# Efficient Lipschitz function evaluation for CSG implicit surfaces

Phap Nguyen and Huub van de Wetering*
Department of Mathematics and Computing Science
Eindhoven University of Technology

## Abstract

The rendering of an implicit surface requires many function evaluations of the underlying implicit function. The time efficiency of the rendering is dominated by *the number of function evaluations* times *the efficiency of these evaluations.* If the implicit surface is created by means of a CSG model the function evaluations can be made more efficient if the underlying function is a Lipschitz function. Here implicit surfaces defined as a blended CSG model with Lipschitz functions are considered. It is shown how bounding volumes can be integrated in this framework and how they can be exploited to improve the evaluation efficiency.

## 1 Introduction

Rendering images of implicit surfaces (see [1]) requires many function evaluations of the underlying implicit function. By reduction of the number of evaluations and by improving the evaluation efficiency, the rendering cost can be reduced. In this paper we will consider implicit functions based on Lipschitz functions and CSG expressions as they are defined in [3]. For these functions we give both a method for reducing the number of evaluations by locally replacing the Lipschitz function by a distance function to a bounding volume and a method for increasing the evaluation speed by lazy evaluation of the CSG expression. This method was originally applied in [7] to collision detection of implicit surfaces.

In section 2 we introduce the basic concepts. In section 3 we show how to improve a Lipschitz function, followed by section 4 where lazy evaluation is handled. The bounding volumes we use in our implementation are given in section 5. The results and conclusion are given in section 6.

*wstahw@win.tue.nl

## 2 Definitions

In this paragraph we give some definitions and properties concerning implicit functions and define the implicit functions we consider in the rest of the paper.

An *implicit function* $f$ is a continuous real function on $\Re^3$. With $f$ we define both an *implicit volume* $V_f$

$$V_f = \left\{ x \in \Re^3 \middle| f(x) \geq 0 \right\}. \tag{1}$$

and an *implicit surface* $S_f$

$$S_f = \left\{ x \in \Re^3 \middle| f(x) = 0 \right\}. \tag{2}$$

So, an implicit function is negative outside the implicit volume and zero on the surface.

For each (closed) set $A \subset \Re^3$ we define an implicit function $d_A$ with $A$ as its implicit volume as follows

$$d_A(x) = \begin{cases} + \inf \left\{ \|y - x\| \middle| y \notin A \right\} & \text{if } x \in A \\ - \inf \left\{ \|y - x\| \middle| y \in A \right\} & \text{if } x \notin A \end{cases} \tag{3}$$

with $x \in \Re^3$. The function $d_A$ is called the *signed distance* to $A$. It is easy to see that for the signed distance the following property holds for all points $x \in \Re^3$ and all subsets $A$ and $B$ of $\Re^3$ with $A \subset B$.

$$d_A(x) \leq d_B(x) \tag{4}$$

A distance function $d_{V_f}$ is denoted by $d_f$.

We introduce for each implicit function $f$ a bounding volume $B_f \supset V_f$, and denote the distance function $d_{B_f}$ of the bounding volume by $b_f$. All our results are equally valid whether we choose bounding boxes, bounding spheres, or any other bounding volume. We only require that the bounding boxes are consistent; that is, $V_f \subset V_g$ implies that $B_f \subset B_g$.

Since $V_f \subset B_f$, for all points $x \in \Re^3$, the following holds according to equation 4

$$d_f(x) \leq b_f(x). \qquad (5)$$

A function $f$ is called Lipschitz with (Lipschitz) constant $\lambda$ if for all $x, y \in \Re^3$,

$$\left| f(x) - f(y) \right| \leq \lambda \|x - y\|. \qquad (6)$$

Every Lipschitz function can be turned in a Lipschitz function with constant 1 by dividing by $\lambda$. Hence, generality is not lost by only considering Lipschitz functions with constant 1; we will do so in the rest of this paper.

Distance functions are examples of Lipschitz functions. They are also an upperbound for Lipschitz functions: for a Lipschitz function $f$ and for $x \in \Re^3$

$$|f(x)| \leq |d_f(x)|. \qquad (7)$$

Geometrically this property states that the sphere with radius $|f(x)|$ and center $x$ is either completely inside or completely outside the implicit volume of $f$.

In [3] new functions are created by combining Lipschitz functions by CSG operators supporting soft blending: $f \cap g$, $f \cup g$, and $f \setminus g$. These operators are defined by

$$\begin{aligned} f \cap g &= \min\{f, g\} - \beta(|f - g|) \\ f \cup g &= \max\{f, g\} + \beta(|f - g|) \\ f \setminus g &= \min\{f, -g\} - \beta(|f + g|). \end{aligned} \qquad (8)$$

The resulting functions are Lipschitz functions if the blending function $\beta$, a real function on $\Re^+$, fulfills the following requirement for its derivative: $-1 \leq \beta' \leq 0$ (see [3] for proof and examples). Using the operators CSG expressions can be built; the basic functions in these expressions can be, for instance, distance functions of geometric primitives like a box, sphere, cylinder, et cetera. We make an extra assumption: *the blending function has a finite support* ; that is, for a suitable $\chi \geq 0$ and for all $x \geq \chi$ the blending function vanishes : $\beta(x) = 0$. Due to the restrictions on the derivative $\beta'$, now also $0 \leq \beta(x) \leq \chi$ holds on the complete domain.

Straightforward evaluation of surfaces defined by CSG operators directly from (8) is time-consuming. In the following we show how the evaluation efficiency can be increased by enhancing the definitions of Lipschitz functions such that they form a better approximation of the corresponding distance function. And finally, we give methods for using these improved functions for lazy evaluations.

## 3 Improving Lipschitz functions

The absolute value of a Lipschitz function value $f(x)$ is a lower bound for the distance of point $x$ to the implicit surface. So, the sphere with radius $|f(x)|$ and center $x$ is disjoint with the implicit surface. This property can be used to prune space in a search for surface points (see [6], [5], and [8]). It is most successfully applied if the Lipschitz function value equals the distance to the surface. However, the *composite* functions as defined in (8) under certain circumstances, even if the basic functions are distance functions, differ greatly from the surface distance. This is illustrated in figure 1. There the volumes $V_f$ and $V_g$ are intersected and produce the volume $V_h$. The implicit functions $f \cap g$ and $h$ differ although they define the same volume. Clearly, in the example, the value $h(x)$ is the preferable function value over $(f \cap g)(x)$=$\min\{f(x), g(x)\}$ since its absolute value is larger. So, there is room for improvement of $f \cap g$. We give a simple, yet general, approach for (possibly) improving our Lipschitz functions in such a way that the absolute function values become larger while on the other hand the implicit volumes remain the same. We define the *improved* function $\bar{f}$ of the Lipschitz function $f$ as follows

$$\bar{f}(x) = \min\{f(x), b_f(x)\}. \qquad (9)$$

Firstly, note that for $x \in B_f$, $f(x) \leq b_f(x)$ holds; so, in $B_f$, $\bar{f}$ equals $f$ and, hence, $V_{\bar{f}} = V_f$. Secondly, for $x \notin B_f$, $\bar{f}(x) \leq b_f(x) \leq 0$. So, an improved function gives outside its bounding volume in absolute value at least the distance to the bounding volume.

Furthermore, $\bar{f}$ is again a Lipschitz function since it is created applying one of the operators defined in (8) (with $\beta \equiv 0$) with two Lipschitz functions as operands.

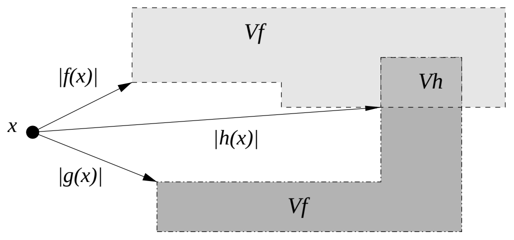

Figure 1:

So, improving an implicit function results in an implicit function with a possibly larger absolute function value while maintaining the same implicit volume.

A function $f$ with $f = \bar{f}$, is called *acceptable*. For instance, distance functions and improved functions are acceptable. After replacing the composite functions in a CSG expression by their improved (and acceptable) versions and by using acceptable basic functions, we only deal with acceptable functions. If we consider, for instance, the union of two *acceptable* functions we see that improving has no effect:

$$\overline{f \cup g} = f \cup g \tag{10}$$

This observation results from the acceptability of $f$ and $g$ and the consistency of the bounding boxes: $f(x) \leq b_f(x) \leq b_{f \cup g}(x)$ and, equivalently, $g(x) \leq b_g(x) \leq b_{f \cup g}(x)$.

Sometimes improving has also no effect on the difference operator (again using acceptability of the operands).

$$B_f = B_{f \setminus g} \Rightarrow \overline{f \setminus g} = f \setminus g \tag{11}$$

This follows from the acceptability of $f$ : $f(x) \leq b_f(x) = b_{f \setminus g}(x)$.

Improving the composite functions in a CSG expressions results in acceptable (sub)functions; this property is used in the next section to prove the lazy evaluation techniques we introduce there.

## 4 Lazy evaluation

The evaluation of our Lipschitz functions based on CSG can be done recursively by evaluating the sub-expressions. In evaluating these sub-expression in a lazy manner, we attempt only to evaluate those expressions that contribute to the final result. This lazy evaluation is based on bounding the values of the sub-expression and deciding on these bounds whether or not the full expression needs to be evaluated. In this section we give a lazy evaluation rule both for the union and the difference operator. Furthermore, we give small algorithms for applying these rules. In section 5 we give an algorithm for the intersection operator.

For the union operator the next rule for lazy evaluation holds

$$g(x) \geq b_f(x) + \chi \Rightarrow \overline{f \cup g}(x) = g(x). \tag{12}$$

This rule can be simply applied as given in algorithm 1. Note that care is taken to evaluate the most expensive functions $f$ and $g$ only if strictly necessary. In fact, the above rule with $f$ and $g$ swapped is also a valid rule. Care must be taken that the rule that is most likely to apply is used. We assume that if $b_f(x) < b_g(x)$ that most likely $f(x) < g(x)$, in which case it is best to apply the rule as given above. Below we give the proof of the lazy evaluation rule for the union operator.

```
union($f$,$g$,x)
   bf := $b_f$(x);
   bg := $b_g$(x);
   if bf $<$ bg
     g := $g$(x);
     if g $\geq$ bf+$\chi$ return g;
     else
       f := $f$(x);
       b := $\beta$(|f-g|);
       return max(f,g)+b;
   else ...{ swap $f$ and $g$ }
```

Algorithm 1: lazy union

**Proof 1** *Let $g(x) \geq b_f(x) + \chi$. Hence, using the acceptability of $f$, $g(x) - f(x) \geq g(x) - b_f(x) \geq \chi \geq 0$. From which follows that both $\beta(|f(x) - g(x)|) = 0$ and $g(x) \geq f(x) + \chi \geq f(x)$ hold. Furthermore, from the acceptability of $g$, it follows that $g(x) \leq b_g(x)$; from the consistency of the bounding boxes follows that $b_g(x) \leq b_{f \cup g}(x)$ since $B_g \subset B_{f \cup g}$. And finally, we see that $\overline{f \cup g}(x) = \min\{\max\{f(x), g(x)\}, b_{f \cup g}\} = g(x)$.*

Lazy difference evaluation can be done according to the following rule. The resulting algorithm is shown in algorithm 2.

$$\begin{aligned} & b_f(x) \leq -b_g(x) - \chi \\ \Rightarrow \quad & \overline{f \setminus g}(x) = min\{f(x), b_{f \setminus g}(x)\} \end{aligned} \tag{13}$$

The proof of this rule is given below.

**Proof 2** *Suppose $b_f(x) \leq -b_g(x) - \chi$. Using that $\beta(x) \leq \chi$ and using the acceptability of $f$ and $g$, we find that $-g(x) - \chi \geq b_f(x) \geq f(x)$. And hence, $-g(x) - f(x) \geq \chi \geq 0$. So, under the given condition, $\beta(|f(x) + g(x)|) = 0$. Furthermore, since (again using bounding box consistency), $b_{f \setminus g}(x) \leq b_f(x) \leq -b_g(x) - \chi \leq -g(x)$, we finish the proof with $\overline{f \setminus g}(x) = \min\{f(x), -g(x), b_{f \setminus g}(x)\} = \min\{f(x), b_{f \setminus g}(x)\}$.*

```
difference($f$,$g$,x)
   bfg := $b_{f\backslash g}$(x);
   bf := $b_f$(x);
   bg := $b_g$(x);
   f := $f$(x);
   if bf < -bg - $\chi$
     return min(f,bfg);
   else
     g := $g$(x);
     b := $\beta$(|f+g|);
     return min(f-b,-g-b,bfg);
```

Algorithm 2: lazy difference

## 5 Bounding volume computations

Both function improvement and lazy evaluation benefit from bounding volumes that fit tightly. In our implementation we use axis-aligned bounding boxes since computations on these boxes are both simple and efficient. Furthermore, we use the S-bounds method as introduced in [2] for computing efficient bounding boxes for a CSG-expression and its subexpressions. This method is based on processing the CSG expression hierarchy a few times from bottom to top and back, updating each time in each node the bounding box based upon the bounding boxes currently known of either its parent or its children. In this way each node obtains a small and, for our purposes, hence, efficient bound. From [2] we know that this method converges in a limited number of traversals through the hierarchy. The S-bounds method requires only that initially the bounding boxes of the basic functions in the expression are known and that the resulting bounding box can be computed for each CSG operators given the bounding boxes of the operands. This is in our case not completely trivial since our operators allow soft-blending. Because the blending functions are bounded ($0 \leq \beta(x) \leq \chi$), and because we use Lipschitz functions, we can give a correct bounding box distance for each of the operators, given the bounding box distances of the operators without blending (indicated by the underlined operators), as follows

$$\begin{aligned} b_{f\cup g} &= b_{f\underline{\cup}g} + \chi \\ b_{f\cap g} &= b_{f\underline{\cap}g} \\ b_{f\backslash g} &= b_{f\underline{\backslash}g}. \end{aligned} \tag{14}$$

The corresponding bounding boxes can be adapted accordingly.

As a result of using S-bounds we can see that for an intersection operator holds that $B_{f\cap g} = B_f = B_g$ since if one of these boxes is smaller it can also be used instead of the other boxes without changing the result of the CSG expression. Consequently, $\overline{f \cap g} = \min\{f, g, b_{f\cap g}\} = \min\{f, g, b_f\} = min\{f, g\}$ if $f$ is acceptable. Which leaves us with algorithm 3 for computing the intersection.

```
intersection($f$,$g$,x)
   f := $f$(x);
   g := $g$(x);
   b := $\beta$(|f-g|);
   return min(f-b,g-b);
```

Algorithm 3: intersection

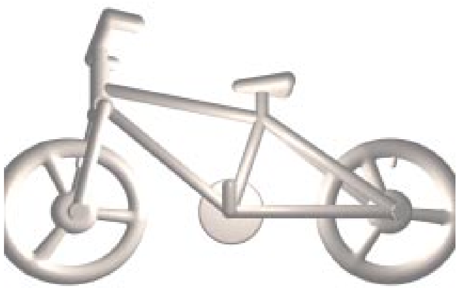

Figure 2: bicycle model consisting of unions of 37 primitives

## 6 Results and conclusions

The function improvement and lazy evaluation have been implemented in a module for the ray-tracer POV-ray ([4]). A ray-tracer provides a good environment for testing the techniques since many function evaluations are performed. Sphere tracing [5] is used for the intersection computations.

We give the results on speedups for three models given in the figures 3, 2, and 4. The first model consists of only unions of primitives, the second example is the difference of a cylinder and the union of three cylinders forming a cross, the third example is the intersection of a torus with the union of a sphere and two cylinders. These models have been

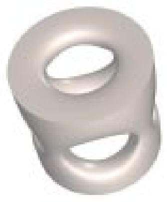

Figure 3: cylinder with a cross-shaped hole subtracted

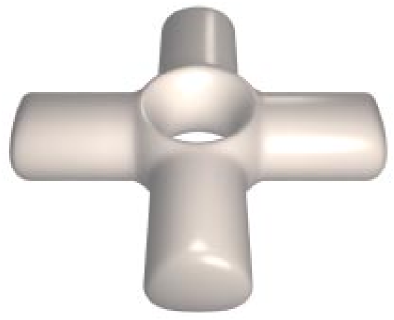

Figure 4: intersection of a torus with a sphere and two cylinders

rendered both with and without the optimizations given in this paper. The models are rendered in 36 different positions with an image size of 120x120 pixels and the resulting measurements are given in the table.

| figure/ | # evaluations/$10^3$ | | time | |
|---|---|---|---|---|
| method | toplevel | total | (s) | gain |
| 3 old | 6769 | 47384 | 237 | |
| new | 6782 | 18582 | 95 | 60% |
| 2 old | 5926 | 485944 | 2136 | |
| new | 5914 | 129060 | 524 | 75% |
| 4 old | 7090 | 63815 | 245 | |
| new | 7091 | 48074 | 178 | 27% |

It shows that in all given cases the improvement in time is significant. The column ”toplevel evaluations” gives the total number of times the corresponding CSG expression is evaluated, the column ”total evaluations” gives the number of expression evaluations including those of subexpressions. So, we can see that lazy evaluation works since the total number of evaluations drastically goes down. In all three cases we see that the number of toplevel calls remains more or less the same; so there is no direct proof that improving functions results in more efficiency. However, our lazy evaluation is only valid for acceptable functions and it is by improving that we get to work with only acceptable functions.

Lazy evaluation works better for larger expressions, in which it is possible to decline on evaluating large subtrees: the bicycle example has the the greatest speedup of the given examples. For small expressions the overhead introduced may make lazy evaluation less effective.

Concluding, we can say that improving the Lipschitz functions allowed us to use lazy evaluation for CSG expressions resulting in considerable speedup of the total evaluation time.

## References

[1] Jules Bloomenthal, Chandrajit Bajaj, Jim Blinn, Marie-Paule Cani-Gascuel, Alyn Rockwood, Brian Wyvill, and Geoff Wyvill, editors. *Introduction to Implicit Surfaces*. Series in Computer Graphics and Geometric Modeling. The Morgan Kaufmann, 1997. ISBN 1-55860-233-X.

[2] Stephen Cameron. Efficient bounds in constructive solid geometry. *IEEE Computer Graphics and Applications*, 11(3):68–74, May 1991.

[3] Daniel Dekkers, Kees van Overveld, and Rob Golsteijn. Combining CSG modeling with soft blending using Lipschitz-based implicit surfaces. *to be published; see http://wwwcg.win.tue.nl/Papers*.

[4] Persistence Of Vision development team. Pov-ray : Persistence of vision. http://www.povray.org.

[5] John C. Hart. Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12(9):527–545, 1996. ISSN 0178-2789.

[6] Devendra Kalra and Alan H. Barr. Guaranteed ray intersections with implicit surfaces. In

Jeffrey Lane, editor, *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23, pages 297–306, July 1989.

[7] V.P. Nguyen. Collision detection of Lipschitz implicit surfaces. Master's thesis, Eindhoven University of Technology, Department of Mathematics and Computing Science, 1998.

[8] Huub van de Wetering, Martijn de Kort, and Kees van Overveld. Scan-conversion of implicit surfaces with Lipschitz condition. In *Implicit Surfaces '98 The Third International Workshop on Implicit Surfaces*, pages 39–45, June 1998. ISSN 1024-0861.