Transparent rendering and slicing of integral surfaces using per-primitive interval arithmetic

M. Aydinlilar[1], C. Zanni[1]

[1]Université de Lorraine, CNRS, Inria, LORIA

1. Supplemental Material

In this supplemental material we provide additional details on the method presented in the paper as well as additional comparisons.

Section 2 presents the information regarding to the independence of the method from the choice of kernel. Section 3 provides more detail on the derivation of the Lipschitz bound. Section 4 compares both our bound computations to the ones obtained using interval arithmetic. Section 5 presents additional results and comparisons.

2. Kernels

Both the per-primitive field bounds and the directional Lipschitz bound can be applied to other kernels whose derivative k' admit a unique maximum. For instance, among the kernels which admit closed-form expression of the integral f, Cauchy kernel defined as:

$$k(d) = N\frac{1}{\left(1+\left(\frac{d}{\sigma}\right)^2\right)^{\frac{n}{2}}} \tag{1}$$

admits a unique maximum value for k' in $d = \frac{\sigma}{\sqrt{n+1}}$.

3. Lipschitz bound

3.1. Integral evaluation

We present here additional details on the derivation of the directional Lipschitz bound, more specifically on bound computation for the following integral:

$$\int_{\mathbf{q}\in S/(\mathbf{p}-\mathbf{q})^T\mathbf{d}>0} \mathbf{d}^T\frac{\mathbf{p}-\mathbf{q}}{\|\mathbf{p}-\mathbf{q}\|}\,\mathrm{d}\mathbf{q}$$

where the line segment primitive S is parametrized by $\mathbf{q} = \mathbf{q}_0 + s\,\mathbf{u}$ and the ray is parametrized by $\mathbf{p} = \mathbf{r}(t) = \mathbf{o} + t\,\mathbf{d}$.

Let's define $\mathbf{m} = \mathbf{o} - \mathbf{q}_0$, we then have:

$$\mathbf{d}^T\frac{\mathbf{p}-\mathbf{q}}{\|\mathbf{p}-\mathbf{q}\|} = \frac{\mathbf{m}^T\mathbf{d} + t - s*\mathbf{u}^T\mathbf{d}}{\sqrt{(\mathbf{m}+t\mathbf{d}-s*\mathbf{u})^T(\mathbf{m}+t\mathbf{d}-s*\mathbf{u})}} \tag{2}$$

which is a quotient between a linear function and the square root of a quadratic function. The denominator theoretically remains positive or null and can only cancel when the numerator cancels. The integral admits a closed-form expression, however the formula is numerically unstable when the minimal distance between the ray line and segment line tends toward zero (floating point value of the denominator can become negative resulting in an infinite value for the integral). We instead bound the integrand on the domain of interest which also has the advantage of being less costly to evaluate.

Local extrema in s of Equation (2) are reached in $\pm\infty$ and in

$$s_{argmax} = -\frac{\mathbf{u}^T(\mathbf{m}-\mathbf{m}^T\mathbf{d}\mathbf{d})}{\mathbf{u}^T(\mathbf{u}-\mathbf{u}^T\mathbf{d}\mathbf{d})}$$

we can therefore easily deduce an upper bound of the integrand in the range of interest (either the value reached at the argminimum if the latter is in range or the maximal absolute value of the integrand on the boundary of the range).

4. Comparison to interval arithmetic on closed form gradient and field values

We have compared the sizes of the intervals calculated with our method to the ones calculated with interval arithmetic for both field values (figure 1) and directional derivatives (figure 2). We calculated the ratio by dividing the size of the interval range calculated

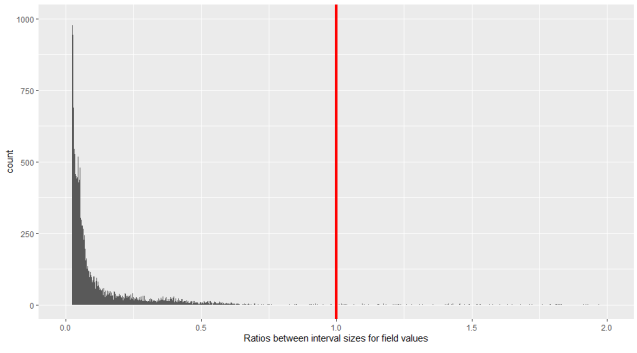

Figure 1: *Histogram of the ratio of the sizes of intervals calculated with our method to the ones calculated with interval arithmetic for compact kernel for field values.*

	[LLZ*21]	[GGPP20]	Our	Our with segment-tracing bisection [GGPP20])
jellyfish	11.1s / 21.9s	4.8s / 5.3s	2.9s / 3.5s	3.1s / 3.7s
microstructure	5.1s / 10.3s	12.9s / 16.2s	6.9s / 8.0s	6.9s / 7.9s

Table 1: *Slicing models into respectively 512 and 1024 slices with an XY resolution of* 512×512 *voxels. All methods use* 64^2 *buckets.*

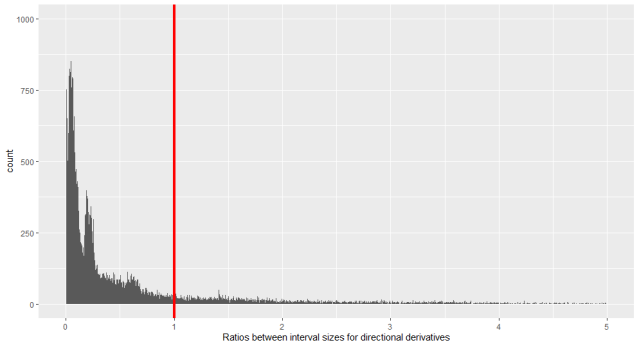

Figure 2: *Histogram of the ratio of the sizes of intervals calculated with our method to the ones calculated with interval arithmetic for compact kernel for directional derivatives.*

by our method with the range calculated by interval arithmetic. We have done this calculation for several ray configurations with different interval widths (66184 configurations).

We produce improved bounds on most cases. For example for the field values we observe that 95% of the samples have 4 times improvement (65% for directional derivatives). The red lines denote where the ratio is equal to one (i.e. when they are equal). We can observe that our samples cluster on the left side of the line, approaching to zero. This is due to the fact that interval arithmetic can create arbitrarily large or infinite bounds, therefore our division approaches zero.

5. Comparisons

We provide two additional slicing comparisons, namely application to iterative segment-tracing [GGPP20] and a combination of our method with segment-tracing bisection (presented in [GGPP20]).

For all methods, we use a minimal interval/step size equal to half the size of a voxel. For the two variants of our method, we also filter the initial ray cuts (e.g. where the kernel function of individual primitives is maximized) in order to obtain a minimal size of 8 voxels for each initial interval where bisection will be computed.

Segment-tracing: In order to provide a fairer comparison, we implemented segment-tracing with empty space skipping based on primitive supports (i.e. by relying on the same approach as our method). When applying segment-tracing to transparency, the minimal step-size is important. The iso-surface itself being a fixed point in the iteration, we need the minimal step size to allow the iteration to go past the iso-surface itself. Without a minimal step size, it is also possible to run into an infinite loop (i.e. when using floating point, small enough steps can left the current ray parameter unchanged, hence never reaching the iso-surface).

Our method with segment-tracing bisection: We can combine our method with the segment-tracing bisection presented in [GGPP20], i.e. in addition to our strategy, we can also reduce the interval size using the directional Lipschitz bounds before applying a bisection step.

Results: Runtimes are presented in Table 1. We can note that all interval arithmetic and segment-tracing methods are less sensitive than [LLZ*21] to an increase in the number of slices. Number of steps per ray are displayed in Figure 3.

From our experiment, iterative segment-tracing with integral surfaces is less efficient than our bisection-based approaches and it is also more sensitive to the minimal step size. However, it is important to note that derivation of tighter directional Lipschitz bound - provided the bounds are not too expensive to compute - is likely to make segment tracing more efficient than our method (the variant without the segment-tracing bisection).

Finally, it is also important to note that we do not use combined evaluation of field and directional Lipschitz bound in our implementation. Doing so would further reduce the run-time of the methods using segment-tracing (including ours with segment-tracing bisection). Similarly, the number of field evaluations required by our method can be reduced : in our implementation, at each bisection step, we recompute field value at intervals end-points - resulting in three field evaluations per step - while this is only required for primitives with ambiguous field variation. Storing the field values independently at interval end-points for primitives with increasing and decreasing fields would largely reduce the required number of field evaluations while having a limited impact on the memory usage.

References

[GGPP20] GALIN E., GUÉRIN E., PARIS A., PEYTAVIE A.: Segment Tracing Using Local Lipschitz Bounds. *Computer Graphics Forum* (2020). 2

[LLZ*21] LIU S., LIU T., ZOU Q., WANG W., DOUBROVSKI E. L., WANG C. C.: Memory-efficient modeling and slicing of large-scale adaptive lattice structures. *Journal of Computing and Information Science in Engineering 21*, 6 (2021). 2

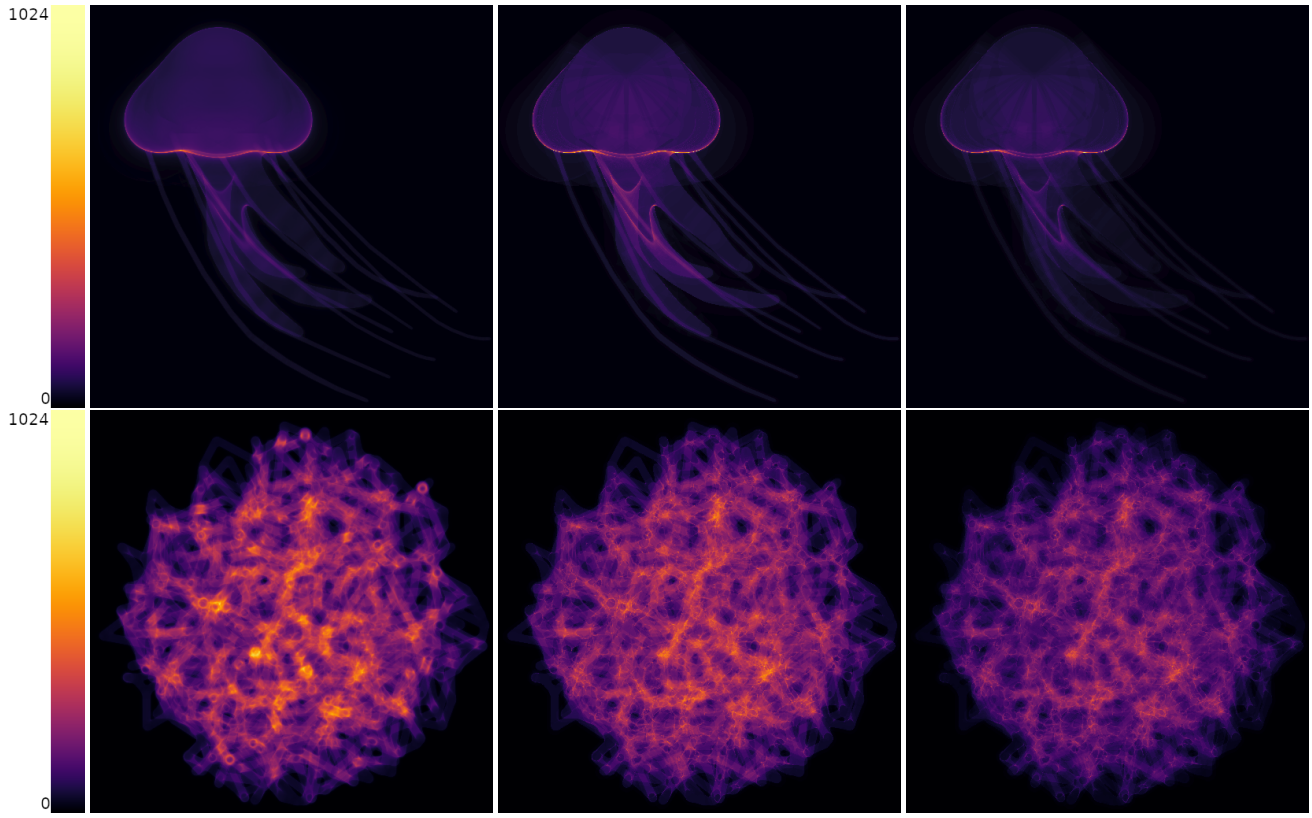

Figure 3: *Number of field evaluation for 1024 slices with segment-tracing (left), our method (middle), our method with segment-tracing bisection (right). Note that both our methods require less steps (darker images).*

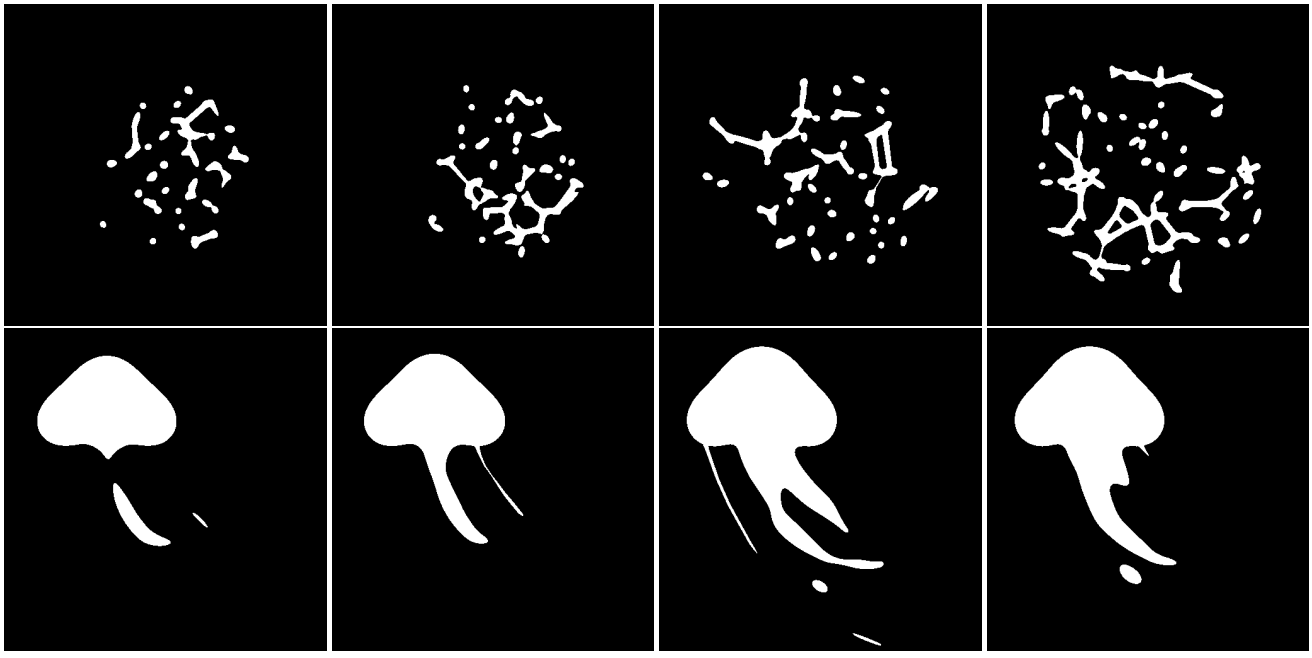

Figure 4: *Slices produced for additive manufacturing.*