate a formula in AA, we must replace each of its elementary operations $z \leftarrow f(x, y)$ on real numbers by an equivalent operation $\hat{z} \leftarrow \hat{f}(\hat{x}, \hat{y})$ on affine forms, where $\hat{f}$ is a procedure that computes an affine form for $z = f(x, y)$ that is consistent with $\hat{x}, \hat{y}$.

When $f$ is an affine function of $x, y$, the value $\hat{z}$ can be expressed exactly as an affine combination of the noise symbols $\varepsilon_i$. More precisely, if

$$\begin{aligned} \hat{x} &= x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n \\ \hat{y} &= y_0 + y_1\varepsilon_1 + \cdots + y_n\varepsilon_n, \end{aligned}$$

and $\alpha \in \mathbf{R}$, then

$$\begin{aligned} \hat{x} \pm \hat{y} &= (x_0 \pm y_0) + (x_1 \pm y_1)\varepsilon_1 + \cdots + (x_n \pm y_n)\varepsilon_n \\ \alpha\hat{x} &= (\alpha x_0) + (\alpha x_1)\varepsilon_1 + \cdots + (\alpha x_n)\varepsilon_n \\ \hat{x} \pm \alpha &= (x_0 \pm \alpha) + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n. \end{aligned}$$

Note that, according to those formulas, the difference $\hat{x} - \hat{x}$ between an affine form and itself is identically zero. In this case, the fact that the two operands share the same noise symbols with the same coefficients reveals that they are actually the same quantity, and not just two quantities that happen to have the same range of possible values. Thanks to this feature, in AA we also have $(\hat{x} + \hat{y}) - \hat{x} = \hat{y}$, $(3\hat{x}) - \hat{x} = 2\hat{x}$, and so on. Such properties are *not* valid in IA, and are one source of error explosion.

When $f$ is not an affine operation, $\hat{z}$ cannot be expressed exactly as an affine combination of the $\varepsilon_i$. In that case, we pick the best affine approximation (best in the Chebyshev sense of minimizing the maximum error), and then append an extra term $z_k\varepsilon_k$ to represent the error introduced by this approximation:

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

Here, $\varepsilon_k$ must be a brand new noise symbol (distinct from all other noise symbols in the same computation) and $z_k$ must be an upper bound for the approximation error. Note that, unlike Hansen's generalized interval arithmetic, new noise symbols are created during a long AA computation.

Using this approach, the multiplication of two affine forms $\hat{x}$, $\hat{y}$ is given by

$$\begin{aligned} z_0 &= x_0 y_0 \\ z_i &= x_0 y_i + y_0 x_i \qquad (i = 1, \ldots, n) \\ z_k &= uv, \end{aligned}$$

where

$$u = \sum_{i=1}^{n} |x_i|, \qquad v = \sum_{i=1}^{n} |y_i|.$$

Similar formulas can be given for the other elementary operations and functions. For instance, it turns out[19] that the square root of an affine form $\hat{x} = x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n$ is given by

$$\begin{aligned} z_0 &= \alpha x_0 + \beta \\ z_i &= \alpha x_i \qquad (i = 1, \ldots, n) \\ z_k &= \delta, \end{aligned}$$

© The Eurographics Association 1996

where

$$\alpha = \frac{1}{\sqrt{a}+\sqrt{b}}$$

$$\beta = \frac{\sqrt{a}+\sqrt{b}}{8} + \frac{1}{2}\frac{\sqrt{a}\sqrt{b}}{\sqrt{a}+\sqrt{b}}$$

$$\delta = \frac{1}{8}\frac{(\sqrt{b}-\sqrt{a})^2}{\sqrt{a}+\sqrt{b}}$$

and $[a\,..\,b]$ is the interval $[\hat{x}]$.

### 4.4. Example revisited

Consider again evaluating $z = x(10-x)$, for $x$ in the interval $[4\,..\,6]$, but now using AA instead of IA:

$$\begin{aligned} \hat{x} &= 5 + 1\varepsilon_1 \\ 10-\hat{x} &= 5 - 1\varepsilon_1 \\ \hat{z} = \hat{x}(10-\hat{x}) &= 25 + 0\varepsilon_1 - 1\varepsilon_2. \end{aligned}$$

Observe that the influence of the noise symbol $\varepsilon_1$ in the factors happened to cancel out (to first order) in the product.

The range of $z$ implied by the affine form $\hat{z}$ is

$$[\hat{z}] = [25-1\,..\,25+1] = [24\,..\,26],$$

which is much closer to $[24\,..\,25]$, the exact range of $z$. Recall that IA gave $[16\,..\,36]$ for this expression.

## 5. Examples of adaptive enumeration

In this section, we show examples that compare the performance of IA and AA in adaptive enumeration. Because pictures of three dimensional decompositions are not easy to understand, we only give pictures of the enumeration of two-dimensional implicit curves.

In all these examples, the goal is to build a cellular model of a curve given implicitly by $h(x,y) = 0$ inside some rectangle $\Omega$. The model consists of a set of cells from some fixed regular grid, whose union is guaranteed to contain all zeros of $h$ in $\Omega$. These cells are shown in grey in the pictures.

### 5.1. A quartic

Take the quartic curve defined by

$$h(x,y) = x^2 + y^2 + xy - (xy)^2/2 - 1/4$$

in the square $\Omega = [-2\,..\,2] \times [-2\,..\,2]$, using a $32 \times 32$ grid of cells. In a full enumeration, $1024 = 32 \cdot 32$ cells have to be scanned, but the curve actually enters only 66 of these cells.

Figure 4 illustrates an adaptive enumeration with interval arithmetic and affine arithmetic, using a 2-d tree. With IA, the range of $h$ was evaluated 847 times and 246 cells remained in the model (i.e., could not be shown to be disjoint from the curve). With AA, the range of $h$ was evaluated 451 times and only 70 cells remained in the model. Thus, IA generated a model with 180 useless cells whereas AA generated a model with only 4 useless cells, even at such a relatively low resolution.

### 5.2. Quadratic electrostatic potential

Consider now the "quadratic" electrostatic potential defined by $n$ charges $q_i$ located at points $(x_i, y_i)$:

$$h(x,y) = \sum_{i=1}^{n} \frac{q_i}{(x-x_i)^2 + (y-y_i)^2} \ .$$

(True electrostatic potential depends on distances, not on square of distances.) In this case, the interesting curves are the equipotential curves $h(x,y) = h_0$. Such "potential" functions and their variations are useful for modeling with implicit objects[21].

Figure 5 illustrates an adaptive quadtree enumeration with interval arithmetic and affine arithmetic of an equipotential curve for the quadratic electrostatic potential defined by four equal charges in the square $\Omega = [1/2\,..\,3/2] \times [1/2\,..\,3/2]$. The grid is $64 \times 64$; a full enumeration would have to scan 4096 cells. In both cases, the range of $h$ was evaluated 1449 times (a coincidence). However, 346 cells remained in the model generated by IA, whereas only 299 remained in the model generated by AA.

### 5.3. The quartic in 3d

As a three-dimensional example, consider an octree enumeration of the quartic surface defined by

$$h(x,y,z) = x^2 + y^2 + xy - (xy)^2/2 - 1/4 - z,$$

in the cube $\Omega = [-2\,..\,2] \times [-2\,..\,2] \times [-2\,..\,2]$.

We give two tables showing how the size of the model (number of leaves in the octree) and the time (in seconds) taken to compute it depend on the resolution of the underlying grid (the resolution is $n = 2^L$ at level $L$). Table 1 shows this data for IA; Table 2 shows this data for AA. Times were measured in an otherwise idle SPARCstation 10.

Reading these tables, we observe that AA evaluates $h$ half as many times as IA, and generates a model half the size of the model generated by IA: the size of the IA model is approximately $5n^2$, whereas the size of the AA model is approximately $2.4n^2$. However, AA is 4.3 times slower than IA: the time for the IA model is approximately $86n^2\mu$sec whereas the time for the AA model is approximately $370n^2\mu$sec (note that the time is linear in the size of the model.)

© The Eurographics Association 1996

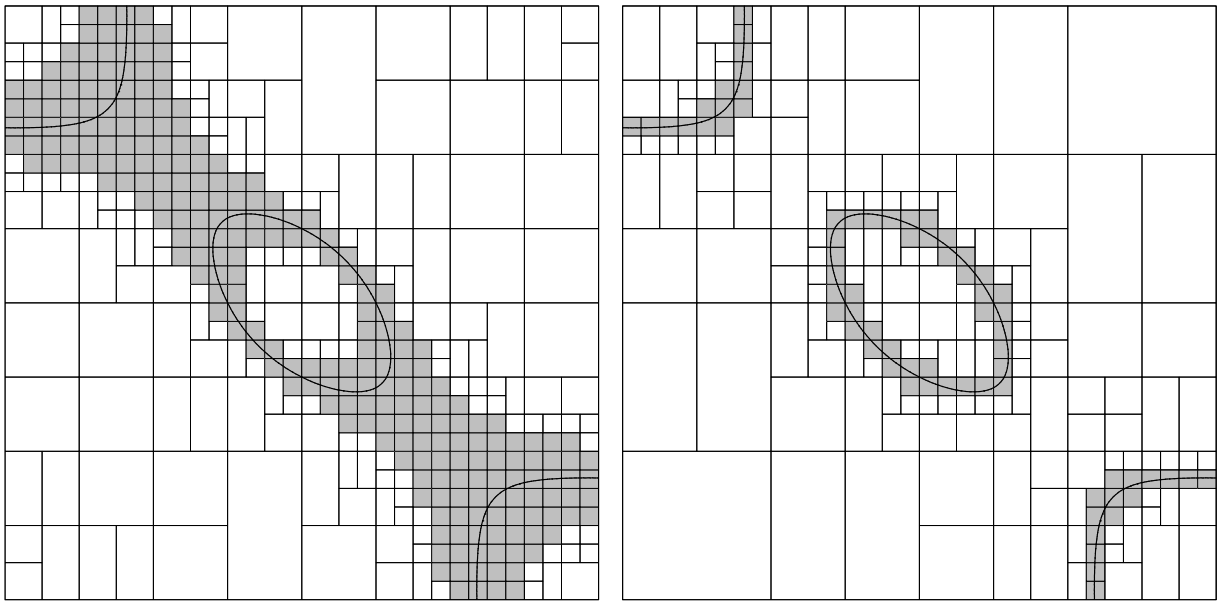

**Figure 4:** *Adaptive enumeration of quartic with IA (left) and AA (right).*

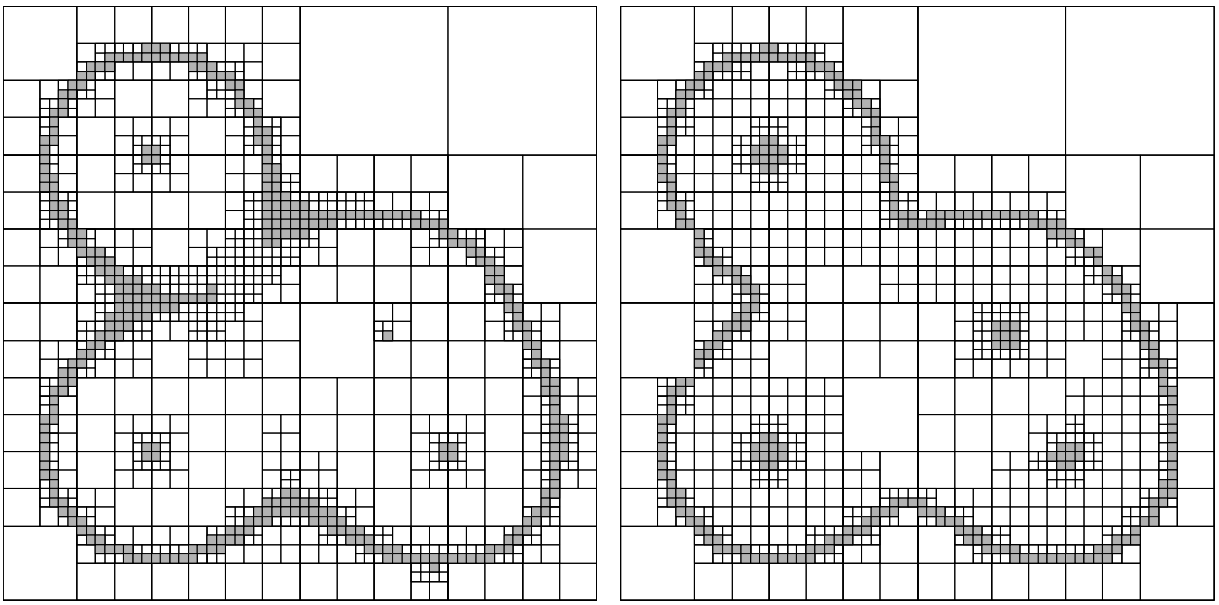

**Figure 5:** *Adaptive enumeration of quadratic electrostatic potential with IA (left) and AA (right).*

© The Eurographics Association 1996

| level | evaluations | leaves | time |
|---|---|---|---|
| 1 | 9 | 8 | 0.0 |
| 2 | 73 | 62 | 0.0 |
| 3 | 569 | 334 | 0.0 |
| 4 | 3241 | 1276 | 0.1 |
| 5 | 13449 | 5122 | 0.2 |
| 6 | 54425 | 20580 | 0.4 |
| 7 | 219065 | 82502 | 1.5 |
| 8 | 879081 | 329746 | 5.7 |
| 9 | 3517049 | 1318058 | 22.5 |
| 10 | 14061513 | 5275400 | 89.5 |

**Table 1:** *Performance of adaptive enumeration of 3d quartic with IA.*

| level | evaluations | leaves | time |
|---|---|---|---|
| 1 | 9 | 8 | 0.0 |
| 2 | 73 | 56 | 0.0 |
| 3 | 521 | 190 | 0.1 |
| 4 | 2041 | 710 | 0.2 |
| 5 | 7721 | 2664 | 0.5 |
| 6 | 29033 | 10104 | 1.7 |
| 7 | 109865 | 39960 | 6.4 |
| 8 | 429545 | 158282 | 24.8 |
| 9 | 1695801 | 630380 | 97.8 |
| 10 | 6738841 | 2516356 | 388.5 |

**Table 2:** *Performance of adaptive enumeration of 3d quartic with AA.*

## 6. Rendering

The octrees built during enumeration can be used to speed up ray tracing algorithms[22, 23]. However, a "lower" quality rendering can be done directly, concurrently with the enumeration; there is no need to explicitly build an octree.

By composing the function $h$ with the appropriate projective map, we can take the domain $\Omega$ to be a rectangular box, aligned with the coordinate axes, and compute the image of the orthogonal projection of the surface $S$ defined by $h$ onto a horizontal plane located above the box $\Omega$. For simplicity, we assume that the resolution of the image is $n \times n$, where $n = 2^L$. The enumeration proceeds down the octree to level $L$. When we reach a leaf of the octree (i.e., a cell at level $L$), we compute the colour of the corresponding pixel in the image. Each cell has an integer address $(x, y, z)$; all cells with the same $x$ and $y$ project onto the same pixel $(x, y)$. To remove hidden parts, we use a "painter's algorithm", carefully traversing the octree from bottom to top (in the $z$ direction).

In order to compute the colour of a pixel in shaded images of $S$, we need to compute also the mean normal direction to $S$ in the cell corresponding to the pixel. The normal direction is the direction of the gradient of $h$. Full enumeration and adaptive enumeration with interval arithmetic do not directly provide an estimate for $\nabla h$. For full enumeration, one typically uses an estimate based on central differences[24]. For adaptive enumeration with interval arithmetic, techniques such as formal differentiation (evaluated in either floating point or IA) or Baur–Strassen differentiation[25] would be required to estimate $\nabla h$ in each visible cell. On the other hand, affine arithmetic directly provides an estimate for $\nabla h$. As explained in Section 4.3, to estimate $h$ in the cell represented by the intervals $\bar{x}$, $\bar{y}$, $\bar{z}$, we take the corresponding affine forms $\hat{x} = x_0 + x_1\varepsilon_1$, $\hat{y} = y_0 + y_2\varepsilon_2$, $\hat{z} = z_0 + z_3\varepsilon_3$, and compute an affine form for $h$:

$$\hat{h} = h(\hat{x}, \hat{y}, \hat{z}) = h_0 + h_1\varepsilon_1 + h_2\varepsilon_2 + h_3\varepsilon_3 + \ldots$$

combining the AA formulas for the primitive operations that appear in $h$. Of interest in this expression are the coefficients $h_1$, $h_2$, $h_3$ that correspond to the "input" noise symbols $\varepsilon_1, \varepsilon_2, \varepsilon_3$.

Note that, as the radius $\rho$ of the cell goes to zero, these three coefficients tend to decrease linearly with $\rho$. Therefore, they will eventually dominate over the the other noise coefficients; which, due to truncation errors of affine approximations, tend to decrease quadratically with $\rho$. In that case, $h$ is closely approximated in the cell by the affine function

$$h_0 + h_1(x - x_0)/x_1 + h_2(y - y_0)/y_2 + h_3(z - z_0)/z_3$$

and the ratios $(h_1/x_1, h_2/y_2, h_3/z_3)$ provide a good estimate of the gradient $\nabla h$ in the cell.

For surfaces of moderate complexity, the pixel-sized cells that are produced by the adaptive enumeration algorithm are usually small enough to allow for this estimation. Figure 6 shows a $256 \times 256$ image of the union of two spheres rendered using this technique.

## 7. Conclusion

Numerical experiments show that affine arithmetic is indeed more accurate than standard interval arithmetic[19]. On the other hand, AA is more complex to implement and more expensive to run than IA. However, its higher accuracy is worth the extra cost for adaptive enumeration of implicit objects because the models resulting from enumeration with AA are smaller. Although the relative speeds of IA and AA depend on the particular function being evaluated, in

© The Eurographics Association 1996

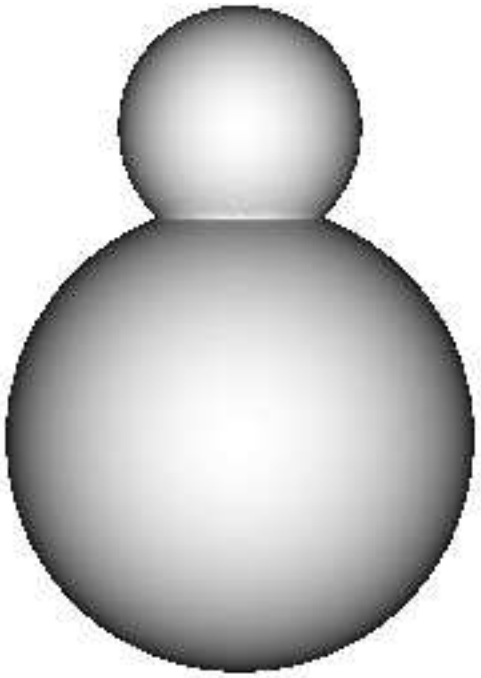

**Figure 6:** *The union of two spheres enumerated and shaded with AA.*

our implementation, AA is typically 4–5 times slower than IA. Work is in progress to improve the performance of AA.

An interesting extension of the techniques described here would be to use the extra information provided by AA to estimate surface curvature, so that the enumeration is adapted not only to the location of the surface in the ambient space, but also to its intrinsic geometry, as adaptive refinement techniques do.

We are currently investigating the use of AA in other problems that already have solutions based on IA, such as rendering implicit surfaces with ray-tracing[26] and intersection of parametric surfaces[27, 28].

## Acknowledgements

We thank João Comba, who helped implement a prototype AA package in Modula-3, and Marcus Vinicius Andrade, who helped debug the C version and wrote an implicit surface ray-tracer based on it. This work was performed on equipment kindly ceded by Paulo Correa de Mello (Chemistry Dept., PUC-Rio), and on equipment acquired within a FAPESP thematic project. The authors are partially supported by research and development grants from the Brazilian Council for Scientific and Technological Development (CNPq). Some of this research was performed while the second author was at the DEC Systems Research Center (SRC) in Palo Alto. The first author is currently a Visiting Professor in the Computer Science Department at the University of Waterloo, Canada, and holds a post-doctoral fellowship from CNPq.

## References

1. H. Ratschek and J. Rokne, *Computer Methods for the Range of Functions*. Ellis Horwood Ltd., (1984).
2. J. Bloomenthal, "Polygonization of implicit surfaces", *Computer Aided Geometric Design*, **5**(4), pp. 341–355 (1988).
3. R. E. Chandler, "A tracking algorithm for implicitly defined curves", *IEEE Computer Graphics and Applications*, **8**(2), pp. 83–89 (1988).
4. E. L. Allgower and K. Georg, *Numerical Continuation Methods: An Introduction*. Springer-Verlag, (1990).
5. D. P. Dobkin, S. V. F. Levy, W. P. Thurston, and A. R. Wilks, "Contour tracing by piecewise linear approximations", *ACM Transactions on Graphics*, **9**(4), pp. 389–423 (1990).
6. M. Hall and J. Warren, "Adaptive polygonalization of implicitly defined surfaces", *IEEE Computer Graphics and Applications*, **10**(6), pp. 33–42 (1990).
7. L. Velho, "Adaptive polygonization of implicit surfaces using simplicial decomposition and boundary constraints", in *Eurographics '90*, pp. 125–136, (1990).
8. M. F. W. Schmidt, "Cutting cubes – visualizing implicit surfaces by adaptive polygonization", *The Visual Computer*, **10**(2), pp. 101–115 (1993).
9. K. G. Suffern, "Quadtree algorithms for contouring functions of two variables", *The Computer Journal*, **33**, pp. 402–407 (1990).
10. K. G. Suffern and E. D. Fackerell, "Interval methods in computer graphics", *Computers & Graphics*, **15**, pp. 331–340 (1991).
11. T. Duff, "Interval arithmetic and recursive subdivision for implicit functions and constructive solid geometry", *Computer Graphics (SIGGRAPH '92 Proceedings)*, **26**(2), pp. 131–138 (1992).
12. J. M. Snyder, "Interval analysis for computer graphics", *Computer Graphics (SIGGRAPH '92 Proceedings)*, **26**(2), pp. 121–130 (1992).
13. G. Taubin, "Rasterizing algebraic curves and surfaces", *IEEE Computer Graphics and Applications*, **14**, pp. 14–23 (1994).

© The Eurographics Association 1996

14. H. Samet, *Applications of Spatial Data Structures*. Addison-Wesley, (1990).

15. H. Fuchs, Z. M. Kedem, and B. F. Naylor, "On visible surface generation by a priori tree structures", *Computer Graphics (SIGGRAPH '80 Proceedings)*, **14**(3), pp. 124–133 (1980).

16. J. L. Bentley, "Multidimensional binary search trees used for associative searching", *Communications of the ACM*, **18**, pp. 509–517 (1975).

17. R. M. Persiano, M. Salim, and L. P. Bueno, "Boundary evaluation of CSG solids by simplicial subdivision", in *COMPUGRAPHICS '91*, vol. II, pp. 220–229, (1991).

18. R. E. Moore, *Interval Analysis*. Prentice-Hall, (1966).

19. J. L. D. Comba and J. Stolfi, "Affine arithmetic and its applications to computer graphics", in *Proceedings of VI SIBGRAPI (Brazilian Symposium on Computer Graphics and Image Processing)*, pp. 9–18, (1990). Available at `http://dcc.unicamp.br/~stolfi/`.

20. E. R. Hansen, "A generalized interval arithmetic", *Lecture Notes in Computer Science*, **29**, pp. 7–18 (1975).

21. B. Wyvill and G. Wyvill, "Field functions for implicit surfaces", *The Visual Computer*, **5**(1/2), pp. 75–82 (1989).

22. A. S. Glassner, "Space subdivision for fast ray tracing", *IEEE Computer Graphics and Applications*, **4**(10), pp. 15–22 (1984).

23. G. Wyvill, T. L. Kunii, and Y. Shirai, "Space division for ray tracing in CSG", *IEEE Computer Graphics and Applications*, **6**(4), pp. 28–34 (1986).

24. M. Levoy, "Display of surfaces from volume data", *IEEE Computer Graphics and Applications*, **8**(3), pp. 29–37 (1988).

25. W. Baur and V. Strassen, "The complexity of partial derivatives", *Theoretical Computer Science*, **22**, pp. 317–330 (1983).

26. D. P. Mitchell, "Robust ray intersection with interval arithmetic", in *Proceedings of Graphics Interface '90*, pp. 68–74, (May 1990).

27. M. Gleicher and M. Kass, "An interval refinement technique for surface intersection", in *Proceedings of Graphics Interface '92*, pp. 242–249, (May 1992).

28. L. H. de Figueiredo, "Surface intersection using affine arithmetic", in *Proceedings of Graphics Interface '96*, (May 1996).

© The Eurographics Association 1996