Automatic Step Size Relaxation in Sphere Tracing

Róbert Bán and Gábor Valasek

Eötvös Loránd University, Hungary
{rob.ban, valasek}@inf.elte.hu

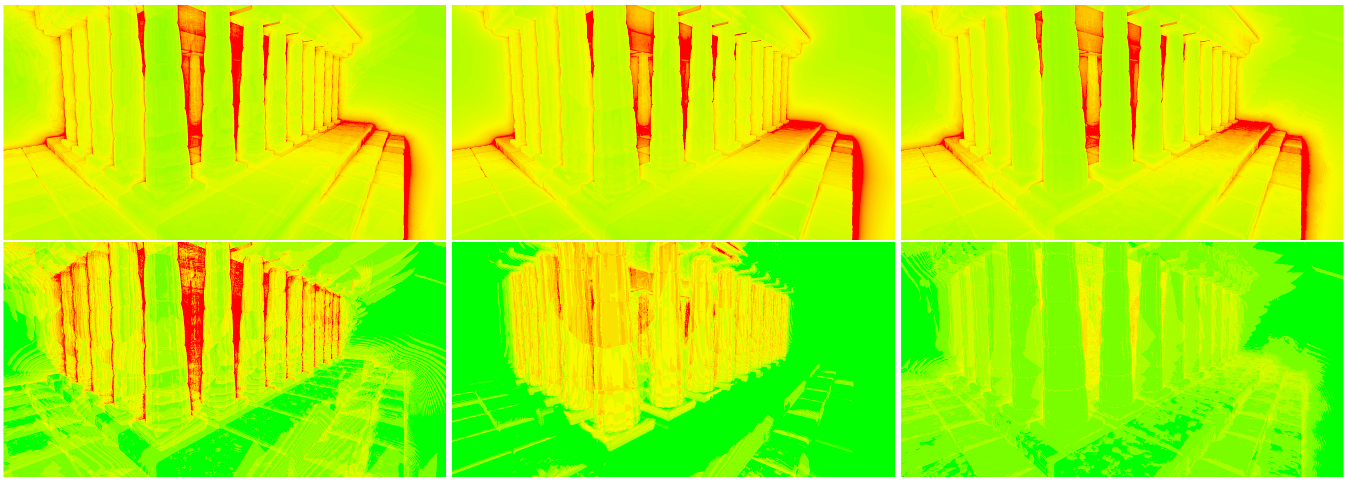

Figure 1: *Comparison of average step counts until convergence (top row) and the number of fallbacks to sphere trace steps due to disjoint unbounding spheres (bottom row) between enhanced sphere tracing with* $\omega = 0.88$ *(left), relaxed sphere tracing using* $\omega = 1.5$ *(center), and our proposed auto-relaxed sphere tracing with* $\beta = 0.3$ *(right). Green is better, red denotes a high number of occurances.*

Abstract

We propose a robust auto-relaxed sphere tracing method that automatically scales its step sizes based on data from previous iterations. It possesses a scalar hyperparemeter that is used similarly to the learning rate of gradient descent methods. We show empirically that this scalar degree of freedom has a smaller effect on performance than the step-scale hyperparameters of concurrent sphere tracing variants. Additionally, we compare the performance of our algorithm to these both on procedural and discrete signed distance input and show that it outperforms or performs up to par to the most efficient method, depending on the limit on iteration counts. We also verify that our method takes significantly fewer robustness-preserving sphere trace fallback steps, as it generates fewer invalid, over-relaxed step sizes.

CCS Concepts

• ***Computing methodologies*** $\rightarrow$ ***Ray tracing; Shape modeling;***

1. Introduction

Signed distance functions (SDF) are versatile scene descriptions used in a wide variety of applications, such as collision detection, physical simulation [MEM*20], auxiliary [Wri15] and primary shape representation [Aal18]. Our paper focuses on the efficient rendering of scenes defined by SDFs. The input SDF may be continuous, e.g. procedural, or discrete, where the SDF values are stored in a 3D texture.

We propose a method that uses automatic step size relaxation to speed up sphere tracing in high performance rendering scenarios. Our solution is inspired by gradient methods with momentum employing exponential decay. It offers a hyperparameter, however, its effect on the performance of the method is smaller than the hyperparameters of other investigated rendering algorithms.

We provide a brief summary of the most efficient sphere tracing methods in Section 2, focusing on algorithmic descriptions.

DOI: 10.2312/egs.20231014

Section 3 describes our proposed auto-relaxation sphere tracing algorithm. We compare it to three sphere tracing variants in Section 4 and summarize our findings in Section 5.

2. Background and Related Work

An $f : \mathbb{R}^3 \to \mathbb{R}$ function is a *distance function* if

$$f(\boldsymbol{x}) = d(\boldsymbol{x}, \{f = 0\}) = \inf\{|\boldsymbol{x} - \boldsymbol{y}| \mid f(\boldsymbol{y}) = 0\}, \quad (1)$$

holds and f is a signed distance function if it is continuous and its absolute value, $|f|$, is a distance function. We assume that the SDF is negative for interior and positive for exterior points.

Let $\boldsymbol{p}_0 \in \mathbb{R}^3, \boldsymbol{v} \in \mathbb{R}^3$ denote a ray starting point and direction. Let us parametrize the ray as $\boldsymbol{s}(t) = \boldsymbol{p}_0 + t\boldsymbol{v} \quad (t \geq 0)$. Then the basic ray-surface intersection problem is to find the smallest $t \geq 0$ ray parameter such that $f(\boldsymbol{s}(t)) = 0$.

The common input arguments for all discussed tracing algorithms are the $f : \mathbb{R}^3 \to \mathbb{R}$ SDF, the $\boldsymbol{s} : \mathbb{R}^+ \to \mathbb{R}^3$ ray, and the termination parameters $\varepsilon \in \mathbb{R}^+, t_{max} \in \mathbb{R}^+$ and $i_{max} \in \mathbb{N}^+$. Possible stopping conditions: a) surface hit, b) miss, c) non-converged.

The SDF returns the signed distance to the closest boundary point. Hart showed in [Har96] that this property facilitates efficient and robust rendering of exact and lower bound SDF functions. The key observation was that the interior of the sphere centered at $\boldsymbol{x} \in \mathbb{R}^3$ with radius $|f(\boldsymbol{x})|$ may not contain boundary or interior points of any volume in the scene if $f(\boldsymbol{x}) > 0$. Thus, $f(\boldsymbol{x})$ is a safe stepping distance. We refer to these spheres as *unbounding spheres*. Algorithm 1 summarizes the method.

Algorithm 1 Hart's sphere trace algorithm

```
1: Input: $f : \mathbb{R}^3 \to \mathbb{R}$ SDF; $\boldsymbol{s} : \mathbb{R} \to \mathbb{R}^3$ ray; $\varepsilon, t_{max} \in \mathbb{R}^+, i_{max} \in \mathbb{N}^+$ termination conditions
2: Output: $t \geq 0$ ray parameter of intersection
3:
4: $t := 0, \quad r := 0, \quad i := 0$
5: repeat
6:     $r := f(\boldsymbol{s}(t))$
7:     $t := t + r$
8:     $i := i + 1$
9: until $r \leq \varepsilon \ \vee \ t \geq t_{max} \ \vee \ i \geq i_{max}$
```

Keinert et al. [KSK*14] proposed a step-size over-relaxation-based method to speed up sphere tracing, shown in Algorithm 2. They use an $\omega \in [1,2)$ constant step size multiplier during trace and fall back to a traditional sphere trace step if the next guess produces an unbounding sphere that is disjoint from the previous one.

Bálint et al. [BV18] use data from the current and previous steps to construct a linear approximation to the SDF function along the ray and use it to infer the closest ray parameter where the unbounding sphere is tangent to the current one. This is an optimal step size if the approximated surface is planar. A basic sphere trace step is taken if the unbounding sphere at the guess is disjoint from the current one. This is summarized in Algorithm 3.

We refer to Hart's algorithm as basic, Keinert et al.'s as relaxed, and Bálint et al.'s as enhanced sphere tracing, similarly to [BV18].

Algorithm 2 Keinert's relaxed sphere trace algorithm

```
1: Input: $f : \mathbb{R}^3 \to \mathbb{R}$ SDF; $\boldsymbol{s} : \mathbb{R} \to \mathbb{R}^3$ ray; $\varepsilon, t_{max} \in \mathbb{R}^+, i_{max} \in \mathbb{N}^+$ termination conditions; $\omega \in [1,2)$ relaxation parameter
2: Output: $t \geq 0$ ray parameter of intersection
3:
4: $t := 0, \quad r := 0, \quad i := 0, \quad z := 0$
5: repeat
6:     $T := t + z, \quad R := f(\boldsymbol{s}(T))$          ▷ Try the calculated step
7:     if $z \leq r + |R|$ then          ▷ Check if the step is correct
8:         $z := \omega \cdot R$          ▷ Calculate next step
9:         $t := T, \quad r := R$          ▷ Apply current step
10:    else          ▷ If the step is not correct,
11:        $z := r$          ▷ revert to basic ST step
12:    end if
13:    $i := i + 1$
14: until $r \leq \varepsilon \ \vee \ t + r \geq t_{max} \ \vee \ i \geq i_{max}$
```

Algorithm 3 Bálint and Valasek's enhanced sphere trace algorithm

```
1: Input: $f : \mathbb{R}^3 \to \mathbb{R}$ SDF; $\boldsymbol{s} : \mathbb{R} \to \mathbb{R}^3$ ray; $\varepsilon, t_{max} \in \mathbb{R}^+, i_{max} \in \mathbb{N}^+$ termination conditions; $\omega \in (0,1]$ relaxation parameter
2: Output: $t \geq 0$ ray parameter of intersection
3:
4: $t := 0, \quad r := f(\boldsymbol{s}(t)), \quad i := 1, \quad z := r$
5: while $r > \varepsilon \ \wedge \ t + r < t_{max} \ \wedge \ i < i_{max}$ do
6:     $T := t + z, \quad R := f(\boldsymbol{s}(T))$          ▷ Try the calculated step
7:     if $z \leq r + |R|$ then          ▷ Check if the step is correct
8:         $z := R + \omega \cdot R \frac{T-t+R-r}{T-t-(R-r)}$          ▷ Calculate next step
9:         $t := T, \quad r := R$          ▷ Apply current step
10:    else          ▷ If the step is not correct,
11:        $z := r$          ▷ revert to basic ST step
12:    end if
13:    $i := i + 1$
14: end while
```

3. Automatic relaxation

Our proposed method uses a step calculation akin to [BV18]. The idea is based on the observation that the optimal step size for planes is characterized as follows. If we take a step and evaluate the SDF, the unbounding sphere at the new position should overlap with the previous one to ensure that the line segment between the two sphere centers is disjoint from the volume. The largest such step is maximal when the two spheres are tangential, i.e. they touch. For planes, the SDF along the ray is a linear function of the ray parameter:

$$F(t) := f(\boldsymbol{s}(t)) = m \cdot t + b \quad (2)$$

This function can be calculated from two evaluations of the SDF along the ray. Let $t_0, t_1 \in \mathbb{R}^+$ denote ray parameters, $t0 \neq t1$, and $r_0 = F(t_0), r_1 = F(t_1)$ distance values. Then

$$m = \frac{r_1 - r_0}{t_1 - t_0} \quad \text{and} \quad b = \frac{r_0 t_1 - r_1 t_0}{t_1 - t_0}. \quad (3)$$

The optimal step would move to a t_2 with $r_2 = F(t_2)$ where $t_1 + r_1 = t_2 - r_2$ holds. Solving for t_2, we get $t_2 = t_1 + \frac{2r_1}{1-m}$.

This is the same step size as in line 8 of Algoritm 3 if $\omega = 1$.

Algorithm 4 Our proposed auto-relaxation sphere trace algorithm

```
1:  Input: $f : \mathbb{R}^3 \to \mathbb{R}$ SDF; $\mathbf{s} : \mathbb{R} \to \mathbb{R}^3$ ray; $\varepsilon, t_{max} \in \mathbb{R}^+, i_{max} \in \mathbb{N}^+$ termination conditions; $\beta \in (0,1)$ exponential averaging coefficient
2:  Output: $t \geq 0$ ray parameter of intersection
3:
4:  $t := 0, \quad r := f(\mathbf{s}(t)), \quad i := 1, \quad z := r, \quad m := -1$
5:  while $r > \varepsilon \ \wedge \ t + r < t_{max} \ \wedge \ i < i_{max}$ do
6:      $T := t + z, \qquad R := f(\mathbf{s}(T))$          ▷ Try the calculated step
7:      if $z \leq r + |R|$ then                             ▷ Check if the step is correct
8:          $M := \frac{R-r}{T-t}$
9:          $m := (1-\beta)m + \beta M$
10:         $t := T, \qquad r := R$                          ▷ Apply current step
11:     else                                                 ▷ If the step is not correct,
12:         $m := -1$                                        ▷ revert to basic ST step
13:     end if
14:     $z := \frac{2r}{1-m}$                                ▷ Calculate next step
15:     $i := i + 1$
16: end while
```

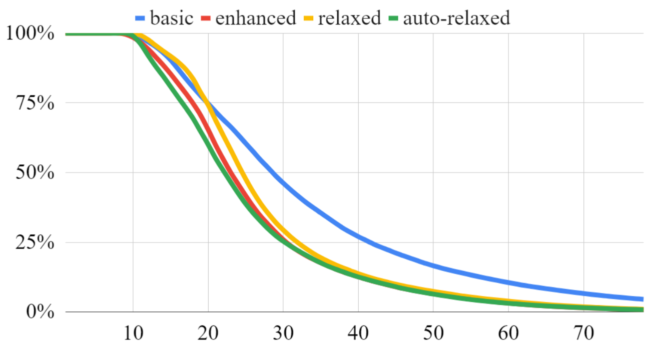

Figure 2: *The ratio of non-converged (active) pixels as a function of step counts, measured on the Temple Scene [TGM22]. Smaller numbers are better. Our auto-relaxed sphere tracing method (green) performs up to par with enhanced sphere tracing ($\omega = 0.88$) and outperforms it on smaller step counts (12-30).*

In general, this step is too large, e.g. if the input is not a plane or not convex. In [BV18], this is solved by adding an $\omega < 1$ constant relaxation parameter. Instead, we propose to use a dynamic multiplier, inspired by how gradient methods with momentum accumulate gradients via exponential averaging. Our algorithm keeps track of the approximate slope m, and we update the approximation using slope M, calculated from the last and the current step. The new value of m is the linear interpolant of m and M with the exponential averaging coefficient $\beta \in (0,1)$:

$$m := (1-\beta)m + \beta M \tag{4}$$

After updating m, we use it to calculate the optimal step as described above. When we need to perform a basic sphere trace step due to an incorrectly large step, we reset m to its initial value. A value of -1 means that the step size will be exactly the SDF value. The geometrical interpretation of $m = -1$ is that the ray is perpendicular to the plane and we reach it in exactly one step.

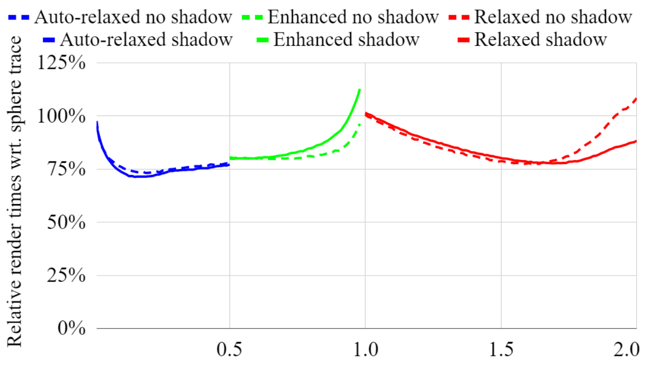

Figure 3: *The performance implications of using different trace hyperparameters, relative to sphere tracing on the 1024^3 discrete Temple SDF with and without shadows. Parameter ranges: $\beta \in (0, \frac{1}{2}]$ – auto-relaxed, $\omega \in [\frac{1}{2}, 1)$ – enhanced, $\omega \in [1, 2]$ – relaxed sphere tracing.*

4. Test Results

We implemented the sphere tracing algorithms in the Falcor [KCK*22] framework, using its DX12 backend. Our code is available on GitHub: https://github.com/Bundas102/auto-relaxed-trace.

We did detailed performance measurements for our proposed auto-relaxed sphere tracing (AR) method on two scenes from the SDF dataset [TGM22], the Girl and Temple models, using the camera positions shown on Figure 4. The tests were run on the original procedural function SDFs and their discrete versions – sampled into 1024^3 resolution 3D textures. We placed one point light source in the scene for which we traced hard shadows. We compared our method to basic (ST) [Har96], enhanced (ENH) [BV18], and relaxed (REL) [KSK*14] sphere tracing. The following table summarizes our measurements. Average frame times (*avg*) are in milliseconds, *rel* denotes the ratio of the given method of the row wrt. enhanced sphere tracing. Timings were taken on an AMD Radeon RX 5700. We set the iteration count cap at 1000 steps.

	Temple				Girl			
	Procedural		Discrete 1024^3		Procedural		Discrete 1024^3	
	avg	*rel*	*avg*	*rel*	*avg*	*rel*	*avg*	*rel*
ST	39.39	125%	8.50	112%	15.66	109%	5.54	102%
ENH	31.57	100%	7.60	100%	14.31	100%	5.43	100%
REL 1.2	33.36	106%	7.21	95%	13.97	98%	5.18	95%
REL 1.5	31.06	98%	6.46	85%	14.24	100%	4.93	91%
AR 0.3	**29.79**	**94%**	6.19	81%	**12.88**	**90%**	**4.71**	**87%**
AR 0.2	**29.79**	**94%**	**5.99**	**79%**	12.98	91%	4.72	87%

Enhanced sphere tracing [BV18] used the $\omega = 0.88$ relaxation parameter. It was handicapped by the high iteration count, as a single step of this method is more ALU intensive than a step of relaxed sphere tracing. On 32 steps, it was consistently better than relaxed sphere tracing and our method was only about 2-6% faster.

Relaxed sphere tracing [KSK*14] was sensitive to the relaxation parameter $\omega \in [1,2)$ but performed well after adjustments. We manually searched for two values that performed consistently good over scenes. Still, it can be seen on the Girl scene that while $\omega = 1.2$ per-

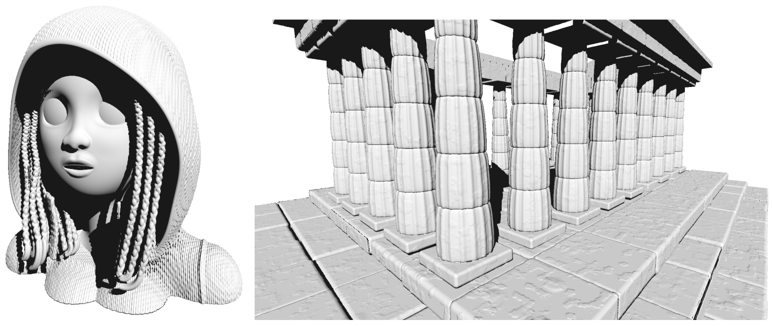

Figure 4: *Test scenes used in our performance measurements from [TGM22].*

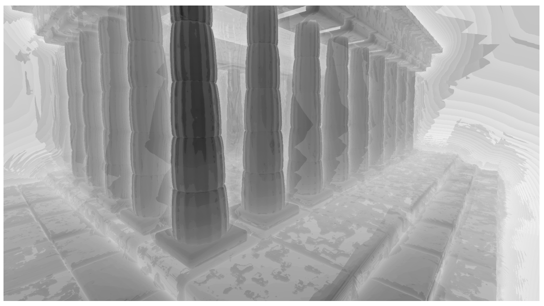

Figure 5: *Average magnitude of step size multipliers computed by our auto-relaxation method with* $\beta = 0.3$. *Black to white encodes step size multipliers from 1 to 2.*

formed better on procedural input, $\omega = 1.5$ was preferable on discrete input and they were 10% apart on the discrete Temple scene.

On the other hand, the two best manually selected parameters ($\beta = 0.2$ and 0.3) of our method stayed within 2% of each other and it consistently outperformed the other sphere tracing variants.

Figure 1 shows the behavior of the compared methods in terms of convergence speed and fallback counts. Fallbacks happen when the step size was over-relaxed and the new and previous unbounding spheres do not intersect. The larger the green areas on the heatmaps are, the better the given method is in the sense of trace steps (top row) and the number of times a fallback had to be taken (bottom row). Figure 2 shows how the ratio of non-converged pixels is distributed over the number of iterations.

Figure 3 shows how the hyperparameters of the methods affect the render performance on the Temple scene. Frame time measurements are relative to basic sphere tracing with and without shadows.

5. Conclusions

We presented a sphere tracing variant that employs exponential decay to automatically track the slope of the signed distance function along the ray that is used to compute a step size multiplier. The algorithm is made robust by taking a sphere tracing step whenever over-relaxation results in disjoint unbounding spheres, as proposed in Keinert et al.'s method [KSK*14].

We verified in various scenes that the exponential combination yields significantly fewer sphere trace fallbacks compared to enhanced [BV18] and relaxed [KSK*14] sphere tracing and performed detailed measurements on two scenes. These showed that auto-relaxed sphere tracing performs 6-19% better than enhanced and 4-5% faster than relaxed sphere tracing with 1000 iteration caps. On smaller iteration limits, such as 32, it performs similarly to enhanced sphere tracing, both outperforming relaxed sphere tracing. This suggests that the increased arithmetic computational requirements of our algorithm per iteration are masked by the less frequent occurrences of fallbacks.

The tests also showed that our method is less sensitive to the coefficient of exponential averaging, while relaxed sphere tracing requires manual adjustment of its step multiplier to individual scenes.

As such, our algorithm is applicable in both high and low iteration count configurations, matching and slightly outperforming the respective best method under the given circumstances. If manual tweaking of relaxation parameters is not feasible, our method provides more convenient automatic high performance rendering with a $\beta = 0.3$ parameter.

Acknowledgement

Supported by the ÚNKP-21-3 New National Excellence Program of the Ministry for Innovation and Technology from the source of the National Research, Development and Innovation Fund.

We would like to thank Visual Concepts for providing the AMD GPU used in the tests.

References

[Aal18] Aaltonen, Sebastian. "GPU-based clay simulation and ray-tracing tech in Claybook, Game Developers Conference 2018". *Game Developers Conference*. San Francisco, CA, Mar. 2018 1.

[BV18] Bálint, Csaba and Valasek, Gábor. "Accelerating Sphere Tracing". *EG 2018 - Short Papers*. Ed. by Diamanti, Olga and Vaxman, Amir. The Eurographics Association, 2018 2–4.

[Har96] Hart, John C. "Sphere Tracing: A Geometric Method for the Antialiased Ray Tracing of Implicit Surfaces". *The Visual Computer* 12 (1996), 527–545 2, 3.

[KCK*22] Kallweit, Simon, Clarberg, Petrik, Kolb, Craig, et al. *The Falcor Rendering Framework*. https://github.com/NVIDIAGameWorks/Falcor. Aug. 2022 3.

[KSK*14] Keinert, Benjamin, Schäfer, Henry, Korndörfer, Johann, et al. "Enhanced Sphere Tracing". *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference*. Ed. by Giachetti, Andrea. The Eurographics Association, 2014. isbn: 978-3-905674-72-9 2–4.

[MEM*20] Macklin, Miles, Erleben, Kenny, Müller, Matthias, et al. "Local Optimization for Robust Signed Distance Field Collision". *Proc. ACM Comput. Graph. Interact. Tech.* 3.1 (May 2020) 1.

[TGM22] Takikawa, Towaki, Glassner, Andrew, and McGuire, Morgan. "A Dataset and Explorer for 3D Signed Distance Functions". *Journal of Computer Graphics Techniques (JCGT)* 11.2 (Apr. 2022), 1–29. issn: 2331-7418 3, 4.

[Wri15] Wright, Daniel. "Dynamic Occlusion with Signed Distance Fields". *Advances in Real-Time Rendering in Games*. Epic Games (Unreal Engine). SIGGRAPH, 2015 1.