High-Quality Shadows for Streaming Terrain Rendering

Matthäus G. Chajdas, Florian Reichl, Christian Dick, and Rüdiger Westermann

Technische Universität München

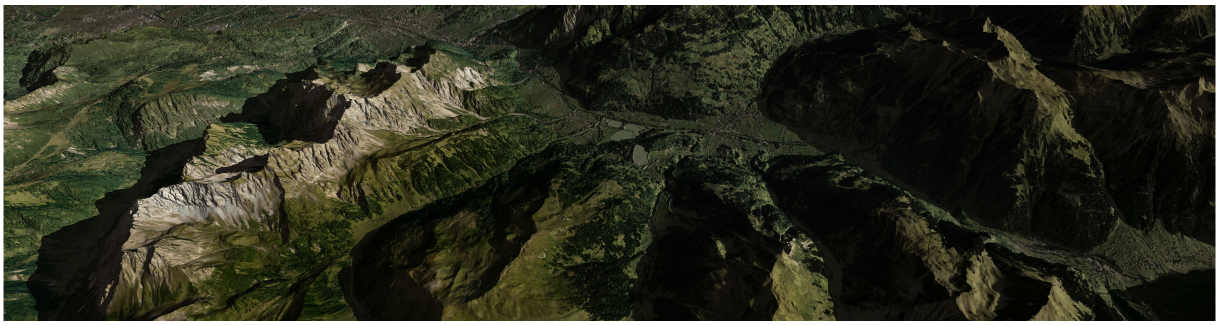

Figure 1: *A 76 gigasample terrain field (`Vorarlberg`) at dawn. Shadows are simulated using pre-computed far-shadows at terrain resolution and ray-traced near-shadows.*

Abstract

Rendering of large, detailed 3D terrains on commodity hardware has become possible through the use of ray-casting, data caching and prefetching. Adding dynamic shadows as they appear during a day-night cycle remais a challenge however, because shadow rendering requires access to the entire terrain, invalidating data streaming strategies.

In this work we present a novel, practicable shadow rendering approach which distinguishes between near- and precomputed far-shadows to significantly reduce data access and runtime costs. While near-shadows are ray-traced using the current cache content, far-shadows are precomputed and stored in a very compact format requiring approximately 3 bit per height-map sample for an entire day-night cycle.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

1. Introduction

Shadows are an important visual cue to enhance the realism of digital terrain fields. Unfortunately, incorporating shadows into terrain rendering is very difficult due to the high resolution of available terrain models. This resolution prohibits the use of precomputed shadow maps, especially when day-night cycles are to be simulated. Moreover, a shadow map—or ray-traced shadows—cannot be easily generated at run-time, because it potentially requires access to the entire data set and, thus, invalidates any view-dependent caching and prefetching strategy.

Especially at dusk and dawn, shadows can extend over several kilometers, making it necessary to keep large parts of the data set in memory. For a directional light like the sun, an additional difficulty arises from the fact that no level-of-detail simplification can be used at all, as there is no foreshortening.

In our work, we describe a practical, yet accurate solution for shadow rendering in high resolution terrain fields which can be easily integrated into streaming terrain rendering systems and requires only modest offline storage. The key idea is to split shadows into near- and far-shadows; the former can be efficiently computed at run-time using the ter-

DOI: 10.2312/egsh.20151014

rain data which is cached for the current view, and the latter can be precomputed and compressed effectively due to its low frequency content. We demonstrate that all far-shadow maps for a full day-night cycle can be compressed to a size which is comparable to the size of the color data. Combined with realtime ray-tracing of near-shadows, our technique adds only a modest computational and memory overhead and enables scalable terrain rendering with shadows.

2. Previous work

Scalable terrain rendering requires compact storage of both height-map and color data, efficient ray-casting and a streaming system. These components have been brought together for the first time in [DKW09]. In their work, a complete terrain rendering system built around a ray-caster is presented. Their system relies heavily on streaming to minimize memory usage. To this end, the terrain is subdivided using a quad-tree and individual tiles are loaded at exactly the right resolution. To ensure high-quality rendering, the level-of-detail is selected such that the absolute pixel error in screen space is at most one pixel. This requires the rendering of sub-pixel geometry, which is handled by the ray-caster. Shadows have not been covered in this work at all.

Shadows for high-resolution height-field rendering have seen nearly no attention in the research community. Two works which are focused on real-time shadows of dynamic height-fields are [SN08] and [TW10]. Unfortunately, both approaches are only suitable for low-resolution terrains, as they require global access to the height-field data. This makes them incompatible with a streaming system, and for the high-resolution terrains we use in this work, would result in excessive memory usage.

Recently, [SKOA14] introduced an approach for rendering of extremely high resolution shadow maps by storing them in a compact tree representation. Even though they support shadow maps with very high resolutions, this technique is not suited for high-resolution terrain data. To achieve shadows without aliasing per height map sample, a projected shadow map has to exceed the resolution of the original terrain, resulting in extremely large shadow maps. Moreover, these large shadow maps cannot be easily streamed depending on the currently visible terrain extents.

3. Algorithm

The core idea underlying our approach is to use a binary shadow map at the resolution of the terrain field. It stores a bit indicating "shadowed" or "lit" for every height-field sample. Such a shadow map can be easily generated in an offline process, for instance, by ray-tracing directly on the height-field. Unfortunately, for high-resolution elevation maps, where the geometric resolution is below 1 meter, the resulting shadows are extremely noisy and cannot be compressed efficiently (see also Figure 2). In general,

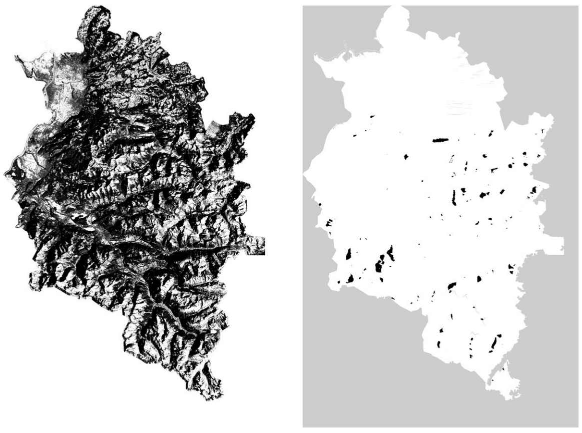
Figure 2: *All shadows and far-shadows only. The left image contains both near- and far-shadows, and cannot be compressed well. The right image contains only the far-shadows, and allows for a much higher compression.*

the noise stems from short-distance shadows which are cast by rather small geometric features like trees and buildings. Long-distance shadows which are cast by mountain ridges, on the other hand, tend to be very smooth.

We take advantage of this observation by splitting the shadow map into near- and far-shadows (see Figure 3). Near-shadows can be efficiently computed at run-time, as they only require local access to the height-field, while the low-frequency shadows resulting from far-shadows can be precomputed and compressed effectively.

3.1. Preprocess

In a preprocess, we first generate the binary shadow map for the entire terrain field for every level-of-detail. For every terrain sample we test for far-shadow casters farther away from the sample than a given far-shadow threshold. Shadow casters closer than this threshold will be later resolved at runtime.

The resulting shadow map is first partitioned into pages of size 8192^2. This is followed by a dictionary-based tile encoder and a final LZMA compression [SM10]. Therefore, all pages are further subdivided into 8×8-sized tiles, and each unique 64-bit tile is identified and stored in a dictionary. The shadow map is replaced by an array of indices into this dictionary. Depending on the number of unique tiles, either 16 or 32 bit pointers are used. Compared to the binary input shadow map, this simple compression scheme results in an average compression ratio of slightly below 1:4, as most input images contain less than 2^{16} unique tiles. The described process converts the shadow map into a highly-uniform tile index map and a high-entropy dictionary. We further com-

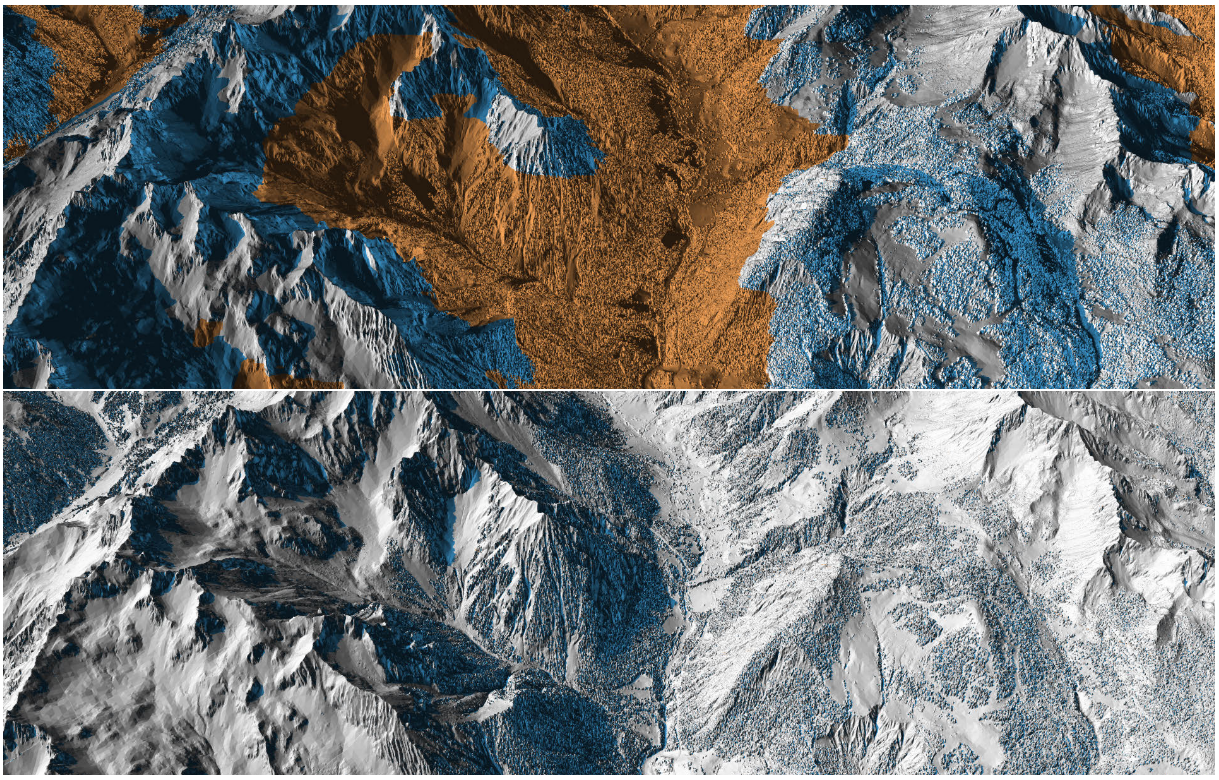

Figure 3: *Near- and far-shadow visualization on the* `Vorarlberg` *data set at dawn (upper image) and noon (lower image). Blue indicates near-shadows which have been ray-traced during the real-time rendering, orange far-shadows which have been precomputed. In the morning, long shadows stored in the far-shadow map dominate the image, while at noon, only near-shadows are present.*

press both separately using standard LZMA compression. Finally, this results in a compression ratio of 1:50.

To allow for simulating shadow over a full day-night cycle, we precompute shadow maps at a discrete set of times over the cycle. However, instead of sampling the sun position at uniform time steps, we use an adaptive sampling of the sun trajectory. The periods close to dusk and dawn are sampled at higher frequency than those around noon, to reduce the "jumping" between individual shadow maps. Completely empty shadow maps, which occur if there are only near-shadows, are omitted. For all data sets, we only create shadow maps between dawn and dusk, which covers approximately 13 hours for each day.

3.2. Runtime

We have integrated our algorithm into an existing terrain rendering system similar to [DKW09]. At run-time, we decompress the precomputed binary shadow map pages required for the current view and upload them to the GPU. To minimize storage we pack the binary shadow map into integers and unpack individual bits on the GPU to retrieve the shadow values. At each pixel, we first check if a far-shadow is already present. If this is not the case, an additional shadow ray is traced from the point under the pixel to check for a near-shadow. The maximum length of the shadow ray is bound by the far-shadow threshold used in the initial construction of the binary shadow map. We chose the far-shadow threshold to be equal to the side length of a single quadtree-tile—512 samples in our system—to limit the additional cache storage requirements to immediate neighbouring tiles.

The near-shadow tracing itself can be easily integrated. Our terrain renderer is already ray-guided, that is, the ray-tracing determines which parts of the terrain are streamed onto the GPU. In this context, casting near-shadow rays requires no modification to the streaming and caching. The main impact of the shadow tracing is reduced performance, due to additional rays, as well as slightly increased memory usage as border tiles have to be kept in memory.

3.3. Interpolation

We do not perform any filtering of the shadow maps during lookup to make transitions between far and near-shadows

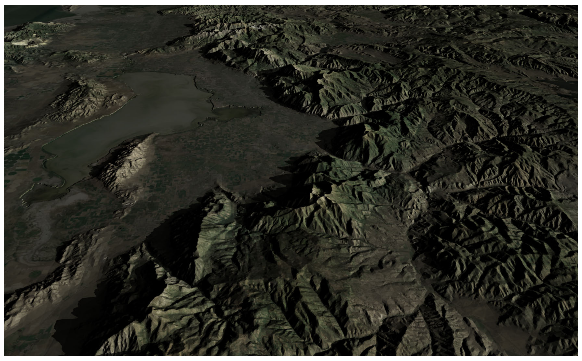

Figure 4: *The* `Utah` *data set with shadows. The data set consists of 69 gigasamples.*

seamless. This does not produce visible artifacts as the level-of-detail selection of the terrain ray-caster is bound to a fixed pixel-to-sample ratio. Optionally, linear interpolation between two shadow maps can be performed to smooth out the transitions between different time steps. In this case, near-shadows are also traced in two precomputed sun directions, and the resulting shadows are blended in the same way. While tracing in a single, interpolated sun direction would completely eliminate the artifacts due to shadow jumps, it would clearly show the shift from near to far-shadows in return.

4. Results

We tested our algorithm on two data sets (see Figure 1 and 4). The compression yields similar results for both the `Vorarlberg` and the `Utah` data set. For `Vorarlberg`, with 76 gigasamples, we achieve 2.77 bits per pixel for the complete day-night cycle consisting of 158 shadow maps. For `Utah`, with 69 gigasamples, we obtain 2.08 bits per pixel, again for a whole cycle with 158 shadow maps. The average compression rates for the far-shadow maps are 0.018/0.013 bits per pixel. For comparison, without the near-/far-shadow split, the shadow maps would compress much worse at 0.45 bits per pixel for `Vorarlberg` and 0.498 for `Utah`.

The rendering performance is halved when real-time shadows are enabled, but due to the high base speed of the terrain rendering, the rendering requires less than 20 ms per frame on a single desktop GPU at 1920×1080 resolution (NVIDIA GTX Titan). The main run-time cost stems from the shadow ray-tracing, which is expensive as it traverses close to the terrain. The shadow map sampling adds only neglible overhead, as it consists of a texture lookup followed by a fast bit extraction.

5. Limitations and Future Work

We have presented a practicable approach to shadow mapping for high-resolution terrain rendering. We see our work as a first step towards fully interactive rendering of dynamic shadows for high-resolution terrain fields.

Currently, there are three main limitations to the algorithm: Level-of-detail transitions, lack of filtering, and popping due to precomputed far-shadows. The first two issues are interrelated. As we compute a separate shadow map for each level of the terrain multiresolution representation, at the same resolution as the terrain height-field, the shadow resolution is bound directly to the terrain. It is necessary in our algorithm to bind the shadow map resolution to the height-field resolution, as this enables us to seamlessly transition between the precomputed and the real-time ray-traced shadows. Switching height-map level-of-detail can easily guarantee sub-pixel error, which cannot be done for the far-shadow maps which may exhibit large changes if for example a tree on a mountain ridge gets removed by level-of-detail.

Because we also do not filter the shadow map, this can lead to popping and aliasing. We expect that a “soft-shadow” filtering on the far-shadows can be used to resolve this issue, but encoding the shadow distance or smooth far-shadows will require additional storage.

Finally, the fixed sampling for one day-night cycle forces us to interpolate between far-shadow maps. This can lead to visible “jumps” if not enough shadow maps are used. In the current work we use only a simple sampling scheme which tries to optimize the expected distance between shadow fronts in consecutive shadow maps. A more sophisticated approach, which takes the actually observed differences between shadow maps into account, could result in superior sampling. Additionally, higher-quality interpolation can be performed using higher-order filters.

Acknowlegments

The authors wish to thank the Landesvermessungsamt Feldkirch, Austria and the State of Utah for providing high resolution geo data.

References

[DKW09] DICK C., KRÜGER J., WESTERMANN R.: GPU ray-casting for scalable terrain rendering. In *Proceedings of Eurographics - Areas Papers* (2009), pp. 43–50. 2, 3

[SKOA14] SINTORN E., KÄMPE V., OLSSON O., ASSARSSON U.: Compact precomputed voxelized shadows. *ACM Transactions on Graphics 33*, 4 (2014). 2

[SM10] SALOMON D., MOTTA G.: *Handbook of Data Compression (5. ed.).* Springer, 2010. 2

[SN08] SNYDER J., NOWROUZEZAHRAI D.: Fast soft self-shadowing on dynamic height fields. In *Eurographics Symposium on Rendering* (2008), pp. 1275–1283. 2

[TW10] TIMONEN V., WESTERHOLM J.: Scalable Height Field Self-Shadowing. *Computer Graphics Forum (Proceedings of Eurographics 2010) 29*, 2 (May 2010), 723–731. 2