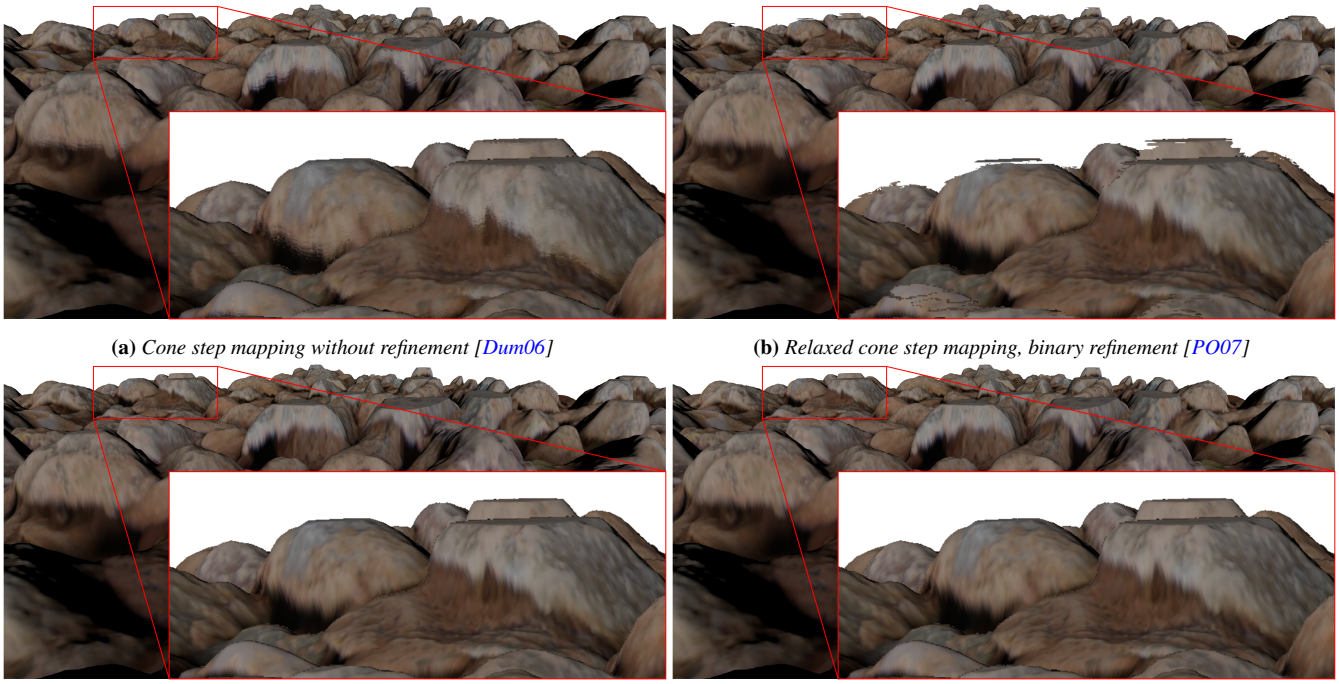
Eurographics Symposium on Rendering (2024)
E. Garces and E. Haines (Editors)

Robust Cone Step Mapping

Róbert Bán[1], Gábor Valasek[1], Csaba Bálint[1], Viktor A. Vad[1]

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



(a) *Cone step mapping without refinement [Dum06]*

(b) *Relaxed cone step mapping, binary refinement [PO07]*

(c) *Our proposed exact relaxed cone maps, binary refinement*

(d) *Our proposed cone post-processing and cell-max trace, binary refinement.*

Figure 1: *Comparing various cone step mapping algorithms with an iteration limit of 200 steps. Even conservative maps can enter the volume defined by the heightmap because bilinear interpolation may yield an incorrect unbounding cone. Note the disintegrating texture on the near vertical parts in Subfigure (a). This may be alleviated by refinement. Incorrectly generated relaxed cone maps may skip intersections as seen in Subfigure (b). We propose generation methods and a change in the trace method for better visual quality.*

Abstract

Per-pixel displacement mapping provides an alternative to high-fidelity geometry and flat textured faces with in-between performance costs. Although cone maps are known to facilitate efficient and robust rendering of height fields, we show that these cannot guarantee robustness under bilinear interpolation, and we propose corrections to this issue. First, we define an artifact-free minimum step size for the cone map tracing algorithm while remaining comparable in performance to that of Dummer. Second, we modify the cone map generation procedure so that at bilinearly interpolated values the unbounding cones remain disjoint from the heightmap, thereby preventing another source of rendering artifacts. Third, we introduce an exact method to generate relaxed cones such that any ray within intersects the heightmap at most once, in contrast to the original algorithm that is both computationally more expensive and generates incorrect relaxed cones. Finally, we demonstrate the applicability of these algorithm improvements with visual and performance comparisons in our C++ and HLSL implementation.

CCS Concepts

• ***Computing methodologies*** → ***Ray tracing; Shape modeling;***

DOI: 10.2312/sr.20241146

1. Introduction

Displacement maps provide intuitive means to define mesogeometric details over shapes. This ease of authoring makes them an invaluable tool for both direct rendering and generating auxiliary data from the height field to enhance visuals, such as normal and occlusion maps. In this paper, we consider the robust direct rendering of heightmaps in the high-performance, high-quality setting. We address robustness issues in existing cone map representations that rely on interpolated cones. Additionally, we investigate an observation made by Baboud et al. [BES12] regarding an inherent problem in Policarpo et al.'s relaxed cone map generation algorithm [PO07] and present a correction to it.

Recent advances on micro-geometry rendering offer an alternative to per-pixel displacement mapping. Solutions such as Nanite [Epi24] can render scenes of extreme geometric complexity efficiently, enabling the integration of high resolution source meshes directly as real-time assets. This is achieved via a complex system of pre-processing and runtime geometry filtering steps. However, this necessitates the storage of large geometries that in turn calls for various compression schemes. Moreover, the preprocessing is not trivially applicable to arbitrary animating structures. Displacement maps on the other hand are scalar valued textures, significantly reducing the storage requirements of small-scale geometric detail, at the expense of prescribing a heightfield structure on the meso-geometry. Additionally, a number of heightmap rendering techniques require no or minimal data pre-processing, allowing their adaptation for dynamic content.

Per-pixel displacement mapping aims to compute ray-heightfield intersections for every visible pixel, procedurally implemented in a fragment or pixel shader. Robustness is often traded for fixed evaluation costs, and approximate solutions, such as linear search and their variants, are applied in the most demanding performance-oriented applications [SU08]. However, by additional pre-processing, it is possible to retain robustness without overcommitting the performance requirements. One such solution is Dummer's cone step mapping [Dum06] that accelerates intersection computations by empty space skipping. These safe volumes are represented as cones, each of which encloses intersection-free regions over the meso-geometry. It was shown that cone map generation costs can be reduced to real-time requirements conservatively [VB22], and the representation itself can be altered to improve runtime performance [PO07]. A hybrid approach of conservative and relaxed cones was also suggested [CHRS17].

Our contributions are summarized as follows. First, we show that even bilinearly interpolated conservative cone maps [Dum06] are subject to robustness issues. We propose a combination of two improvements to fix this. In Section 3, we modify the cone step mapping algorithm by replacing the original heuristic minimal step hyperparameter with one that moves the ray onto the next cell. In Section 4, we show a cone map post-processing method that alters apertures to accommodate the effect of bilinear interpolation. These two improvements in conjunction allow the bilinearly interpolated cone maps to render even unit impulse-like heightmaps.

Finally, in Section 5, we propose an exact relaxed cone map generation algorithm replacing the approximate solution of Policarpo et al. [PO07]. Although it significantly improves robustness and is faster to generate, it reduces the performance gains of relaxed versus conservative cone maps. Our proposed modifications and algorithms are evaluated in Section 6.

2. Related Work

We consider direct approaches to rendering height fields, that is, methods where explicit ray-field intersections are carried out, dominantly in the fragment shader. Consequently, approximate and explicit techniques, such as direct tessellation of the displaced surface are out of our scope. For an overview on the broader topic, we refer to Szirmay-Kalos et al.'s survey [SU08].

Cone maps, proposed by Dummer [Dum06], transform the input heightmap into a texture that encodes right circular cones at each texel. The apex of every cone is on the displaced surface and its aperture, i.e., the angle between the generatrix lines, is chosen such that the cone interior is disjoint from the displaced surface. In other words, these are maximal unbounding cones that define empty volumes that ray tracing can skip over without missing an intersection. Cone step mapping against such cones usually terminates the ray-surface intersection search outside the volume defined by the displacement. The most notable exception to this is when bilinear interpolation yields a cone that is no longer unbounding, which often causes rendering artifacts. Adding a root refinement phase after cone step mapping may fix some of these issues, however, certain intersections cannot be recovered, as shown in Figure 4. We address the interpolation problem of cone maps in Section 4.

Policarpo and Oliveira [PO07] noted that the cone apertures can be relaxed to improve rendering performance. In theory, the cones are extended such that rays may intersect the displaced surface at most once. Since these cone maps guide the rays inside the displacement volumes, a binary search root-refinement is employed to find the actual ray-heightfield intersection after the cone step trace.

For practical reasons, their proposed algorithm [PO07] only considered rays originating from the top of the displacement bounding box when computing the widest cone aperture. Moreover, the intersection property is verified via ray marching, i.e., by taking constant-sized steps along rays originating from the top of the displacement volume descending onto each texel. The number of steps affects both the correctness and the pre-processing time of relaxed cone map generation, resulting in an approximate solution.

Indeed, Baboud et al. [BES12] highlighted that the generated relaxed cone maps are only conservative for descending rays that entered at the maximum displacement height, i.e., only for the first step of cone step mapping. All other rays can skip over intersections. The resulting visual artifacts manifest as holes on the displaced surfaces, most visible at locations with large height variations. In contrast, our relaxed field generation algorithm, detailed in Section 5, yields exact cone apertures compared to the overestimations of the original method [PO07]. Consequently, the performance advantage of correct relaxed cone maps is not as prominent compared to the original cone maps. This poses a performance-quality trade-off.

Applications use bilinear interpolation to expand both conservative and relaxed discrete cone maps into continuous functions.

Algorithm 1 Overstep-corrected cell-max cone map tracing

input: $\boldsymbol{p} \in [0,1]^3, p_z = 1$ start position at top of heightmap,
$\boldsymbol{v} \in \mathbb{R}^3, \|\boldsymbol{v}\| = 1, v_z < 0$ downward tracing direction,
$H, C : [0,1]^2 \to [0,1]$ heightmap and cone map textures
output: $0 < t$ heightmap intersection distance along $\boldsymbol{p} + t\boldsymbol{v}$ ray
$t \coloneqq 0,\ i \coloneqq 0$
repeat
  $\boldsymbol{q} \coloneqq \boldsymbol{p} + \boldsymbol{v} \cdot t$ ▷ In heightmap volume $\boldsymbol{q} \in [0,1]^3$
  $h \coloneqq H(\boldsymbol{q}_{xy}),\ \tan\alpha \coloneqq C(\boldsymbol{q}_{xy})$ ▷ A single texture read
  $\Delta t_{cone} \coloneqq \tan\alpha \cdot \dfrac{q_z - h}{\|\boldsymbol{v}_{xy}\| - \tan\alpha \cdot v_z}$ ▷ Based on [Dum06]
  $\Delta t_{cell} \coloneqq \min\left(d_{cell}(q_x, v_x, N), d_{cell}(q_y, v_y, M)\right)$ ▷ Equation (1)
  $\Delta t \coloneqq \max\left(\Delta t_{cone}, \Delta t_{cell}\right)$ ▷ Cone step or to cell border
  $t \coloneqq t + \Delta t,\ i \coloneqq i + 1$ ▷ Take step along ray
until $1 + v_z \cdot (t - \Delta t) \le h$ **or** $i \ge i_{\max}$
return $t - \Delta t$

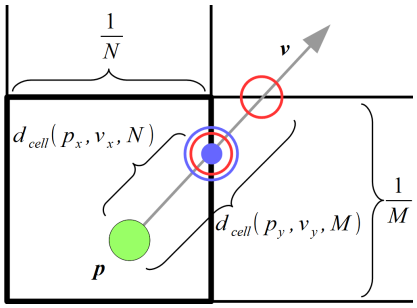


Figure 2: *Starting from* $\boldsymbol{p}$*, Equation* (1) *returns the distance to the red intersection points between the ray and the X and Y gridlines. Algorithm 1 selects the distance to the blue intersection.*

However, previous works did not consider the effect of such interpolation on the correctness of the inferred cones. Interpolating an otherwise conservative cone map can cause a tracing method to miss two consecutive intersections. Even root refinement is insufficient to address this issue in its entirety. However, our cone aperture post-processing strategy allows these methods to robustly trace bilinearly interpolated cone maps.

Cone map generation, regardless of relaxation, is a computationally intensive process because it involves iterating over every possible pair of texels. Bán and Valasek addressed this aspect [VB22] by proposing a conservative, mipmap-based approach to cone aperture computations. However, this method is also susceptible to bilinear interpolation issues should the cone aperture minimizing height value lie on the actual MIP region corner. Our novel approach corrects these interpolation issues and any tracing artifacts due to the global step size epsilon.

Tevs et al. [TIS08] explored alternative acceleration structures for per-pixel displacement mapping. They pre-computed a maximum mipmap structure, where each level stores the height maximums from covered lower levels. These MIP levels allowed skipping empty spaces, starting from the lowest resolution 1×1 level. The method greedily attempts to read from higher levels during a heuristic traversal of this acceleration structure. If a cell of potential intersection is found, an analytic ray-bilinear patch intersection computation is carried out. Due to their choice of representation, they had to rely on 3D textures for this acceleration structure instead of 2D mipmapped textures.

Drobot [Dro18] published a similar method. The main idea of using maximum mipmaps is the same; however, the author's formulation was suitable for conventional mipmap textures. In addition, a simple linear refinement found the final intersection as opposed to a costly analytical solution.

3. Cone step mapping

Let us consider a heightmap and a cone map, both with bilinear filtering. The associated locations of the stored heightmap values are assumed to be in the center of their texel. These centers define a secondary cell grid, each of which contains a bilinear surface patch.

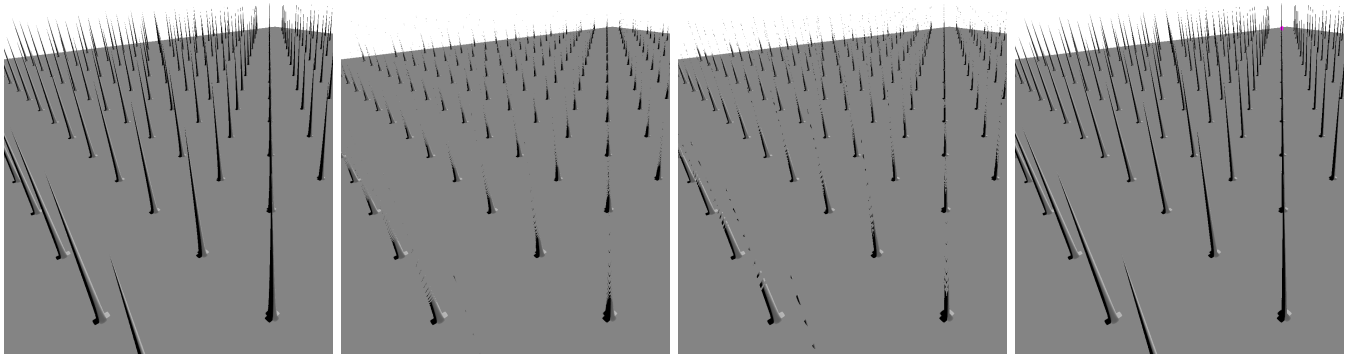
The rendering algorithm proposed by Dummer [Dum06] limits the extent of the smallest step the algorithm takes with a minimum feature size parameter. This is usually set to the size of a single texel. Assuming that we always get a correct unbounding cone and consistently step to the exact intersection of the cone and our ray, the trace slows down as the ray closes in on the ray-surface intersection. Depending on the approaching angle and the surface geometry, this iteration might never reach the intersection point, so in practice, either a distance threshold termination condition or a small minimum trace step size is used. When a cone tracing method with a minimum step size oversteps an intersection, it might end up below the surface, and hence a root is detected. Often, the trace oversteps two intersections at once, resurfacing above the heightmap producing visible rendering artifacts. These artifacts may occur with any minimal step setting, as the spike-like formations depicted in Figure 3 can be arbitrarily thin.

We propose an adaptive correction to this minimal stepsize for bilinearly interpolated heightmaps. Our adaptive minimal step is the distance to the next bilinear cell border depicted in Figure 2. This step ensures that the ray stops under the surface and thus it cannot overstep due to the resolution limit, as long as the current cone is correct. For a ray $\boldsymbol{p} + \boldsymbol{v} \cdot t$ in texture space $(\boldsymbol{p}, \boldsymbol{v} \in [0,1]^2)$, let the $d_{cell} : [0,1] \times [0,1] \times \mathbb{N} \to \mathbb{R}$ function compute the step size towards the closest X and Y gridlines. Thus,

$$d_{cell}(p_i, v_i, n) = \left(\left\lfloor p_i \cdot n + \frac{1}{2} \right\rfloor + \frac{1}{2} \operatorname{sgn}(v_i) \right) \frac{1}{v_i n} - \frac{p_i}{v_i}, \qquad (1)$$

where p_i, v_i are the X and Y coordinates of $\boldsymbol{p}$ and $\boldsymbol{v}$, and n is either the horizontal or vertical resolution of the cell with $\frac{1}{n}$ being the extent of the cell along the appropriate dimension.

Note that this correction may still omit features that are smaller than the ray span within the texel but it proved to be sufficient in our empirical tests. However, it relies on the presence of conservative cone maps, which does not hold under bilinear interpolation, as discussed next.



(a) *Ground truth* **(b)** *Cone step mapping [Dum06]* **(c)** *Relaxed cone map [PO07]* **(d)** *Our corrected cone maps*

Figure 3: *Comparing ground truth (10000 linear search steps) to existing and our proposed cone step mapping algorithms with 200 steps. Our method uses post-processed cone maps with cell-max cone step mapping. Magenta pixels denote unconverged rays.*

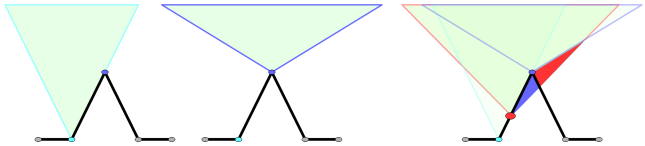


Figure 4: *Interpolation of two unbounding cones can yield a cone that intersects the surface as this 1D counterexample demonstrates. Tracing into the red regions may cause an unrecoverable overstep and rendering artifacts.*

4. Bilinearly interpolated cone maps

Unbounding cones are not closed under interpolation; that is, interpolating the cone parameters between texels may yield a cone that intersects the surface as illustrated by Figure 4. Even though the cone apex will be interpolated correctly, the cone opening angle will not, and this will cause the tracing algorithm to overstep by large distances causing rendering artifacts. Figure 3 showcases the artifacts due to the oversized interpolated cone opening angles even with our stepsize-corrected algorithm from Section 3.

Our second contribution offers a conservative solution by post-processing the cone map before tracing it. We replace each opening angle C_{ij} texel of the cone map by the minimum of its 3×3 neighborhood, that is

$$C_{ij} = \min_{k,l \in \{-1,0,1\}} \{C_{i+k.j+l}\} \,. \tag{2}$$

Practically, this means that in Figure 4, the opening angle of the top cone becomes that of the bottom, meaning that all interpolated cones at most touch the surface. The tracing step sizes will be reduced above such peaks, but now, no interpolation error can occur during tracing. This is because, in each cell, we take the most conservative cone of the four and translate the apex according to the bilinear interpolation. This cone cannot intersect the surface, as it gets further away from touching the surface, except when interpolating along an edge. In that case, the cone could only intersect a nonlinear surface feature between texels which is not possible, as bilinear interpolation along an edge is linear interpolation.

This cone map post-processing step can be applied to relaxed cone maps as well. However, there is an inherent problem in the most commonly used algorithm for relaxed cone map generation that we address next.

5. Relaxed Cone Map Generation

Relaxed cone maps offer fast ray-heightmap intersections for descending rays. It achieves this by overrelaxing the cone opening angles such that the tracing algorithm may overstep, but at most one intersection, that is, it cannot pass through the interior of the heightmap. This means even though the relaxed cones intersect the surface, the first intersection can be identified because the trace cannot miss two roots in a single step. Even though the original relaxed cone map generation method [PO07] is slow due to being a brute force approach, the resulting cone map is only approximate. Even a descending ray might skip over two intersections with the surface within a relaxed cone, as it was previously observed [BES12].

In our third contribution, we seek to improve the efficiency of the relaxed cone map generation step allowing for high-resolution but fast-to-render heightmaps. Observe that for an $H : [0,1]^2 \to [0,1]$ differentiable heightmap and a given $\boldsymbol{u} \in [0,1]^2$ texture coordinate, we only have to consider the set of limiting points $R(\boldsymbol{u}) \subseteq [0,1]^2$ such that

$$R(\boldsymbol{u}) = \left\{ \boldsymbol{v} \in [0,1]^2 \;\middle|\; (\boldsymbol{v}-\boldsymbol{u})^\top \cdot \nabla H(\boldsymbol{v}) < 0 \right\}, \tag{3}$$

where $(\boldsymbol{v}-\boldsymbol{u})^\top \cdot \nabla H(\boldsymbol{v})$ is the directional derivative of the heightmap. Thus, from a given point $\boldsymbol{u}$, the limiting point set $R(\boldsymbol{u})$ consists of texture coordinates where the heightmap is descending, viewed from point $\boldsymbol{u}$ as depicted in Figure 5. Then, the cone map function $C : [0,1]^2 \to \mathbb{R}$ is the reciprocal of the maximum elevation change between $\boldsymbol{u} \in [0,1]^2$ and $\boldsymbol{v} \in R(\boldsymbol{u}) \subseteq [0,1]^2$ points:

$$C(\boldsymbol{u}) = \left(\max_{\boldsymbol{v} \in R(\boldsymbol{u})} \frac{H(\boldsymbol{v}) - H(\boldsymbol{u})}{\|\boldsymbol{v}-\boldsymbol{u}\|} \right)^{-1} . \tag{4}$$

This defines a cone centered at $\boldsymbol{u} \in [0,1]^2$ with height $H(\boldsymbol{u}) \in [0,1]$ and half opening angle $\alpha = \arctan(C(\boldsymbol{u}))$. Specifically, this relaxed

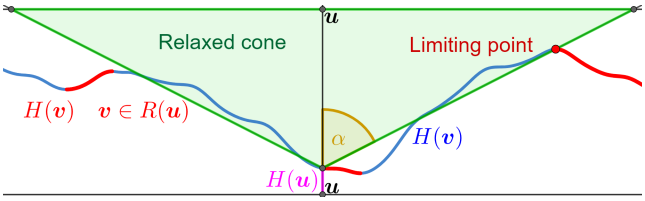


Figure 5: *A 1D slice of the heightmap (blue curve) with the relaxed cone (green triangle). From the cone apex, the cone angle* α *is the largest such that the cone still excludes the red backfacing terrain.*

Algorithm 2 Corrected relaxed cone map generation

input: $H \in [0,1]^{N\times M}$ heightmap image
output: $C \in [0,1]^{N\times M}$ cone map image
$C_{ij} \coloneqq 1 \qquad \forall (i,j) \in \{1..N\} \times \{1..M\}$
for $\forall (i,k,j,l) \in \{1..N\}^2 \times \{1..M\}^2$ **do** ▷ $O(N^2M^2)$
  if $H_{kl} - H_{ij} \leq 0$ **then** ▷ C_{ij} cannot be negative since local
    continue ▷ derivative is negative in a direction.
  end if
  $h_{00} \coloneqq H_{k,l}$ $h_{10} \coloneqq H_{k+\mathrm{sgn}(k-i),l}$
  $h_{01} \coloneqq H_{k,l+\mathrm{sgn}(l-j)}$ $h_{11} \coloneqq H_{k+\mathrm{sgn}(k-i),l+\mathrm{sgn}(l-j)}$
  ▷ If (k,l) is limiting, update cone map:
  if $h_{00} > h_{10}$ **or** $h_{00} > h_{01}$ **or** $h_{10} > h_{11}$ **or** $h_{01} > h_{11}$ **then**
    $C_{ij} \coloneqq \min\left(C_{ij}, \frac{\sqrt{(i-k)^2+(j-l)^2}}{H_{kl}-H_{ij}}\right)$
  end if
end for

cone interior is defined as

$$\left\{\boldsymbol{x} \in [0,1]^3 \;\middle|\; \left\|\boldsymbol{x}_{xy} - \boldsymbol{u}\right\| > \left(x_z - H(\boldsymbol{u})\right) C(\boldsymbol{u})\right\}. \tag{5}$$

This cone ensures that any downward ray $\boldsymbol{p} + t\boldsymbol{v}$ $(v_z < 0 < t)$ starting from above the cone apex, that is, $\boldsymbol{p}_{xy} = \boldsymbol{u}$, $p_z > H(\boldsymbol{u})$, intersects the heightmap at most once within this cone.

However, the usual heightmap texture with bilinear interpolation, $H : [0,1]^2 \to [0,1]$, is not differentiable. Thus, we consider every heightmap texture cell and determine if any point within the cell has a negative directional derivative from our query point. This leads to an $O(N^2M^2)$ method in Algorithm 2, where we operate with range-restricted raw images $H, C \in [0,1]^{N\times M}$. Algorithm 2 calculates C_{ij} by iterating over all bilinear cells considering its closest corner with height H_{kl}. Thus, at the opposing corner of this cell, the height value is $H_{k+\mathrm{sgn}(k-i),l+\mathrm{sgn}(l-j)}$. Due to bilinearity, the u and v derivatives no longer depend on the respective variables, as such, a cell at (k,l) contains a limiting point exactly when any side of the cell realizes a negative height difference viewed from the cone apex (i,j).

We can improve upon the generation time of Algorithm 2 by changing how we loop over each texel pair. Since each C_{ij} cone angle monotonically decreases throughout the algorithm, we can enumerate the cells starting near the query point (i,j) and increase the distance in the $\left\|(i,j)-(k,l)\right\|_\infty$ norm monotonically as we iterate over every other possible (k,l) cell. This means we can terminate our search early if the next square of texels falls under the current cone even with the maximum height of 1, assuming increasingly larger squares around (i,j). This change does not alter the worst-case performance but, in practice, significantly speeds up the generation because the achievable minimum value is usually found much earlier than iterating through all texels.

Moreover, we can parallelize Algorithm 2 for the GPU by calculating each C_{ij} in a separate thread. The second loop may be called in batches for larger images, and later batches can be omitted to obtain an approximate result faster. This method is trivially adaptable to repeating heightmaps too.

	[Dro18]	[TIS08]	[Dum06]	[VB22]	[PO07]	Ours
Base	0.1	0.08	380	0.35	592	11.29
Corr.	-	-	381	0.42	592	11.35

Table 1: *Generation times of constructing the acceleration maps for a* 512^2 *heightmap in milliseconds. The row* Base *refers to the original generation algorithms.* Corr. *refers to the bilinear interpolation correction method for cone maps presented in Section 4.*

6. Test Results

We implemented the proposed and the baseline methods in the NVIDIA Falcor [KCK*22] framework using the DirectX 12 backend. All measurements were taken on an AMD RX 5700 at FullHD resolution. Table 2 shows render times in milliseconds. All cone maps were 512×512, binary16 floating point per channel textures. All generation and rendering algorithms run on the GPU. Our source code can be downloaded from `https://github.com/Bundas102/robust-cone-map`.

Relative frame times wrt. approximate relaxed cone maps increased significantly when applying both of our bilinear interpolation-aware corrections. However, these are necessary to make cone step mapping capable of robustly handling impulse-like features. Moreover, all proposed fixes retain a significant improvement over brute-force ray marching. For example, 200 iterations of the linear search took 1.88ms in the test scene for Table 2. Also note that relaxation is not necessarily possible in general, e.g., in the presence of single-texel thin features; hence corrected conservative and corrected relaxed cone map render times are close.

In comparison, we implemented QDM [Dro18] and maximum mipmaps [TIS08] as a robust alternative to rendering height fields. We omitted Baboud et al.'s solution [BES12], as it was significantly outperformed by relaxed cone maps according to their measurements. Rendering the same scene as in Table 2, we measured QDM with linear refinement (32 steps – 1.84ms, 200 steps – 2.31ms) and maximum mipmaps without refinement (32 steps – 1.89ms, 200 steps – 2.83ms). QDM offers better performance than maximum mipmaps, however, even conservative, fully corrected and cell-max traced cone maps are 2.5 times faster at 200 iterations than QDM.

In terms of pre-processing times, QDM and maximum mipmap are orders of magnitudes faster than the cone map generation algorithms [Dum06; PO07] and our proposed Algorithm 2. The pre-processing correction step for bilinear filtering for a 512^2 map was less than 0.1ms; consequently, it is marginal in the former cases,

Iter. cap: 32		Cone maps			Quick cone maps			Approx. relaxed cone maps			Our relaxed cones (Alg. 2)		
Trace	Ref.	[Dum06]	Corr.	R. cost	[VB22]	Corr.	R. cost	[PO07]	Corr.	R. cost	Original	Corr.	R. cost
Original trace	Lin.	0.59	0.61	3%	0.63	0.65	3%	**0.51**	0.52	2%	0.58	0.61	5%
Cell-max (ours)		0.71	0.77	8%	0.84	0.90	7%	**0.56**	0.57	2%	0.70	0.75	7%
Algorithm cost		20%	26%	31%	33%	38%	43%	10%	10%	12%	21%	23%	29%
Original trace	Bin.	0.66	0.69	5%	0.70	0.73	4%	**0.57**	0.58	2%	0.65	0.68	5%
Cell-max (ours)		0.76	0.82	8%	0.89	0.94	6%	**0.60**	0.62	3%	0.75	0.80	7%
Algorithm cost		15%	19%	24%	27%	29%	34%	5%	7%	9%	15%	18%	23%
Iter. cap: 200		Cone maps			Quick cone maps			Approx. relaxed cone maps			Our relaxed cones (Alg. 2)		
Trace	Ref.	[Dum06]	Corr.	R. cost	[VB22]	Corr.	R. cost	[PO07]	Corr.	R. cost	Original	Corr.	R. cost
Original trace	Lin.	0.74	0.81	9%	0.92	0.98	7%	**0.57**	0.59	4%	0.72	0.78	8%
Cell-max (ours)		0.79	0.87	10%	1.09	1.24	14%	**0.58**	0.60	3%	0.77	0.85	10%
Algorithm cost		7%	7%	18%	18%	27%	35%	2%	2%	5%	7%	9%	18%
Original trace	Bin.	0.79	0.85	8%	0.98	1.04	6%	**0.62**	0.64	3%	0.78	0.84	8%
Cell-max (ours)		0.84	0.92	10%	1.10	1.19	8%	**0.62**	0.64	3%	0.82	0.89	9%
Algorithm cost		6%	8%	16%	12%	14%	21%	0%	0%	3%	5%	6%	14%

Table 2: *Performance measurements in milliseconds taken on an AMD RX 5700 at FullHD resolution. The scene and camera configuration are shown in Figure 1. The* Ref. *column denotes the root refinement method: either a single secant (*Lin.*) or 7 binary search (*Bin.*) steps. The columns* R. cost *list the relative price of applying our bilinear interpolation correction to the generated cone maps. Thus, percentages show the relative increase of render times on the corrected, more conservative cone map. The* Algorithm cost *rows enumerate the relative performance cost of using our proposed cell-max tracing algorithm of Algorithm 1 in comparison to the respective other cone step mapping algorithm [Dum06; PO07]. Their intersections show the overall relative increase of frame times when applying both our cone map representation and cone step tracing robustness improvements to infer conservative cone maps with bilinear interpolation. Note that even though the approximate relaxed cone maps [PO07] are of the highest performance, rendering with them results in artifacts, as in Figure 3. In comparison, the linear search with 32 and 200 steps from the same configuration takes 0.56 ms and 1.88 ms. However, 32 steps result in severe visual artifacts.*

whereas it incurs a 20% additional runtime cost on quick cone map generation [VB22]. The generation times for constructing 512^2 acceleration maps are presented in Table 1.

Our results are mainly focused on artifacts which occur around thin features on heightmaps. This is best visible in Figure 3 but it can be seen on more practical heightmaps, such as in Figures 1 and 6, around silhouettes.

7. Conclusions

This paper presented three modifications to cone step mapping, each improving the robustness of these methods by removing visual artifacts. Over-relaxed cone maps [PO07] offer a performance advantage at the expense of correctness. If visual quality is preferred over the 2-10% additional render time costs, our algorithms present a significantly higher quality alternative to traditional conservative and relaxed cone maps. The advantages of our method are especially present when the surface contains high-frequency details and sharp features or is viewed at shallow angles. However, for dynamically changing heightmaps QDM [Dro18] and maximum mipmaps [TIS08] offer superior generation times, at increased rendering cost compared to our corrected cone map methods.

As such, the most optimal per-pixel displacement method depends on the nature of the application. Performance-centric settings that are permissive about rendering artifacts are still more likely to choose a simple linear search with limited step counts. Similarly, QDM is more suitable for dynamically changing height fields, if quality compromises are not acceptable. In high-quality, high-performance applications with static heightmaps, however, the bilinearly and conservativeness corrected relaxed cone maps proposed here offer a better alternative. As future work, a combination of the quick cone map generation method [VB22] with our relaxed cone map improvements would provide an alternative for dynamically changing heightmaps as well.

Acknowledgement

Supported by the ÚNKP-23-4 New National Excellence Program of the Ministry for Culture and Innovation from the source of the National Research, Development and Innovation Fund. We thank Visual Concepts for providing the AMD GPU used in the tests.

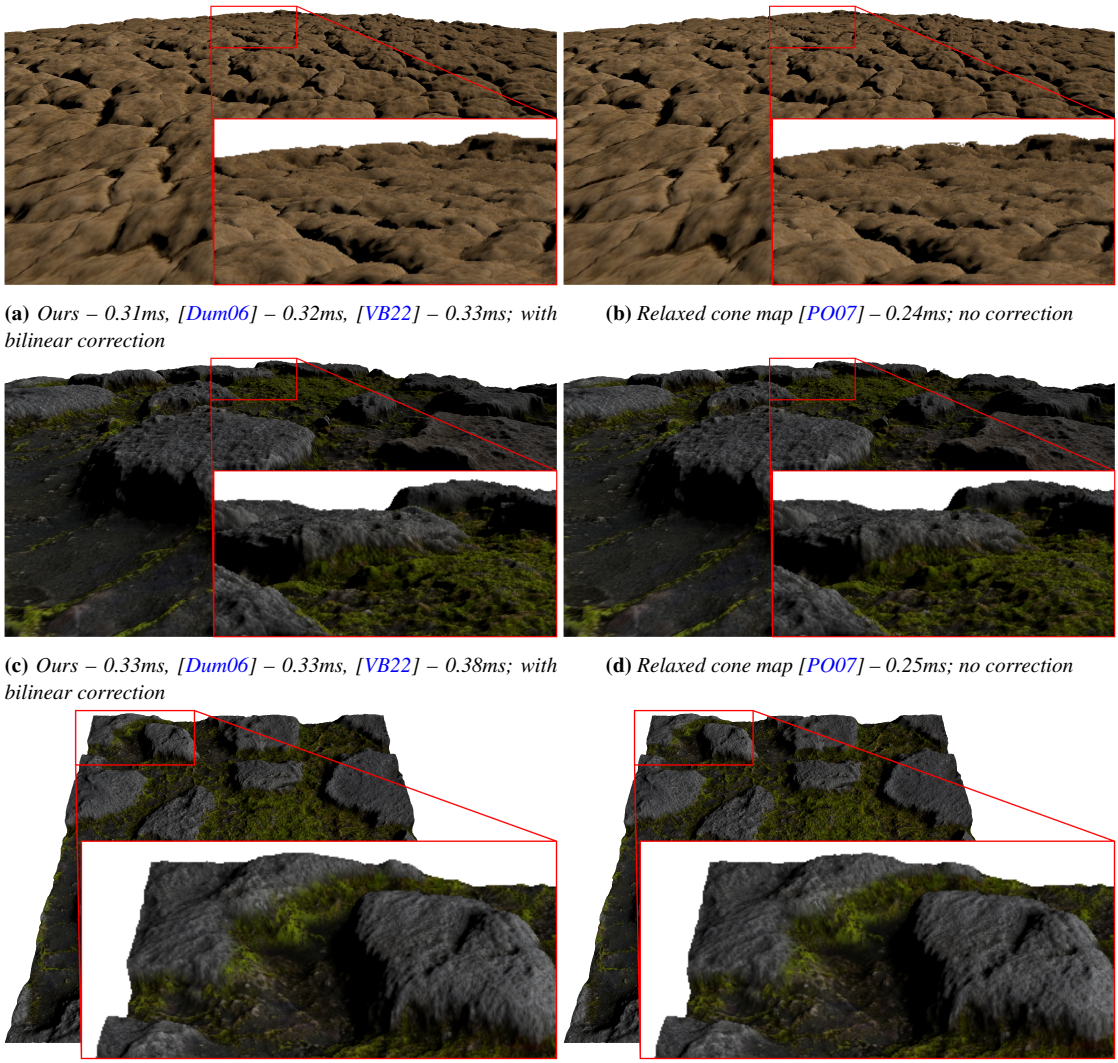
References

[BES12] BABOUD, L., EISEMANN, E., and SEIDEL, H-P. "Precomputed Safety Shapes for Efficient and Accurate Height-Field Rendering". *IEEE Transactions on Visualization and Computer Graphics* 18.11 (Nov. 2012), 1811–1823. DOI: 10.1109/TVCG.2011.281 2, 4, 5.

[CHRS17] CHAHDI, ADNANE OUAZZANI, HALLI, AKRAM, RAGRAGUI, ANOUAR, and SATORI, KHALID. "Per-pixel displacement mapping using hybrid cone approach". *2017 International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*. Fez, Morocco: IEEE, May 2017, 1–4. ISBN: 978-1-5386-0551-6. DOI: 10.1109/ATSIP.2017.8075577 2.

[Dro18] DROBOT, MICHAŁ. "Quadtree Displacement Mapping with Height Blending". *GPU Pro 360 Guide to Rendering*. 1st. A K Peters/CRC Press, 2018, 1–32. ISBN: 978-1-351-26152-4 3, 5, 6.

[Dum06] DUMMER, JONATHAN. "Cone Step Mapping: An Iterative Ray-Heightfield Intersection Algorithm". 2006 1–7.



(a) *Ours – 0.31ms, [Dum06] – 0.32ms, [VB22] – 0.33ms; with bilinear correction*

(b) *Relaxed cone map [PO07] – 0.24ms; no correction*

(c) *Ours – 0.33ms, [Dum06] – 0.33ms, [VB22] – 0.38ms; with bilinear correction*

(d) *Relaxed cone map [PO07] – 0.25ms; no correction*

(e) *Ours – 0.18ms, [Dum06] – 0.18ms, [VB22] – 0.20ms; with bilinear correction*

(f) *Relaxed cone map [PO07] – 0.14ms; no correction*

Figure 6: *Comparison of our corrected cone maps to the original relaxed cone maps. The left pictures are rendered using our method (relaxed cone map with bilinear correction) and a single secant refinement step. Aside form a few pixels, the original cone maps [Dum06] and quick cone maps [VB22] are equivalent to this. The right pictures show renderings using the original relaxed cone maps [PO07] with 7 binary refinement steps. The difference is mainly visible around silhouettes of peaks.*

[Epi24] EPIC GAMES. *Nanite Virtualized Geometry*. 2024. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/nanite-virtualized-geometry-in-unreal-engine 2.

[KCK*22] KALLWEIT, SIMON, CLARBERG, PETRIK, KOLB, CRAIG, et al. *The Falcor Rendering Framework*. Aug. 2022. URL: https://github.com/NVIDIAGameWorks/Falcor 5.

[PO07] POLICARPO, FABIO and OLIVEIRA, MANUEL M. "Relaxed Cone Stepping for Relief Mapping". *GPU Gems 3*. 2007. ISBN: 0-321-51526-9 1, 2, 4–7.

[SU08] SZIRMAY-KALOS, LÁSZLÓ and UMENHOFFER, TAMÁS. "Displacement Mapping on the GPU - State of the Art". *Computer Graphics Forum* 27.6 (Sept. 2008), 1567–1592. ISSN: 01677055, 14678659. DOI: 10.1111/j.1467-8659.2007.01108.x 2.

[TIS08] TEVS, ART, IHRKE, IVO, and SEIDEL, HANS-PETER. "Maximum mipmaps for fast, accurate, and scalable dynamic height field rendering". *Proceedings of the 2008 symposium on Interactive 3D graphics and games*. I3D '08. New York, NY, USA: Association for Computing Machinery, Feb. 2008, 183–190. ISBN: 978-1-59593-983-8. DOI: 10.1145/1342250.1342279 3, 5, 6.

[VB22] VALASEK, GÁBOR and BÁN, RÓBERT. "Quick Cone Map Generation on the GPU". *Eurographics 2022 - Short Papers*. Ed. by PELECHANO, NURIA and VANDERHAEGHE, DAVID. ISSN: 1017-4656. The Eurographics Association, 2022, 4 pages. ISBN: 978-3-03868-169-4. DOI: 10.2312/egs.20221021 2, 3, 5–7.