Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Multi-resolution Terrain Rendering Using Summed-Area Tables

Shi Li, Chuankun Zheng, Rui Wang*, Yuchi Huo, Wenting Zheng, Hai Lin, Hujun Bao

State Key Lab of CAD&CG, Zhejiang University, HangZhou, 310058, China

ARTICLE INFO

Article history:
Received 2021

Keywords: Terrain Rendering, Summed-Area Tables (SATs), Level of Detail (LOD)

ABSTRACT

Due to the fundamental weaknesses of level-of-detail (LOD) control and rich details in the Geometry Clipmaps, we propose a multi-resolution terrain rendering algorithm that utilizes summed-area tables (SATs) [1] to facilitate the rendering of terrain with better geometric and shading details. First, our algorithm introduces a novel geometric error bound on the screen-based terrain rendering approach that juggles low rendering throughput and better LOD control. Geometric errors are estimated in real-time from SATs, enabling error-bounded geometry clipmap. Second, we utilize Spherical Gaussian (SG) functions to approximate lighting and bidirectional reflection distribution functions (BRDFs), and efficiently calculate outgoing radiance with self-occlusions of the terrain. SATs are utilized to enable the mipmapping of visibility and normal maps. We demonstrate the improvements of our method with experiments on accuracy and efficiency.

1. Introduction

Real-time terrain rendering plays an important role in many applications, such as geographic information systems, virtual reality, 3D games, battlefield simulation, etc. Numerous methods have been proposed to address the issue of large terrain rendering while maintaining rich details. According to the space of refinement hierarchy, these methods can be divided into two categories. The first category, namely Mesh Hierarchy, is based on the object space. It adopts a hierarchy of mesh refinement operations to control LOD for large terrain heightmap. The structure of these hierarchies irregular meshes [2, 3, 4, 5], bin-tree hierarchies [6, 7, 8], bin-tree regions [9, 10, 11] and tiled blocks [12, 13, 14]. With the expansion of hardware capability and GPU pipelines, especially tessellation shader, have been utilized to enhance algorithm[15, 16, 17, 18, 19, 20, 21, 22] performance. All these methods control the LOD on the terrain through split-merge operations [12] or quadtree structure [20] in object space. However, such hierarchies have several shortcomings. For example, the refinement operations must be pre-computed and consume additional memory; and the data structures involve random-access traversals with poor cache-coherence.

While the viewpoint moves, the clipmap levels shift and are only filled with data. It provides considerable advantages. First, it takes constant memory footprint, where all vertices reside in video memory and no explicit vertex I/O at runtime. Second, it has no irregular traversal of geometry structures and no tracking of refinement dependencies, thereby has better rendering throughput and efficiency. Geometry Clipmaps has been widely used in the game engines or virtual globes, such as OGRE engine [24] and OpenGlobe [25]. However, it still has fundamental weakness that the LOD in Geometry Clipmaps [23] is essentially based on the 2D distance to the viewpoint, thus lacking accurate estimation of geometric errors.

In this work, we present a new screen-based terrain rendering

*Corresponding author: Tel.: +86-0571-88206681-430;
e-mail: rwang@cad.zju.edu.cn (Rui Wang)

approach that inherits the advantages of screen and object space approaches. Technically, compared with methods of Mesh Hierarchy, our method resides all vertices data in video memory, and not need to tessellate the grids or track refinement dependencies at runtime. Compared with methods of Geometry Clipmaps, our method dynamically selects a LOD level of heightmap to bound the geometric error using SATs-based filtering.

Moreover, we found SATs do not only facilitate the rendering of terrain geometry but be used for better terrain shading. Realistic terrain shading that usually incorporates lighting, normal maps, visibility maps, etc., brings more challenges to multi-resolution representations and rendering techniques. Quite a few related works [26, 27, 28, 29, 30, 31, 32] have been proposed to filter normal or visibility maps. In this work, we employ Spherical Gaussian(SG) functions to approximate the lighting, visibility and non-diffuse BRDFs. As such, the outgoing radiance can be computed with visibility and normal values fetching values from SATs.

Our major contributions can be summarized as follows:

- A novel screen space terrain rendering method that estimates geometric error bound from SATs (Section 4);
- A novel high-quality terrain shading method composing SG-functions-approximated lighting and BRDFs with precomputed visibility maps to effectively calculate outgoing radiance (Section 5);
- An efficient GPU implementation of terrain rendering using SATs. (Section 6).

2. Related Work

2.1. Terrain Geometry Rendering

Terrain rendering has long been an active topic in computer graphics and geographical sciences. Please refer to this survey [33] for a comprehensive review.

2.1.1. Geometry Representation

The geometry representation can be divided into two categories. The first category adopts a hierarchy of mesh refinement operations to control LOD for large terrain heightmap. The structures of these hierarchies include irregular meshes, bin-tree hierarchies, bin-tree regions, and tiled blocks. Irregular meshes [2, 3, 5] approximated a number of faces with the tracking of mesh adjacencies and refinement operations. Bin-tree hierarchies [6, 7, 8] applied the recursive bisection of right triangles to optimize performance and memory dynamically. Bin-tree regions [9, 10, 11] used a bin-tree structure to define refinement operations. Tiled blocks [12, 13, 14, 16, 20, 21, 34] partition the terrain into square patches and then tessellate the patches into different resolutions.

Rather than defining a world-space hierarchy mesh, Frank Losasso and Hugues Hoppe [23] introduced Geometry Clipmaps, which defines a hierarchy centered the viewer, and this greatly simplifies inter-level continuity in both space and time. Geometry Clipmaps cache nested rectangular extents of the pyramid for rendering in the video memory. As the viewpoint moves, the clipmap levels shift and the terrain data updates toroidally. The LODs purely depend on the 2D distance from the clipmap's center. Clasen and Hege [35], Aleksandar and Dejan [36] further extended the Geometry Clipmaps to spherical terrain and an ellipsoid. With the continuous development of hardware, Dirk and Klaus [37],

Ge et al. [22] used GPU-based ray casting and tessellation shader to optimize Geometry Clipmaps.

2.1.2. LOD Control

LOD control is essential to adjust the terrain tessellation as a function of the view parameters. Viewer distance [16, 23] is the most common metric but it lacks an accurate assessment of screen-space geometric errors. Many related works have been focused on this topic in Mesh Hierarchy. The vertical distance [7, 39] of a removed vertex with respect to its linear interpolation provided by the parent node had been applied to approximate object-space errors. Duchaineau et al. [6] calculate the thickness of a bound wedgie for each triangle to approximate object-space errors, and then the thickness could be used to estimate the maximal screen-space distortion. Lindstrom et al. [40, 41, 8] used the screen-space error metric to determine the removal and inclusion of vertices for a given viewpoint. Kang et al. [20] utilized GPU tessellation shader to subdivide the terrain mesh in real-time, and selected the tessellation factors on the basis of screen-space errors. However, these approaches be directly applied to Geometry Clipmaps due to viewer-centric grids. Therefore we prefer to add the screen-space geometric error metric on the Geometry Clipmaps because we still agree with the authors[23] that Geometry Clipmaps has many advantages Mesh Hierarchy, such as simplicity, optimal rendering throughput, visual continuity, steady rendering.

2.2. Terrain Surface Shading

The common way of terrain shading is to combine texture maps with lighting. Oat and Pedro [42] proposed an approach that considers ambient light. Technically, it precomputes a spherical cap, which stores average, contiguous regions of visibility, and computes how much of their light passes through the spherical cap at runtime. Snyder and Nowr-ouzezahrai [43] precomputed a horizon map for a group of azimuthal directions in the order-4 spherical harmonic (SH) basis. Therefore it enabled soft-shadowed results by the SH triple product of 16D vectors among lighting, visibility, and diffuse reflectance. Given the fact that SH basis functions only capture low-frequency lighting effects and often "ringing" artifacts, high-quality terrain shading with spatially-varying reflectance remains a challenging problem. Chajdas et al. [44] presented a shadow rendering approach that distinguishes between near- and precomputed far-shadows to significantly reduce data access and runtime costs.

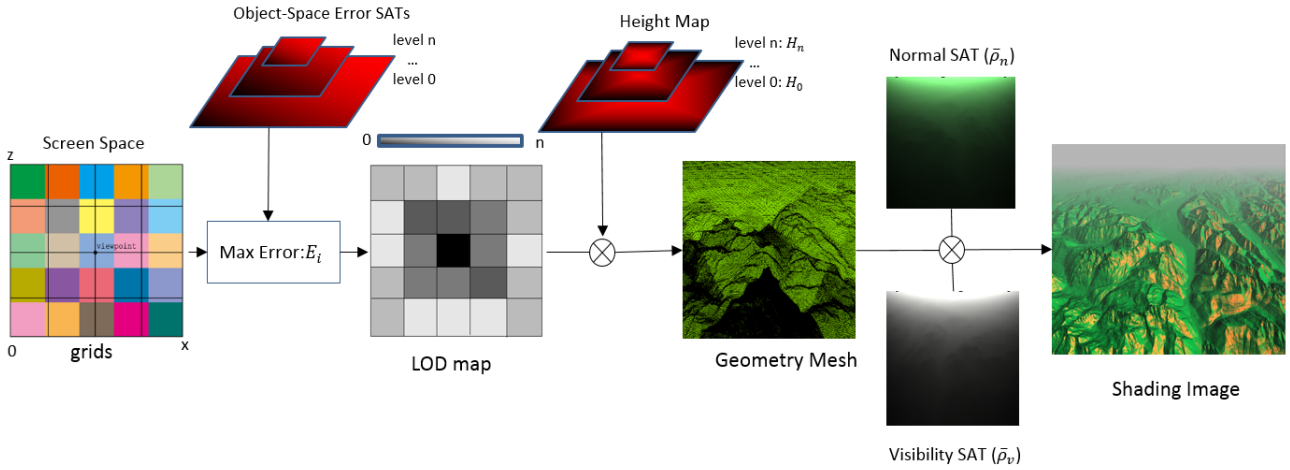


Fig. 1. Our method pipeline for rendering. We first fetch the object-space errors from SATs and calculate the corresponding screen space geometry error bound. Next, we select LOD level for each clipmap's grid using the geometry error bound. Finally, the LOD mesh, normal SATs, and visibility SATs are used to perform the shading efficiently.

2.3. Prefiltering Methods for Surface Shading

Finding nonlinear filtering methods for accurate surface shading has been one of the central issues in computer graphics. Bruneton and Neyret[26] conducted a comprehensive review of this topic. Several approaches [28, 45, 31] introduced new kernel functions to convolute with normal distribution function (NDF). LEAN mapping[30] transferred normal distribution into the slope domain and utilized the shape invariance property of Beckmann distribution to linearly filter the normal map. Then, Eric Heitz et al.[29] investigated the procedural color map associated with the displacement map and slope domain. Later, LEADR [27] generalized the LEAN mapping framework to jointly consider masking and shadowing functions. Recently, SGGX[46] is proposed to consider filtering light transports in voxel data. Xu et al.[32] gave a real-time BRDF without precomputing. Wu et al. [47] extended LEAN and LEDAR to general bi-scale appearance models. However, these methods still focus on filtering materials and are unsuitable for shading of large-scale scenes. We propose a novel method to prefilter terrain surface shading with visibility.

3. Overview

3.1. Problem Definition

In general, algorithms based on Geometry Clipmaps cache the terrain in a set of grids centered on the viewer. These grids represent filtered versions of the terrain at power-of-two resolutions and are stored as vertex buffers in video memory. As the viewpoint updates, the shift and are refilled with filtered heightmap [23]. The LOD control for filtering each grid's heightmap allows a fast exploration of huge terrain data with limited memory.

The mipmap level rendered at each grid should consider view parameters [23] and the content of terrain data to avoid and increase efficiency. However, no previous method jointly considers both factors. Here, we introduce a novel approach to find a tight and efficient bound on the screen-space geometric error to guide the LOD control for each grid and reduce the shading cost. The screen-space geometric error E_i of level i for the grid B_j is defined as follows:

(1)

The general shading of terrain surface calculates the outgoing radiance [48] on point x toward direction ω_o:

$$L_o(x,\omega_o) = \int_{S^2} L_i(x,\omega_i)\rho(x,\omega_i,\omega_o)\langle n(x),\omega_i\rangle d\omega_i, \qquad (2)$$

where S^2 represents the upper hemisphere, $L_i(x,\omega_i)$ denotes the incident radiance from direction ω_i, ρ bidirectional reflectance distribution function (BRDF), $n(x)$ normal at point x, the operator $\langle\cdot,\cdot\rangle$ gives the dot product of two vectors, clamped to zero if it is negative.

Given the footprint Ω around the point x, the antialiased integration [28, 49, 50, 27] can be approximated as follows:

$$L_o(\Omega,\omega_o) = \frac{1}{|\Omega|}\int_\Omega\int_{S^2} L_i(y,\omega_i)\rho(y,\omega_i,\omega_o)V(y,\omega_i) \langle n(y),\omega_i\rangle d\omega_i dy, \qquad (3)$$

where $|\Omega|$ denotes the surface area of Ω and $V(y,\omega_i)$ is the visibility function, which returns 1 if y is not blocked along the direction ω_i and 0 otherwise.

In general, we decompose the integrand of Equation 3 into a lighting term, $L_i(x,\omega_i)$, a BRDF term, $\bar{\rho}_n(\omega_i,\omega_o)$, and a visibility term, $\bar{\rho}_v(\omega_i)$, as follows:

$$\begin{aligned} L_o(\Omega,\omega_o) &= \int_{S^2} L_i(x,\omega_i)\bar{\rho}_n(\omega_i,\omega_o)\bar{\rho}_v(\omega_i)d\omega_i, \\ \bar{\rho}_n(\omega_i,\omega_o) &= \frac{1}{|\Omega|}\int_\Omega \rho(y,\omega_i,\omega_o)\langle n(y),\omega_i\rangle dy, \\ \bar{\rho}_v(\omega_i) &= \frac{1}{|\Omega|}\int_\Omega V(y,\omega_i)dy. \end{aligned} \qquad (4)$$

Notice that y has been removed from the integrand by assuming $L_i(y,\omega) \approx L_i(x,\omega)$ for all $y \in \Omega$ and $\omega \in S^2$ when Ω is small

neighbor around x. Please refer to supplementary materials for more details.

3.2. Algorithm Overview

Figure 1 illustrates the pipeline of our terrain rendering algorithm. Initially, clipmap's grids centered at camera position will be generated. Subsequently, two stages, namely geometry stage and shading stage, will be applied to generate shapes and colors of terrain respectively.

In the geometry stage, each level is chosen by bounding the screen-space geometric error E_i, derived from maximum filtering on SATs object-space geometric errors for each level of heightmap.

In the shading stage, SG functions are utilized to approximate lighting and BRDFs. Further, we precompute visibilities on terrain and use SATs to filter visibility and normal maps. The smoothed visibility ($\bar{\rho}_v$) and normal ($\bar{\rho}_n$) are combined with approximated lighting and BRDFs to facilitate the shading computation of terrain geometry.

In the following subsections 4 and 5, we describe our terrain rendering algorithm based on screen-space geometric error bound and illustrate the shading process.

4. Geometric-Error Bounded Rendering in Screen Space

Geometry Clipmaps [23] proposes to select LOD levels based on distance. It builds a multi-level terrain pyramid, representing nested extents at successive power-of-two resolutions. However, only using the distance to select LOD levels might cause artifacts or reduce efficiency. For example, if smooth meshes are located near the viewpoint, the coarse-level height map is enough to provide visual fidelity with high efficiency; in contrast, distant steep slopes might still cause high screen-space errors and thus requires a fine-level heightmap. Based on these observations, our approach adaptively selects the level of heightmap and number of vertices for each grid according to screen-space geometric errors, rather than the distance.

4.1. Error Bound Calculation

We calculate the screen-space errors by projecting object-space geometry errors to the screen space. However, calculating the screen-space errors at each point is time-consuming, thus for each grid, we compute the upper bound of screen-space error, called error bound, to guide the selection of the LOD level.

The screen-space error is related to two factors, the projection f that is inversely proportional to the distance to the viewpoint, and the object-space geometric error that is related to the heightmap. Therefore, we can combine the independent maximum value of each factor to calculate the error bound for grid B_j. Specifically, the maximum object-space geometry error of level i can be calculated as:

$$\eta_i = \max_{(x,z)\in B_j} |H_0(x,z) - H_i(x,z)|. \tag{5}$$

Then we can select the maximum LOD level i of heightmap for grid B_j with Equation 1 that produces geometric error smaller than a given threshold, ξ, at every pixel:

$$\begin{aligned} E_i \le \hat{E}_i < \xi, \\ \hat{E}_i = max\{|P(x_0,y_0,z_0) - P(x_0,y_0+\eta_i,z_0)|, \\ |P(x_0,y_0,z_0) - P(x_0,y_0-\eta_i,z_0)|\} \end{aligned} \tag{6}$$

where $\hat{E}_i$ is the maximum screen-space error of grid B_j with level i. (x_0,y_0,z_0) is the closest point to the viewpoint in grid B_j. E_i is the true screen-space error of grid B_j with level i, which has been denoted in Equation 1.

4.2. Efficient η_i Calculation using SFAMF and SATs

An efficient algorithm to calculate object-space geometry error η_i is critical for reducing the overhead of our algorithm. Because the change of the viewpoint produces a dynamic grid layout on the clipmap, we precompute η_i as Kang et al. [20] did. Instead, we adopt Simple Fast approximation Minimum/Maximum filter (SFAMF) [51] algorithm to calculate the maximum object-space error for each clipmap's grid. SFAMF is an efficient algorithm for estimating the minimum/maximum value of a region by using p-norm to approximate the infinity norm in constant time with SATs.

Specifically, given a 1D vector $\mathbf{x} \in \mathcal{R}^N$, when $0 < x_i < 1$ and $p >> 1$, the p-norm can approximate the infinity norm, i.e., the maximum element, as:

$$\max_{i=1}^{N} x_i \approx (\frac{1}{N}\sum_{i=1}^{N} x_i^p)^{\frac{1}{p}}. \tag{7}$$

We apply SFAMF in 2D texture data with SATs. The value at any point in the SATs is the sum of all the pixels above and to the left of the point. Therefore the sum of a rectangle sub-region can be directly calculated from the SATs as [1]:

$$\begin{aligned} T_\Theta = \sum_{\substack{x_0<x\le x_1 \\ y_0<y\le y_1}} T(x,y) = S(x_1,y_1) - S(x_0,y_1) - \\ S(x_1,y_0) + S(x_0,y_0). \end{aligned} \tag{8}$$

where Θ denotes the region $\Theta \triangleq \{x,y; x_0 < x \le x_1, y_0 < y \le y_1\}$, $T(x,y)$ means the value on the point (x,y) of the original table, $S(x,y)$ represents the value on the point (x,y) of SATs.

Then we can calculate the mean $A_\Theta = \frac{1}{(x_1-x_0)(y_1-y_0)} T_\Theta$ and the maximum value $M_\Theta = [A_\Theta]^{\frac{1}{p}}$ in the rectangle sub-region.

In conclusion, our approach selects levels of the clipmap's grids by bounding the maximum screen-space errors under a given threshold. The maximum screen-space errors are estimated by projecting object-space geometric errors to the screen space. We precompute object-space geometric errors for each level and store in the SATs, then SFAMF is utilized to estimate maximum object-space geometric errors of each clipmap's grid in constant time.

5. Terrain Shading

To compute the integration in Equation 4 efficiently in real-time, we process each term separately. Similar to previous related works [52, 53, 32], we adopt spherical Gaussian (SG)

functions to approximate lighting terms $L(x,\omega_i)$ and BRDF terms $\bar{\rho}_n(\omega_i,\omega_o)$. Subsequently, we prebake the visibility from a few directions for each texel of the heightmap and store these values as textures. Furthermore, we split the hemisphere of the integration in Equation 4 into a few rectangle patches and sum each patch's result as the final integration. The visibility term $\bar{\rho}_v(\omega_i)$ of each patch will be estimated with the prebaked visibility maps.

Technically, can be represented as von Mises-Fisher(vMF) distribution [28, 32]. Then, a more compact approximation can be achieved by merging vMF lobes into a few representatives to reduce the integration overhead [32]. Therefore, the BRDF term $\bar{\rho}_n(\omega_i,\omega_o)$ in Equation 4 can be shown as follows:

$$\bar{\rho}_n(\omega_i,\omega_o) = vMF(\omega_i,\omega_o;r_M), \tag{9}$$

where the vMF distribution is characterized by an unnormalized vector $r_M \in R^3$. The detailed derivation is in supplementary materials.

Since a vMF lobe r_M can be regarded as a normalized SG function and the lighting term $L(x,\omega)$ can also be represented as multiple SG functions, meanwhile, the vector product of two SG function is still represented as another SG, Equation 4 can be represented as the sum of the integrations of multiple SG functions over the hemisphere:

$$L_o(\Omega,\omega_o) = \sum_{t=0}^{T}\int_{S^2} SG_t(\omega;p,\lambda,\mu)\bar{\rho}_v d\omega, \tag{10}$$

where T is the number of SG function, which is the product of the numbers of light SG and BRDF SG; $SG_t(\omega;p,\lambda,\mu) = \mu e^{\lambda(\omega\cdot p-1)}$; $p \in S^2$ is the lobe axis; $\lambda \in (0,\infty)$ is the lobe sharpness; $\mu \in R$ is the lobe amplitude; ω denotes ω_i under diffuse components and ω_h under specular components respectively. A spherical warp [52] is needed to transform light space to half vector space.

Wang et al. [52] introduced SSDF to represent visibility and approximated Equation 10 through parameterizing the interior of the visible region, Iwasaki et al. [53] introduced integral SG to Equation 10. Both of them are complex, time-consuming and unsuitable for large-scale scenes.

Instead, we construct local spherical coordinates for each SG function by aligning the SG axis as $\mathbf{z}$ axis, where the direction ω can be parameterized by a polar angle θ, and an azimuthal angle φ, as $\omega(\theta,\varphi) = (\sin\theta\cos\varphi, \sin\theta\sin\varphi, \cos\theta)$. The integration of SG function can be represented as follows:

$$\begin{aligned} f_t(\theta_1,\theta_2,\varphi_1,\varphi_2) &= \int_{\theta_1}^{\theta_2}\int_{\varphi_1}^{\varphi_2} SG_t(\omega(\theta,\varphi);\mathbf{z},\lambda,\mu)\sin\theta d\varphi d\theta \\ &= \frac{\mu(\varphi_2-\varphi_1)}{\lambda}(e^{\lambda(cos\theta_1-1)} - e^{\lambda(cos\theta_2-1)}). \end{aligned} \tag{11}$$

Given this parameterization, we can split the hemisphere into a few rectangle patches and easily sum each patch's result and calculate the final integration as follows:

$$L_o(\Omega,\omega_o) \approx \sum_{t=0}^{T}\sum_{m=0}^{M}\sum_{n=0}^{N} f_t(\Omega_{m,n})\bar{\rho}_v(\Omega_{m,n}), \tag{12}$$

where the patch number is $M\times N$, $\Omega_{m,n}$ denotes the patch region $\Omega_{m,n} \triangleq \{\theta,\phi;0 \le \theta_m \le \theta \le \theta_{m+1} \le \frac{\pi}{2}, 0 \le \phi_n \le \phi \le \phi_{n+1} \le 2\pi\}$, and $\bar{\rho}_v(\Omega_{m,n})$ is the average visibility in this region.

Finally, we utilize SATs to store and acquire unnormalized vectors r and visibilities according to the current pixel footprint Ω. r_M in Equation 9 can directly be acquired through SATs. Moreover, since we prebake the visibility in the object space, a coordinate transformation is necessary to estimate $\bar{\rho}_v$ for each patch because Equation 11 is applied in the local coordinate of the SG axis.

6. Implemention

We now outline the implementation of our approach described in Section 4 and Section 5 regarding the following aspects:

1. Preprocessing;
2. LOD selection of clipmap's grids (Section 4);
3. Terrain shading (Section 5);
4. Performance Optimization.

Preprocessings:

LOD selection of the clipmap's grid: As the viewer moves, change accordingly, and thus each grid's LOD selection needs to be updated at run-time. Our approach traverses from the coarsest level to the finest level of the heightmap for each grid. The screen-space errors of each level will be calculated through the projection of the precomputed object-space errors stored in the SATs. With the help of SATs, the maximum object-space error for each grid can be efficiently acquired through our SFAMF [51] algorithm. If a screen-space error is less than a predefined threshold ε, set to 4 pixels as default in our implementation, the current level is chosen as the grid's LOD level. Moreover, we add flanges [54] around grids to avoid visual crack due to different resolutions of neighboring grids. Algorithm 1 illustrates the whole process about how to select an accurate LOD level of the clipmap's grid. For the selection of p-norm in Equation 7 to approximate the infinity norm, we tested various values and compared the corresponding errors in the SFAMF algorithm. While large p

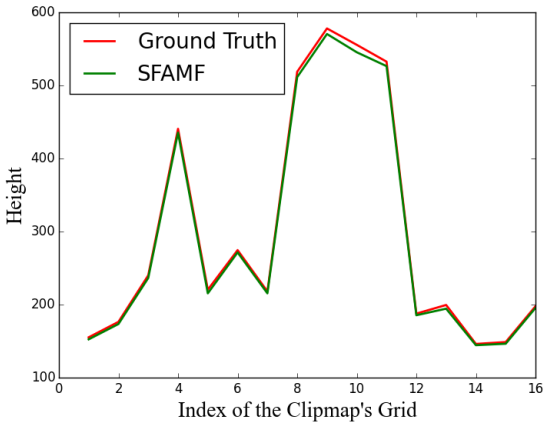


Fig. 2. Comparison of the estimated maximum value between the SFAMF and the Ground Truth in clipmap's grids.

yields accurate estimates, it will cause a bug related to numerical overflow, i.e., the intermediate output is too small for floating numbers.

The Root Mean Square Error (RMSE) between the SFAMF algorithm and Ground Truth is about 4.96, while the average height is 298.94.

Algorithm 1 Selection of LOD level of clipmap's grid

Input: terrain heightmap's SATs; grid; threshold ε
Output: level of clipmap's grid

$level \leftarrow MAXGridLOD$
$uv_{lb} = grid.center - vec2(grid.size/2, grid.size/2)$
$uv_{rb} = grid.center + vec2(grid.size/2, -grid.size/2)$
$uv_{lt} = grid.center + vec2(-grid.size/2, grid.size/2)$
$uv_{rt} = grid.center + vec2(grid.size/2, grid.size/2)$
$area = grid.size * grid.size$
while $level \geq 1$ **do**
  $value = (SATs_{level}[uv_{rt}] - SATs_{level}[uv_{lt}] - SATs_{level}[uv_{rb}] + SATs_{level}[uv_{lb}])/area$
  $max\_object\_error = pow(value, 1.0/p)$
  $max\_screen\_error = Project(grid, max\_object\_error)$
  if $max\_screen\_error \leq \varepsilon$ **then**
    return level
  end if
  $level \leftarrow level - 1$
end while
return level

Terrain shading: Outgoing radiance is calculated as described in Section 5. The lighting, clamped cosine and BRDF terms are fitted by SG functions. SATs are represented as a 2D four-channel 32-bit floating texture, which can be uploaded to the GPU for shading and applied to filter visibility map and normal map in real-time. Algorithm 2 denotes the pseudocode for filtering texture through SATs in the fragment shader.

Algorithm 2 Filter texture through SATs

Input: SATs for texture; uv coordinate
Output: filtering value;

$dx = dFdx(uv)$
$dy = dFdy(uv)$
$bbox = vec2(max(dx.x, dy.x), max(dx.y, dy.y))$
$vec4\ \ ret = texture(SATs, uv + vec2(-0.5 * bbox.x, -0.5 * bbox.y))$
$ret = ret - texture(SATs, uv + vec2(0.5 * bbox.x, -0.5 * bbox.y))$
$ret = ret - texture(SATs, uv + vec2(-0.5 * bbox.x, 0.5 * bbox.y))$
$ret = ret + texture(SATs, uv + vec2(0.5 * bbox.x, 0.5 * bbox.y))$
return $ret/(bbox.x * bbox.y)$

Algorithm 3 shows the pseudocode for computing diffuse/specular radiance, where m and n are tessellation factors of spherical space. In our implementation, we set m, n as 4 and 8 respectively. The function "TransferVis" in Line of Algorithm 3 is to transfer the pre-sampled visibility to the local coordinate system. Then, "PatchVis" function in Line is to average the pre-sampled visibility in the region of each rectangle patch and regard this value as the approximated visibility.

Algorithm 3 Diffuse/Specular radiance

Input: ;
Output: ;

r = SATs(map)
$\mu = \frac{r}{||r||}, \kappa = \frac{3||r|| - ||r||^3}{1 - ||r||^3}$
$\bar{\rho}_n = vMF(\omega; \mu, \kappa)$
vis = SATs(visibility map)
for i in SG light numbers **do**
  if Isspecular **then**
    localSG = warpSG(SGlight[i], ω_h)
  else
    localSG = SGlight[i]
  end if
  $tempSG \leftarrow$ SGProduct(localSG, $\bar{\rho}_n$)
  $newvis \leftarrow$ TransferVis(vis, tempSG)
  $L \leftarrow 0$
  for j in range(0,m) **do**
    for k in range(0,n) **do**
      $\bar{\rho}_v \leftarrow$PatchVis(newvis)
      $L = L + \bar{\rho}_v \cdot f(\theta_j, \theta_{j+1}, \varphi_k, \varphi_{k+1}, tempSG)$
    end for
  end for
  $L_d/L_s += k_d/k_s * L$
end for

Performance Optimization: First, we employ backface culling and frustum culling for performance optimization. We use a bounding box representing a grid, where the height of

a bounding box is the precomputed maximum and minimum height in the grid.

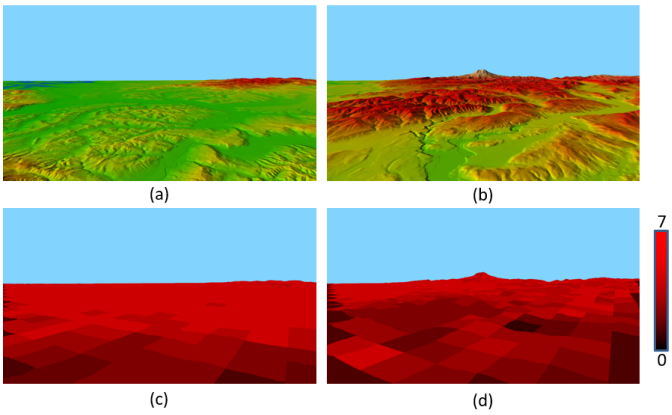


Fig. 3. Comparison of LODs between terrain with gentle slope and steep slope. Darker color means heightmap with lower levels. (a) A gentle slope terrain result, (b) a steep slope terrain result, (c),(d) the levels of the clipmaps' grids.

7. Results

The algorithms represented in this paper are implemented in OpenGL on a 3.4GHz Intel Core i7-3770 CPU with an NVIDIA GeForce GTX 1080Ti GPU. The terrain data is a 16385^2 grids of the Puget Sound area with 16-bit height values at 10m spacing and represents a wide range of $160km \times 160km$.

The texture is a 16384×16384 image and the screen resolution is 1280×720.

7.1. LOD Selections on Gentle vs. Steep Slopes

Figure 3 represents a comparison between gentle and steep slope terrain rendering. There are two pictures for each group: Figure 3(a) and 3(b) show the rendering results, Figure 3(c) and 3(d) represent the levels of clipmap's grids. We use the red color to represent the highest level and black color to denote the lowest level. In this paper, the 0th level is regarded as the finest result.

In general, low levels of clipmap's grids are used at positions close to the viewpoint, while high levels are applied for distant positions. However, our approach will choose levels according to the screen-space errors. As shown in Figure 3(b), low levels are even adopted in some distant grids to preserve more geometric details for mountains in the distance. The results demonstrate that the screen-space error works better than screen distance for LOD level selections.

7.2. Different Error Thresholds

Figure 4 denotes three results with different error thresholds. The top images are shading results, while the bottom images are the difference between each result and the ground truth. The red pixels of the error image express different locations with Ground Truth. The RMSE of 1/2/4 pixel error threshold is 0.0068, 0.014, 0.026, respectively. Besides, our method also supports on-line interactive adjustments on error thresholds.

7.3. Our Method vs. Geometry Clipmaps

are based on screen space but differ in the mechanism of LOD determination. Our method uses screen-space geometric error rather than distance as the criterion to determine the LOD levels of clipmap's grids. Our method suits especially well for cliffy terrain rendering, preserving enough details even for distant mountains. Figure 7 illustrates two groups of images for comparison, where each image contains four insets about local details or one difference image between the method and Ground Truth. The red pixels of the difference image on the bottom right corner express error locations. Ge et al. [22] had combined the Geometry Clipmaps with novel hardware techniques, which was considered as one of the latest improvement methods of Geomtry Clipmaps. In the distant mountain areas, our method depicts mountain gullies and ridges vividly, while the mountains of Geometry Clipmaps [23, 22] are overly smoothened without details.

We choose three typical scenes with different pixel error thresholds in the experiment. As can be seen from Table 1, for Geometry Clipmaps [23] and Ge et al. [22], triangle counts and time remain relatively stable among three scenes, while these two attributes change reasonably with pixel errors in different scenes for our method. Our method outperforms Geometry Clipmaps [23, 22] in a flat terrain scene since we have lower triangle counts. Although the time is reduced in the steep terrain scene, there is still enough room for interactive rendering, yet we preserve better details. Also, modern high-performance GPU gives us an edge over other terrain-rendering algorithms.

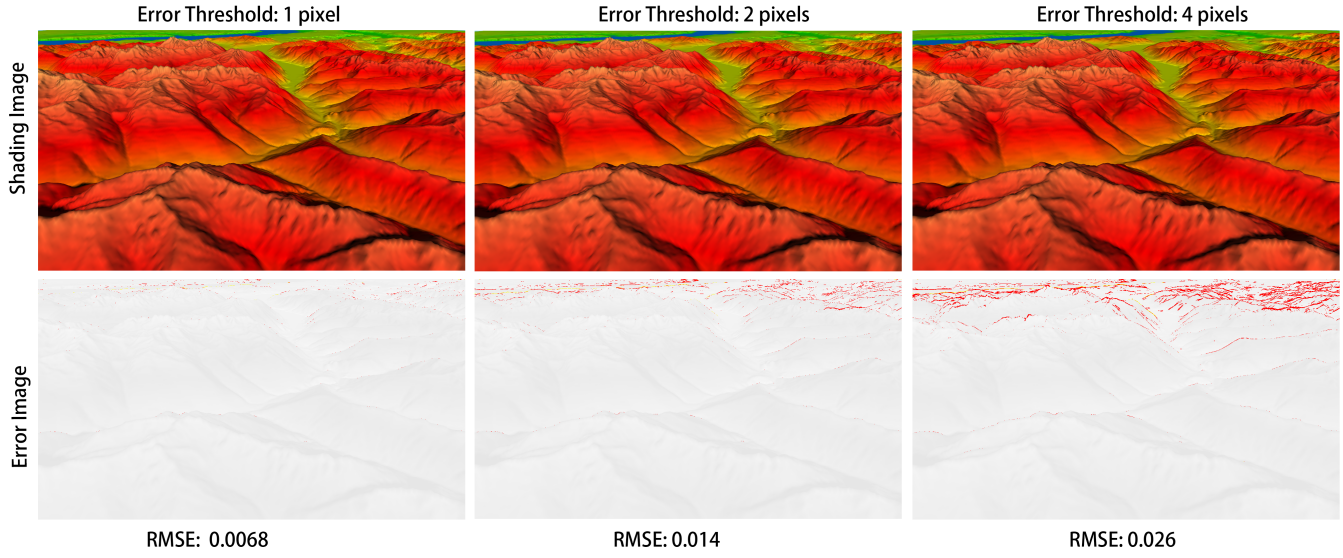


Fig. 4. Comparison of geometric details between different error thresholds. SSIM values are shown in the bracket.

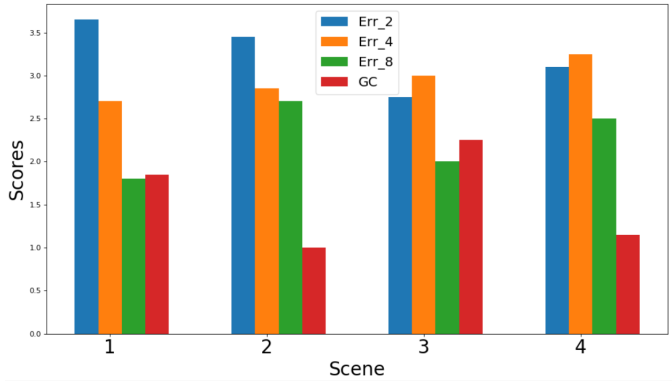


Fig. 5.

Scene	GC[23]		Ge[22]	ε	Our method		Kang[20]
	Time	Tris.	Time		Time	Tris.	Time
general	0.286	1065k	0.286	4	0.312	1658k	0.322
				2	0.439	2459k	0.455
				1	0.535	3150k	0.662
steep	0.278	1059k	0.303	4	0.408	2341k	0.382
				2	0.5	3133k	0.469
				1	0.625	4238k	0.671
gentle	0.263	1269k	0.286	4	0.238	842k	0.312
				2	0.270	1112k	0.344
				1	0.361	1762k	0.404

Table 1. Performance comparison. ε is the screen-space error threshold and the time unit is milliseconds. The chosen scenes are shown in Figure 8

7.4. Our Method vs. Mesh Hierarchy

In Table 1 with the same scenes, we also compare the performance of our method and quadtree-based terrain rendering algorithm [20], which is the state of art in Mesh Hierarchy. Both two methods share the same basis of the geometric error bound, so we have similar results. Both Kang et al. [20] and Ge et al. [22] tessellate the mesh in real-time, which also are limited by tessellation factors and may result in additional time, while our method could pre-store the level of grids as vertex buffers in the video memory. Therefore as shown in Table 1,

Moreover, we also have implemented an Android version of our method in OpenGL ES 3.0, which doesn't support tessellation shader.

7.5. Our Method vs. Other Shading Filtering Methods

To evaluate our shading algorithms, we compared images rendered by our methods with generated through direct sampling, non-linear filtering method. The ground truth image is generated by 256× super-sampling.

However, Xu et al. [32] handle the visibility term. LEDAR [27] works better on microfacet theory but it doesn't have any effect on macroscopic visibility. Moreover, it lacks the ability to handle multi-lobe BRDFs.

In order to validate our shading model, we extend our experiments to . As shown in Figure 9, our results are closer to Ground Truth.

Furthermore, we demonstrate our approach in real complex terrain datasets, as shown in Figure 10. Our experiments compare three algorithms in the Puget Sound datasets. Similarly, direct sampling will be brighter than other methods and quadtree structure will blur the scenes. The bottom in Figure 10 demonstrate the difference between each method and Ground Truth. The red pixels show color differences between the two images are beyond the shading error metric. Our experiments indicate that our method produces fewer errors than others.

As shown in Figure 6, our approach has nearly three times of

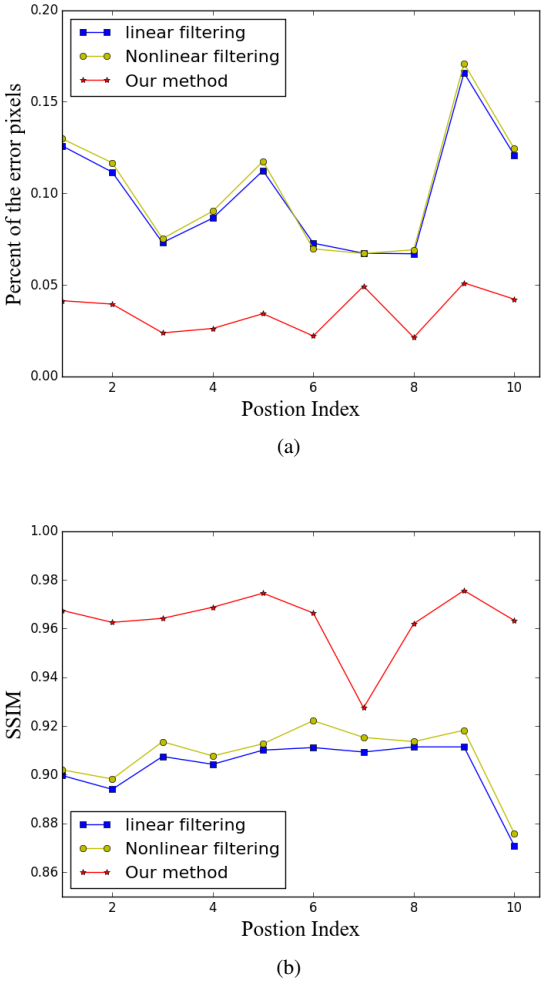


Fig. 6. Comparison of shading errors of different algorithms, (a) denotes the percentage of different pixels, (b) represents the similarity between each algorithm and ground truth.

improvements in the shading errors. SSIM [56] algorithm is also applied in our experiments, we could conclude that our results closely resemble the ground truth. The average frame rate of our shading method with dynamic lighting and shadows is 50-90 fps depending on terrain slopes, which is still enough for real-time rendering.

8. Conclusion and Future Work

Firstly, our approach uses the precomputed object-space geometric errors and the proposed 2D SFAMF method to select the LOD level of clipmap's grids SATs are adapted to conservatively estimate geometric errors in real-time. Secondly, SG functions are applied to approximate the lighting, BRDFs and clamped cosine components, and the outgoing radiance is acquired in real-time by integrating the SG functions and visibility terms, where visibility and normal values are fetched from SATs.

We demonstrate interactive terrain rendering over the Puget Sound area model with a wide range of $160km \times 160km$. Our method preserves more geometric details in comparison with Geometry Clipmaps while still maintaining relatively high performance. Moreover, our approach supports shadow effects, dynamic lighting, and different kinds of BRDFs. SATs enable general texture filtering to acquire more accurate terrain surface signals. Experiments reveal that, in terms of rendering quality, our method outperforms direct sampling and quadtree structure.

In the future, we would like to investigate more sophisticated methods for filtering the overall rendering equation to reduce shading errors and explore more photorealistic terrain-rendering algorithms.

Acknowledgements

This work was supported in part by NSFC (No. 61872319), Zhejiang Provincial NSFC (No. LR18F020002) and National Key R&D Program of China (No. 2017YFB1002605).

References

[1] Crow, FC. Summed-area tables for texture mapping. In: ACM SIGGRAPH computer graphics; vol. 18. ACM; 1984, p. 207–212.
[2] De Floriani, L, Marzano, P, Puppo, E. Multiresolution models for topographic surface description. The Visual Computer 1996;12(7):317–345.
[3] Cohen-Or, D, Levanoni, Y. Temporal continuity of levels of detail in delaunay triangulated terrain. In: Proceedings of Seventh Annual IEEE Visualization '96. 1996,.
[4] Fioriani, LD, Magillo, P, Puppo, E. Building and traversing a surface at variable resolution. Proceedings IEEE Visualization 1997;:103–110.
[5] Hoppe, H. Smooth view-dependent level-of-detail control and its application to terrain rendering. In: Proceedings of the Conference on Visualization '98. 1998,.
[6] Duchaineau, M, Wolinsky, M, Sigeti, DE, Miller, MC, Aldrich, C, Mineev-Weinstein, MB. Roaming terrain: Real-time optimally adapting meshes. In: Proceedings. Visualization '97 (Cat. No. 97CB36155). 1997,.
[7] Pajarola, R. Large scale terrain visualization using the restricted quadtree triangulation. In: Proceedings Visualization '98 (Cat. No.98CB36276). 1998,.
[8] Lindstrom, P, Pascucci, V. Terrain simplification simplified: a general framework for view-dependent out-of-core visualization. IEEE Transactions on Visualization and Computer Graphics 2002;.
[9] Levenberg, J. Fast view-dependent level-of-detail rendering using cached geometry. In: IEEE Visualization, 2002. VIS 2002. 2002,.
[10] Cignoni, P, Ganovelli, F, Gobbetti, E, Marton, F, Ponchio, F, Scopigno, R. Bdam batched dynamic adaptive meshes for high performance terrain visualization. Computer Graphics Forum 2003;.
[11] Cignoni, P, Ganovelli, F, Gobbetti, E, Marton, F, Ponchio, F, Scopigno, R. Planet-sized batched dynamic adaptive meshes (p-bdam). In: Proceedings of the 14th IEEE Visualization 2003 (VIS'03). VIS '03; 2003,.
[12] Hitchner, LE, McGreevy, M. Methods for user-based reduction of model complexity for virtual planetary exploration. In: Electronic Imaging. 1993,.
[13] Bishop, L, Eberly, D, Whitted, T, Finch, M, Shantz, M. Designing a pc game engine. IEEE Computer Graphics and Applications 1998;18(1):46–53.
[14] Wagner, D. Appeared in shader-x 2 terrain geomorphing in the vertex shader 2004;.
[15] Bösch, J, Goswami, P, Pajarola, R. Raster: Simple and efficient terrain rendering on the gpu 2009;.
[16] Strugar, F. Continuous distance-dependent level of detail for rendering heightmaps. Journal of graphics, GPU, and game tools 2009;14(4):57–74.

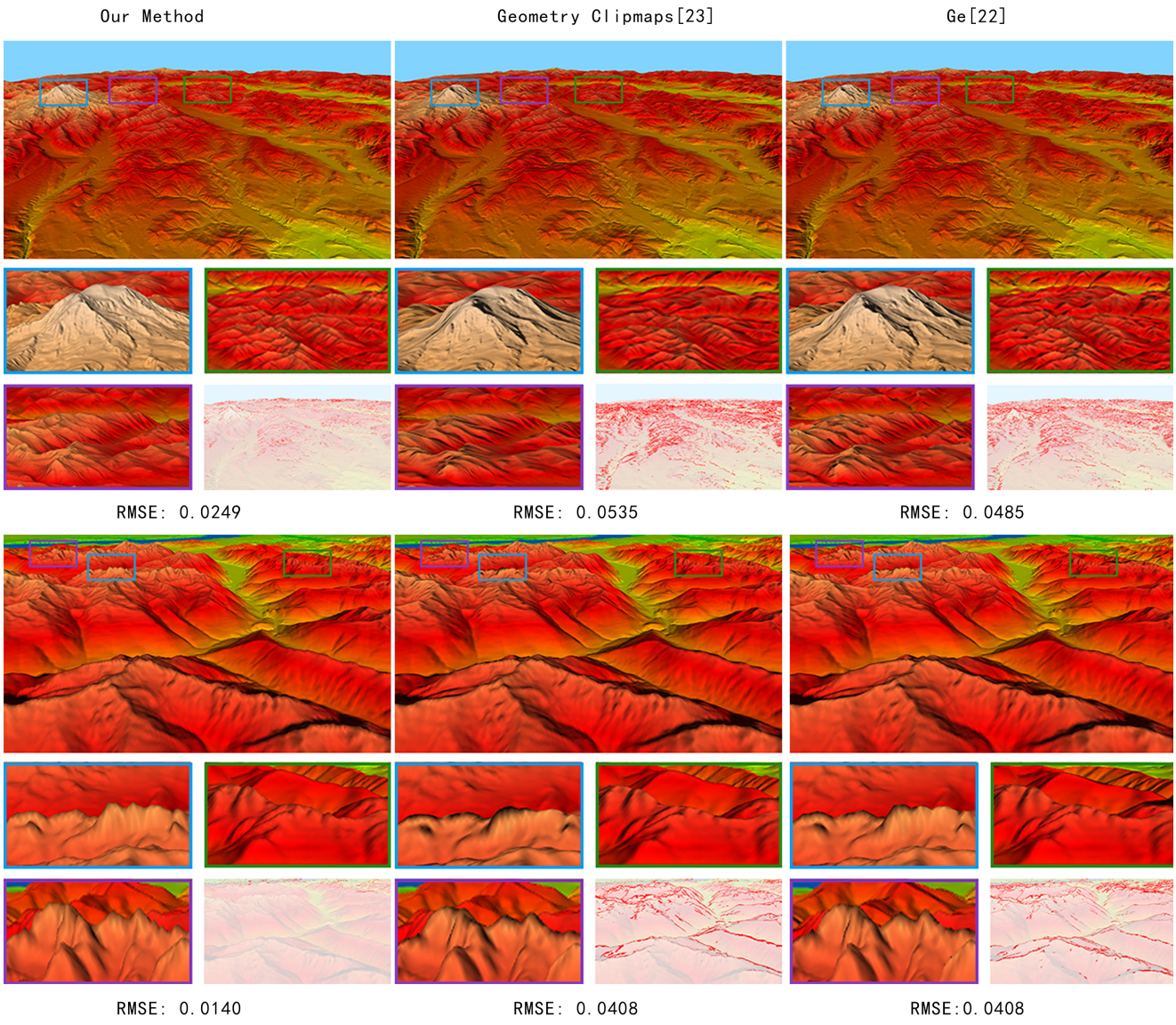


Fig. 7. Comparison of geometric details between Our Methods, Geometry Clipmaps [23] and Ge et al. [22].

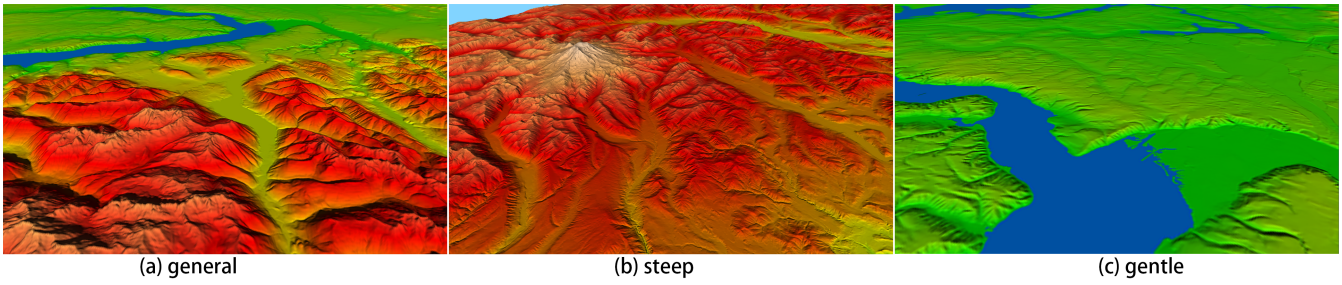


Fig. 8. Scenes of Table 1

[17] Dick, C, Krüger, J, Westermann, R. Gpu-aware hybrid terrain rendering. Proceedings of IADIS computer graphics, visualization, computer vision and image processing 2010;10:3–10.
[18] Cantlay, I. Directx 11 terrain tessellation. Nvidia whitepaper 2011;8(11):3.
[19] Ripolles, O, Ramos, F, Puig-Centelles, A, Chover, M. Real-time tessellation of terrain on graphics hardware. Computers Geosciences 2012;41(none):147–155.
[20] Kang, H, Jang, H, Cho, CS, Han, J. Multi-resolution terrain rendering with gpu tessellation. The Visual Computer 2015;31(4):455–469.
[21] Zhai, R, Lu, K, Pan, W, Dai, S. Gpu-based real-time terrain rendering: Design and implementation. Neurocomputing 2016;171:1–8.
[22] SONG, G, YANG, H, JI, Y. Geometry clipmaps terrain rendering using hardware tessellation. IEICE Transactions on Information and Systems 2017;E100.D(2):401–404.
[23] Asirvatham, A, Hoppe, H. Terrain rendering using gpu-based geometry

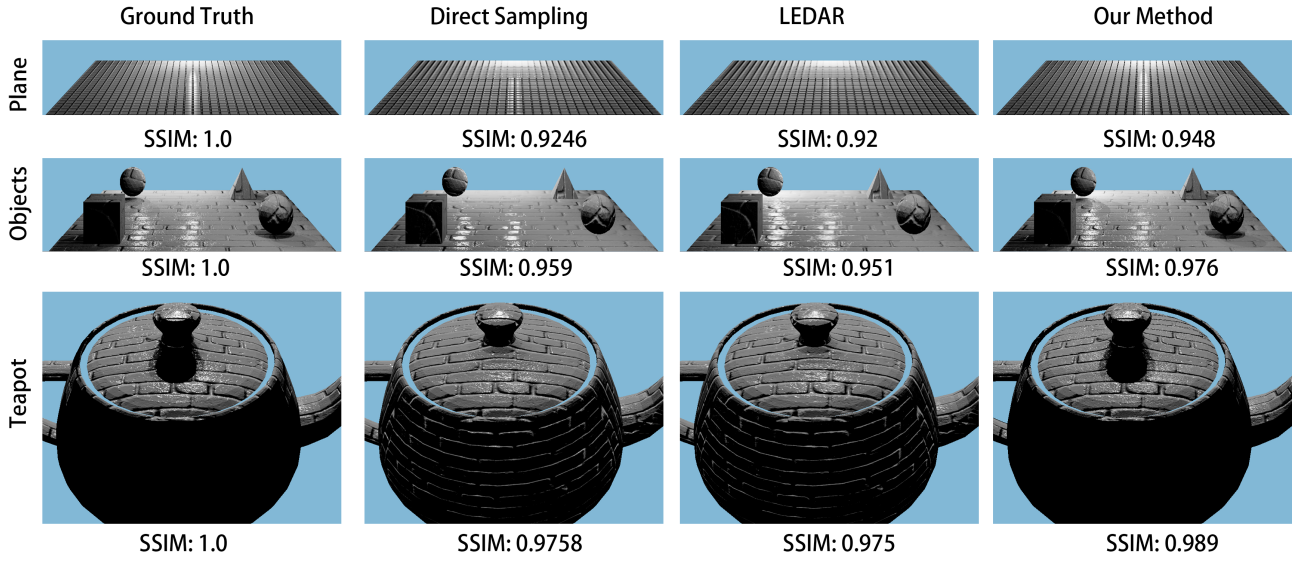


Fig. 9. Comparison of renderings of three models with (a) Ground Truth, (b) Direct sampling, (c) LEDAR [27], and (d) Our method.

clipmaps. GPU gems 2005;2(2):27–46.

[24] OGRE - Open Source 3D Graphics Engine. ???? URL: https://www.ogre3d.org/.

[25] Cozzi, P, Ring, K. 3D Engine Design for Virtual Globes. 1st ed.; CRC Press; 2011. ISBN 978-1568817118.

[26] Bruneton, E, Neyret, F. A survey of nonlinear prefiltering methods for efficient and accurate surface shading. IEEE Transactions on Visualization and Computer Graphics 2012;18(2):242–260.

[27] Dupuy, J, Heitz, E, Iehl, JC, Poulin, P, Neyret, F, Ostromoukhov, V. Linear efficient antialiased displacement and reflectance mapping. ACM Transactions on Graphics (TOG) 2013;32(6):211.

[28] Han, C, Sun, B, Ramamoorthi, R, Grinspun, E. Frequency domain normal map filtering. ACM Transactions on Graphics (TOG) 2007;26(3):28.

[29] Heitz, E, Nowrouzezahrai, D, Poulin, P, Neyret, F. Filtering color mapped textures and surfaces. In: Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. ACM; 2013, p. 129–136.

[30] Olano, M, Baker, D. Lean mapping. In: Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games. ACM; 2010, p. 181–188.

[31] Toksvig, M. Mipmapping normal maps. journal of graphics tools 2005;10(3):65–71.

[32] Xu, C, Wang, R, Zhao, S, Bao, H. Real-time linear brdf mip-mapping. In: Computer Graphics Forum; vol. 36. Wiley Online Library; 2017, p. 27–34.

[33] Pajarola, R, Gobbetti, E. Survey of semi-regular multiresolution models for interactive terrain rendering. The Visual Computer 2007;23(8):583–605.

[34] Kang, H, Sim, Y, Han, J. Terrain rendering with unlimited detail and resolution. Graphical Models 2018;97:64 – 79.

[35] Clasen, M, Hege, HC. Terrain rendering using spherical clipmaps. In: Proceedings of the Eighth Joint Eurographics/IEEE VGTC conference on Visualization. Eurographics Association; 2006, p. 91–98.

[36] Dimitrijevic, AM, Rancic, DD. Ellipsoidal clipmaps - a planet-sized terrain rendering algorithm 2015;.

[37] Feldmann, D, Hinrichs, K. Gpu based single-pass ray casting of large heightelds using clipmaps. 2012,.

[38] Barnard, C. Applying tessellation to clipmap terrain rendering 2014;.

[39] Evans, W, Kirkpatrick, D, Townsend, G. Right-triangulated irregular networks 2001;30(2).

[40] Lindstrom, P, Koller, D, Ribarsky, W, Hodges, LF, Faust, N, Turner, GA. Real-time, continuous level of detail rendering of height fields. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH 96; 1996,.

[41] Lindstrom, P, Pascucci, V. Visualization of large terrains made easy. In: Proceedings of the Conference on Visualization 01. VIS 01. ISBN 078037200X; 2001,.

[42] Oat, C, Sander, PV. Ambient aperture lighting. In: Proceedings of the 2007 symposium on Interactive 3D graphics and games. ACM; 2007, p. 61–64.

[43] Snydre, J, Nowrouzezahrai, D. Fast soft self-shadowing on dynamic height fields. In: Computer Graphics Forum; vol. 27. Wiley Online Library; 2008, p. 1275–1283.

[44] Chajdas, MG, Reichl, F, Dick, C, Westermann, R. High-quality shadows for streaming terrain rendering. In: Proceedings of Eurographics 2015 - Short Papers. 2015, p. 57–60.

[45] Tan, P, Lin, S, Quan, L, Guo, B, Shum, H. Filtering and rendering of resolution-dependent reflectance models. IEEE Transactions on Visualization and Computer Graphics 2008;14(2):412–425.

[46] Heitz, E, Dupuy, J, Crassin, C, Dachsbacher, C. The sggx microflake distribution. ACM Transactions on Graphics (TOG) 2015;34(4):48.

[47] Wu, L, Zhao, S, Yan, LQ, Ramamoorthi, R. Accurate appearance preserving prefiltering for rendering displacement-mapped surfaces. ACM Trans Graph 2019;38(4). URL: https://doi.org/10.1145/3306346.3322936. doi:10.1145/3306346.3322936.

[48] Kajiya, JT. The rendering equation. In: ACM Siggraph Computer Graphics; vol. 20. ACM; 1986, p. 143–150.

[49] Wu, H, Dorsey, J, Rushmeier, H. Characteristic point maps. Computer Graphics Forum 2009;28(4):1227–1236.

[50] Iwasaki, K, Dobashi, Y, Nishita, T. Interactive bi-scale editing of highly glossy materials 2012;.

[51] Tulleken, H. Simple fast approximate minimum/maximum filters. http://code-spot.co.za/2010/04/16/simple-fast-approximate-minimum-maximum-filters; 2010.

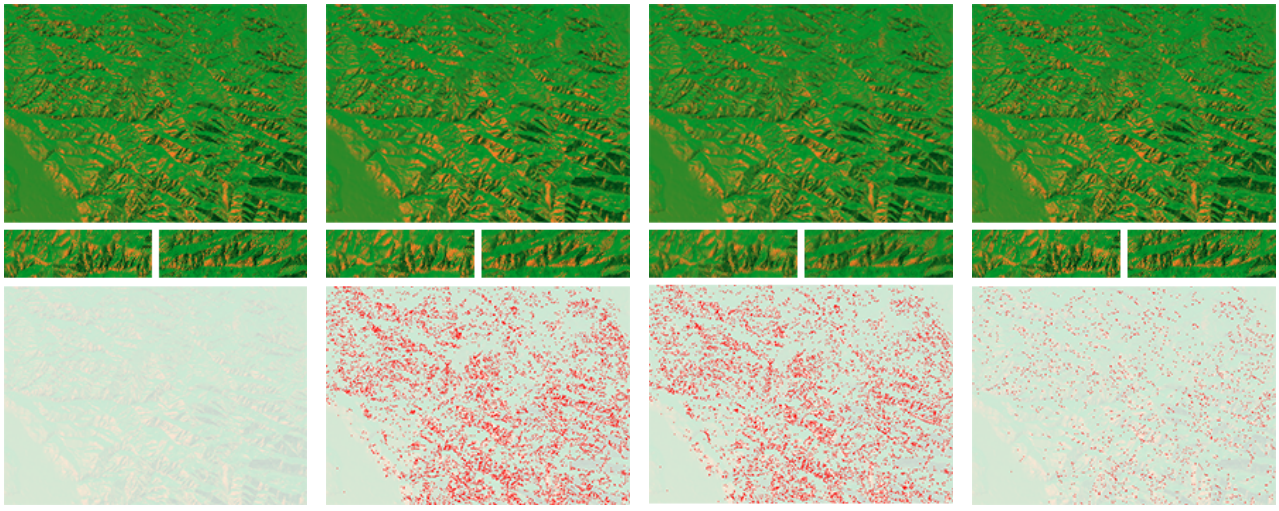
[52] Wang, J, Ren, P, Gong, M, Snyder, J, Guo, B. All-frequency rendering of dynamic, spatially-varying reflectance. In: ACM Transactions on Graphics (TOG); vol. 28. ACM; 2009, p. 133.

[53] Iwasaki, K, Furuya, W, Dobashi, Y, Nishita, T. Real-time rendering of dynamic scenes under all-frequency lighting using integral spherical gaussian. In: Computer Graphics Forum; vol. 31. Wiley Online Library; 2012, p. 727–734.

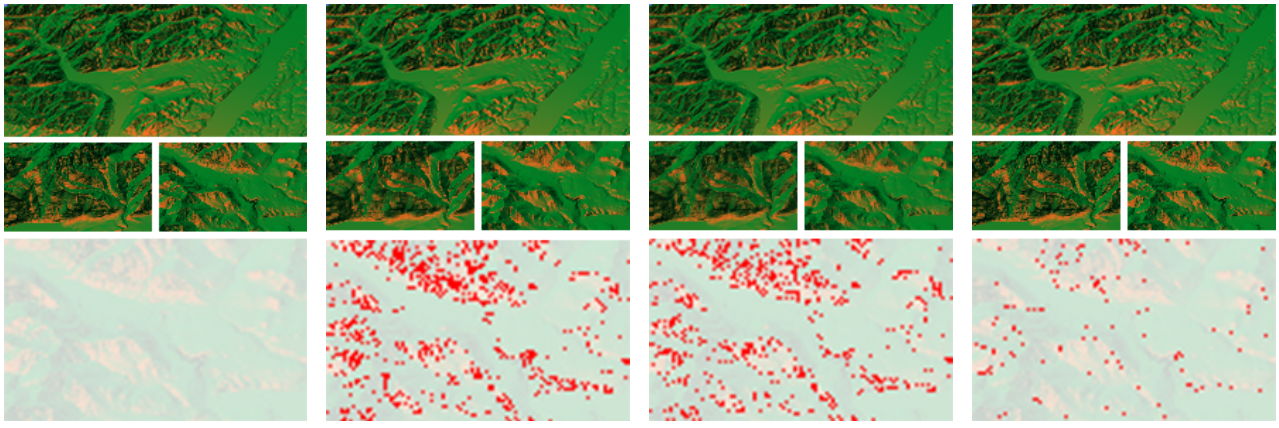
[54] Ulrich, T. Rendering massive terrains using chunked level of detail control. SIGGRAPH Course Notes 2002;.

[55] van Waveren, J. Software virtual textures 2012;.

[56] Wang, Z, Bovik, AC, Sheikh, HR, Simoncelli, EP. Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing 2004;13(4):600–612.



(a) Ground Truth (b) Direct Sampling (c) Nonlinear filtering with Mip-mapping (d) Our method with SATs



(a) Ground Truth (b) Direct Sampling (c) Nonlinear filtering with Mip-mapping (d) Our method with SATs

Fig. 10. Comparison of renderings of Puget Sound areas with (a) Ground Truth, (b) Direct sampling, (c) non-linear filtering with quadtree, and (d) Our methods. The middle show two close-up views of two local regions, the bottom show the difference between each method and Ground Truth