Charles University in Prague

Faculty of Mathematics and Physics

MASTER THESIS

Bc. Michal Lašan

Height map compression techniques

Department of Software and Computer Science Education

Supervisor of the master thesis:	Mgr. Martin Kahoun
Study programme:	Informatics
Specialization:	Software Systems

Prague 2016

Dedication.

Název práce: Komprese výškových map

Autor: Michal Lašan

Katedra: Kabinet software a výuky informatiky

Vedoucí diplomové práce: Mgr. Martin Kahoun, Univerzita Karlova v Praze

Abstrakt:

Klíčová slova:

Title: Height map compression techniques

Author: Michal Lašan

Department: Department of Software and Computer Science Education

Supervisor: Mgr. Martin Kahoun, Charles University in Prague

Abstract: The goal of this thesis is to design a suitable method for lossy compression of heightmap terrain data. This method should accept blocks of float samples of dimensions $2^n x 2^n$ at the input, for which it should be able to perform progressive mip-maps (progressive lower-resolution representations) decompression. For every mip-map, it should keep the reconstructed data within a certain maximum absolute per-sample error bound in the unit of meters adjustable by the user. Given these constraints, it should be as efficient as possible. Our method is inspired by the second generation of progressive wavelet-based compression scheme modified to satisfy the maximum-error constraint. We simplified this scheme by factoring out unnecessary computations in order to improve the efficiency. Our method can compress a 256x256 block in about 30 ms and decompress it in about 1 ms. Thanks to these attributes, the method can be used in a real-time planet renderer. It achieves the compression ratio of 37:1 on the whole Earth 90m/sample terrain dataset transformed and separated into square blocks, while respecting the maximum error of 5m.

Keywords: heightmap, lossy, compression, mip-map, guaranteed maximum error bound

Contents

1 Introduction **2**

2 The wavelets **4**

2.1 The introduction to second-generation wavelets 4

2.2 Comparisons between the wavelets, C-BDAM and our method . . 6

3 The outline of the method **8**

4 Details of the method **10**

5 Functional comparison to C-BDAM and wavelets **15**

6 Results **17**

7 Conclusion **20**

Conclusion **21**

List of Tables **24**

List of Abbreviations **25**

Attachments **26**

1. Introduction

A real-time whole-planet renderer must work with huge amounts of terrain data. In order to reach reasonable frame rates, its rendering pipeline has to use some kind of multiresolution (LOD-ing[1]) approach. There is a survey paper summarizing the best known multiresolution terrain rendering methods [7]. Some of them are designed to render just a flat area, others are able to render the whole planet. In the rest of this chapter, we will briefly describe those of them which also contain terrain data compression.

C-BDAM[2] [5] and P-BDAM[3] [2] perform the compression in the refinement of a node of their LOD hierarchy. Once the values of a certain node are known, they are used to predict the values of its children as accurately as possible. After that, the differences between these predictions and the real values are computed. These are called residuals. With the help of them, the real values can be restored with absolute accuracy. However, the residuals are then quantized to achieve better compression ratio which means that the compression is lossy. Then, they are losslessly compressed by an entropy codec. Both these methods are able to compute the residuals in the way which ensures that the error of the reconstructed data is kept within a maximum error bound adjustable by the user in every node of their LOD hierarchy. This can be achieved by a slight modification of the second-generation wavelet lifting scheme [9]. C-BDAM is designed to render just a flat area, whereas P-BDAM is able to render the whole planet.

Another paper [6] describes a method for rendering a flat portion of terrain. This method contains data compression based on the same principle - the residuals needed to reconstruct the children of a square node of the terrain LOD hierarchy are compressed. The computation of residuals is based on the wavelet-based JPEG2000 standard. This method is not able to reconstruct the data within a certain maximum-error bound which makes it less interesting to us. Besides, the visual artifacts between adjacent nodes of different LODs are not handled by its rendering pipeline.

The aim of our meth

In practice, many applications handle the real-time rendering well with LOD schemes tailored to their needs. In such cases, a compression method tied to a concrete LOD scheme (which is the case of the mentioned methods) is not feasible. This method handles only the compression, so it can be used as a plug & play component in an existing real-time renderer. Its only job is to compress a block of terrain height samples sized $2^n x 2^n$ and to provide fast progressive decompression of its mip-maps, while respecting the maximum error bound at every mip-map. The source code of the method is written modularly, so that any representation of the height samples can be compressed - doubles, floats or even arbitrary structures. It is inspired by C-BDAM - the compression method is extracted from the LOD scheme and simplified.

As a case study we have implemented this method as a plugin into an appli-

[1]LOD is the abbreviation of level of detail - degradation of quality of the displayed data with the growing distance in order to optimize the rendering

[2]Compressed Batched Dynamic Adaptive Meshes

[3]Planet Sized Batched Dynamic Adaptive Meshes

cation, which transforms the heights on the planet surface into 256x256 blocks of 32-bit float samples in the unit of meters, which are then stored separately and during the run fetched into a quadtree-based LOD hierarchy. The mip-maps of the blocks are used while looking at them from a side.

This approach introduces heavy redundancy of the data - a block corresponding to a certain quadtree node contains simplified blocks of its children and all these blocks are stored separately. To the contrary, in C-BDAM only the residuals needed to reconstruct the children from the parent node are stored.[4] However, the reason why this approach is used is that the user can navigate to any area almost immediately - only the data needed for the scene has to be fetched, without having to reconstruct it by traversing from the root. Moreover, this approach enables the user to flexibly extend the terrain data by high-resolution insets. The mentioned redundancy of the data emphasizes the need for as efficient compression method as possible, doing only what is required - providing the mip-maps while respecting the maximum-error bound of the samples inside each one of them.

In Chapter 2, we briefly describe the basic theory of wavelets and link C-BDAM and this method to it, in Section 3, we briefly describe the basic outline of the method. In Chapter 4, we describe the details of the method. In Chapter 5, we compare the core algorithm of this method to the algorithm of C-BDAM. We present the results in Section 6 and then discuss them in Section 7.

[4]The LOD structure in C-BDAM is not a quadtree, though

2. The wavelets

This chapter consists of two sections. In the first one (2.1), we will briefly and formally describe the main principle and usage of second generation wavelet transformation methods which are relevant for this thesis. In the second one (2.2), we will compare C-BDAM and our method to these methods. Even though C-BDAM is based on the same principle, it differs from these methods a bit, so we will describe the basic differences. Then we will perform the same basic comparison with our proposed method. Our method differs from the described wavelet scheme and C-BDAM a bit more which will be clarified in that section.

2.1 The introduction to second-generation wavelets

Basically, there are two generations of wavelets. The first generation uses dilated and translated wavelet function [1] for computation. The second one uses filter banks to perform high-pass and low-pass filtering [3]. The computational equivalency of these two approaches has been proven [4].

For this work, the second generation of discrete wavelet transform methods is most relevant, so we will briefly describe it in this section in order to give the reader an idea of the wavelet concept which is referred to in many places of this thesis. The second wavelet generation is much easier to understand than the first one, so it is possible to describe its basic idea in a few pages.

Every method of this generation consists of just several subsequent applications of lifting onto the input. The lifting is the basic step of the method. It splits the set of its input signal samples into two parts - low-pass (the low frequency information) and high-pass (residuals, the high frequency information). The lifting is firstly applied on the input set of signal samples and then is recursively applied to the low-pass part produced in the previous iteration until the length of the latest low-pass part is 1. In order to make this recursion possible, the count of samples of the original input of the method must be a certain power of two. If the length of the input is 2^n, the method performs n iterations of lifting. The described successive application of lifting on smaller and smaller input is called the bottom-top pass. We can imagine this as building a pyramid of low-pass outputs the first tier of which is the input itself and every following higher tier is the low-pass output of lifting applied to the tier right below. Every tier is half the width of the previous one and after the bottom-top pass, the highest tier has the width of 1.

After this bottom-top pass, we can perform the inverse top-bottom pass. This pass does not know how the produced pyramid looks, it only knows its highest tier, sized 1. However, it is supposed to be able to progressively reconstruct the whole pyramid from the top to the bottom, only utilizing the knowledge of the high-pass information. Producing a certain tier from the previos one is called the reconstruction which is the exact opposite of lifting.

At this point, you might ask what all these decompositions and backward compositions are good for. What makes them interesting is the fact that the bottom-top pass just needs the high-pass information (residuals) to fully reconstruct the input. This information tends to be sparse and input-dependent - the smoother the input, the less high-pass information it contains. If we compress it well, we can save much storage space. Thus, if we want to store a set of samples the count of which is a power of two in as little space as possible, we will not store the samples directly, but we will store just the compressed residuals produced by the successive iterations of lifting applied to the input. If we are not required to accurately reconstruct the input, we can even decimate (quantize) the residuals. Because this information often contains just details, its careful decimation does not deform the reconstruction much and ensures better compression ratio. One more interesting fact is that the residuals bound to lower tiers of the pyramid carry finer details than those bound to the higher ones. Thanks to this, the more-detailed (larger) sets of residuals can be compressed more aggressively than the less-detailed (smaller) ones. This is called progressive compression and it is used for example in JPEG standard [8].

In the following lines, we will describe the lifting and reconstruction steps more formally. Let us say that the lifting is given the input samples x_k. It splits them into the even ones: $x_{2k} = x_e$ and the odd ones: $x_{2k+1} = x_o$. This splitting is not yet based on any frequency properties of the samples, it is based just on their order. However, these two sets of samples will subsequently be modified, so that the even ones will contain the low-pass information and the odd ones will become the residuals - the high-pass information. This will be performed with the help of two operators: the prediction operator P and the update operator U. P will be used to produce the residuals d from x_o and U will be used to produce the low-pass part s from x_e.

Up to this point, just the common properties of the second-generation methods have been described. Now will come the differences between them. The only thing they differ in is the way they perform lifting and reconstruction. The way the lifting step is performed clearly determines the way how the reconstruction is performed, as the reconstruction must be the exact inverse of lifting. The lifting step varies in the order in which the operators P and U are applied. According to this, the methods can be split into two main groups - the prediction-first ones and the update-first ones.

In the prediction-first methods, the prediction is applied first:

$$d = x_o - P(x_e)$$
$$s = x_e + U(d)$$

The reconstruction must be the exact inverse:

$$x_e = s - U(d)$$
$$x_o = d + P(x_e)$$

In the update-first methods, the update operator is applied first:

$$s = x_e + U(x_o)$$
$$d = x_o - P(s)$$

Here is how the reconstruction looks then:

$$x_o = d + P(s)$$

$$x_e = s - U(x_o)$$

2.2 Comparisons between the wavelets, C-BDAM and our method

In this section, we will describe how C-BDAM and our method differ from the basic second-generation wavelet scheme and from each other. The lifting inside C-BDAM is a slight variation of the update-first approach. The main difference is that the input to the first update is not only x_o, but the whole x. In addition, the computation of s is not the summation of the product of x_e and U anymore, because inside $U(x)$, x_e is multiplied:

$$s = U(x)$$
$$d = x_o - P(s)$$

The inverse reconstruction is then:

$$x_o = d + P(s)$$
$$x_e = U^{-1}(x)$$

Moreover, the samples x are regularly distributed in the plane, so the spliting into x_o and x_e no longer depends on the indices of the samples, but on their positions instead (Fig. 2.1). Nevertheless, this is just a formal difference which has no effect on the computation. The size of x_o and x_e is still half the size of x which is crucial to keep the original form of lifting. Note that if the residuals d were simply quantized after lifting and used in the reconstructions inside the second top-bottom pass, each step of the reconstruction would increase the maximum absolute deviation from the original low-pass values produced on the first bottom-top pass. To ensure that the reconstructed values are within the maximum-error bound from their corresponding values produced in the first pass at each tier, the residuals computed in the first pass are slightly corrected according to the actual values in another additional top-bottom pass which then turns out to be identical to the reconstruction (decompression), except for the fact that in the following decompression, just the corrected residuals are used to progressively reconstruct the data.

The method proposed in this thesis shares the same main lifting principle with C-BDAM - it is update-first and uses the whole x as the input to U, but has several differences: the size of x_e and thus s is not half the size of x, but one fourth of it instead, as each four neighboring pixels of x are collapsed into one inside s (Fig. 4.1). Additionally, the lifting is not complete, because the prediction operator is not applied there and the computation of residuals is not performed there either. In the lifting of C-BDAM, just temporary approximate residuals are computed and they are corrected in the subsequent top-bottom pass, whereas in our method, the correct residuals which already ensure the satisfaction of the maximum-error bound constraint are computed directly in the second top-bottom pass, also utilizing the prediction operator. Similarly to C-BDAM, the computations inside this pass are identical to the reconstruction of the data, except for the fact that during the reconstruction, the residuals are not computed

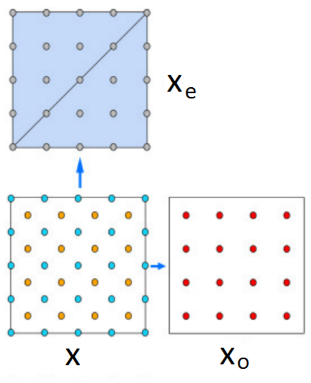

Figure 2.1: Lifting in C-BDAM - the samples x are split into the even ones (x_e) which will become low-pass (s) and the odd ones (x_o) which will become high-pass (d)
Source: C-BDAM [5] (edited)

anymore. Additionally, the prediction operator is applied multiple times in one step of the reconstruction which is explained in Chapter 4. The rationale behind these differences is explained in Chapter. 5.

3. The outline of the method

In this chapter, we will briefly describe how the compression works. Basically, two consecutive passes are performed on the input heightmap. The first bottom-top pass computes the target mip-maps - from the largest one to the smallest one. Those will be the mip-maps, against which the accuracy of reconstruction will be measured. The largest mip-map is the input itself. The second top-bottom pass constructs the compressed mip-maps from the smallest one to the largest one with respect to the target mip-maps in order to ensure that the maximum deviation of every compressed mip-map from its corresponding target mip-map is within the maximum error bound set by the user. The smallest of these mip-maps is just the suitably quantized sole value of the corresponding target mip-map. It is directly stored as first. The values of each following compressed mip-map are predicted from its previous compressed mip-map. For these mip-maps, we store just the residuals which are added to the predictions to satisfy the maximum deviation constraint.

More formally, the first pass is given the input square block of float height samples $\boldsymbol{L_n}$ sized $2^n x 2^n$ and produces n mip-maps $\boldsymbol{L_{n-1..0}}$ from it, one by one. The dimension of $\boldsymbol{L_i}$ is half the dimension of $\boldsymbol{L_{i+1}}$. Generally, $\boldsymbol{L_i}$ can be computed from $\boldsymbol{L_{i+1}}$ by any form of averaging of pixels - see the details in the following chapter.

The second top-bottom pass has already $\boldsymbol{L_{0..n}}$ available and computes $\boldsymbol{L^\bullet_{0..n}}$ - the compressed mip-maps. The dimension of $\boldsymbol{L_i}$ and $\boldsymbol{L^\bullet_i}$ is the same. The computation ensures that the maximum absolute deviation between their corresponding samples is not greater than $\boldsymbol{D}$ - the parameter set by the user. This will be denoted by:

$$maxdev(\boldsymbol{L_i}, \boldsymbol{L^\bullet_i}) \leq \boldsymbol{D},$$

where

$$maxdev(A, B) = \arg\max_{x,y} |A[x][y] - B[x][y]|$$

We will achieve this with the help of the uniform quantizer Q_D the quantization step of which is set to the maximum value which still respects this error bound:

$$maxdev(Q_D(x), x) \leq \boldsymbol{D},$$

where x is an arbitrary float sample or block of samples. The quantizing step of this quantizer is $2\boldsymbol{D} - 1$ in case $\boldsymbol{D} \geq 0.5$ and $2\boldsymbol{D}$ otherwise.

As we already mentioned, $\boldsymbol{L^\bullet_0}$ is just the quantized sole value of $\boldsymbol{L_0}$:

$$\boldsymbol{L^\bullet_0} = Q_D(\boldsymbol{L_0})$$

Thanks to the fact that the quantizer respects the maximum-error bound $\boldsymbol{D}$, $maxdev(\boldsymbol{L_0}, \boldsymbol{L^\bullet_0}) \leq \boldsymbol{D}$.

Then, the values of every following $\boldsymbol{L^\bullet_{i+1}}$ are predicted from the values of the previous $\boldsymbol{L^\bullet_i}$. The raw differences between the target values and the predicted values are denoted as $\boldsymbol{E_{i+1}}$ (the residuals). With the help of them and the predictions from $\boldsymbol{L^\bullet_i}$, we would be able to accurately reconstruct the target mip-map $\boldsymbol{L_{i+1}}$. However, these residuals are then quantized with the uniform quantizer Q_D to $\boldsymbol{E^{\bullet}_{i+1}}$. With the help of the quantized residuals, we are no longer able to accurately reconstruct $\boldsymbol{L_{i+1}}$, but thanks to the fact that the used quantizer keeps the maximum absolute error within the bound $\boldsymbol{D}$, we can guarantee that the reconstructed $\boldsymbol{L^{\bullet}_{i+1}}$ will satisfy the maximum-error constraint: $maxdev(\boldsymbol{L^{\bullet}_{i+1}}, \boldsymbol{L_{i+1}}) \leq \boldsymbol{D}$. Here is how we construct $\boldsymbol{L^{\bullet}_{i+1}}$:

$$\begin{aligned} \boldsymbol{E_{i+1}} &= \boldsymbol{L_{i+1}} - P(\boldsymbol{L^{\bullet}_{i}}) \\ \boldsymbol{E^{\bullet}_{i+1}} &= Q_D(\boldsymbol{E_{i+1}}) \\ \boldsymbol{L^{\bullet}_{i+1}} &= P(\boldsymbol{L^{\bullet}_{i}}) + \boldsymbol{E^{\bullet}_{i+1}} \end{aligned} \tag{3.1}$$

Thanks to the fact that the residuals $\boldsymbol{E_{i+1}}$ are computed with respect to the target mip-map $\boldsymbol{L_{i+1}}$, the maximum-error constraint is satisfied, no matter what values are in $\boldsymbol{L^{\bullet}_{i}}$ and what the prediction operator P looks like. At the end, the quantized residuals $\boldsymbol{E^{\bullet}_{0..n}}$ are compressed with the help of an entropy codec (Zlib) and stored ($\boldsymbol{E^{\bullet}_{0}} = \boldsymbol{L^{\bullet}_{0}}$). The order of their storage is from $\boldsymbol{E^{\bullet}_{0}}$ to $\boldsymbol{E^{\bullet}_{n}}$, so that progressive decompression is possible. The more accurate P is, the smaller the residuals are, thus the higher the compression ratio is. The details of the prediction operator used in this method are described in the following chapter. The higher $\boldsymbol{D}$ is, the less entropy there is among the residuals, thus the higher the compression ratio is, but the lower the reconstruction quality is.

The real-time decompression then just reads the stored quantized residuals and decompresses them with the help of the same entropy codec. Thanks to the fact that the residuals of a smaller mip-map are stored before the residuals of a larger mip-map, progressive decompression of mip-maps $\boldsymbol{L^{\bullet}_{0..n}}$ is possible, utilizing the same principle of producing predictions from the previous reconstructed mip-map and adding residuals to them (eq. 3.1). Of course, in order for this to work, the prediction operator must be identical to the one used in the compression.

4. Details of the method

In this section, we describe the details of the method, namely how the target mip-maps are constructed and what the prediction operator P looks like.

The down-sampling of the mip-maps can be performed by any form of averaging. As we saw in the previous chapter, the maximum absolute error does not depend on how the mip-maps look, as long as they contain valid values. However, the way the mip-maps are constructed affects the compression ratio. Moreover, various mip-map constructions produce different visual artifacts. In terms of the visual artifacts, the best way to down-sample a mip-map is to just average the four neighboring pixels [2n, 2n+1], [2n, 2n+1] at $\boldsymbol{L_{i+1}}$ into [n][n] at $\boldsymbol{L_i}$.

In the previous section, we made a simplification claiming that a decompressed mip-map $\boldsymbol{L^\bullet_{i+1}}$ is constructed from the previous $\boldsymbol{L^\bullet_i}$ in just one step (eq. 3.1). We did that in order to emphasize the fact, that $maxdev(\boldsymbol{L^\bullet_{i+1}}, \boldsymbol{L_{i+1}}) < \boldsymbol{D}$. In fact, three such steps happen. Nevertheless, the residuals are checked after each of these steps and all the predictions are made from the decompressed values, so the maximum error bound is still kept. So, when we construct the following decompressed mip-map, every pixel p from $\boldsymbol{L^\bullet_i}$ is substituted by four pixels in $\boldsymbol{L^\bullet_{i+1}}$ as shown in Fig. 4.1.

The first one of them, labeled a, is predicted directly from $\boldsymbol{L^\bullet_i}$ by a simple prediction operator $P_a(\boldsymbol{L^\bullet_i}) = p$. Following this, the residuals $\boldsymbol{E_a}$ and $\boldsymbol{E^\bullet_a}$ are computed according to the target value a_t in $\boldsymbol{L_{i+1}}$ and a is assigned the final value $a\bullet$ (eq. 4.1). It holds that $maxdev(a\bullet, a_t) \leq \boldsymbol{D}$.

$$\begin{aligned} \boldsymbol{E_a} &= a_t - p \\ \boldsymbol{E^\bullet_a} &= Q_D(\boldsymbol{E_a}) \\ a\bullet &= p + \boldsymbol{E^\bullet_a} \end{aligned} \tag{4.1}$$

The second one of them, labeled b, is predicted from the pixels $a\bullet$ in $\boldsymbol{L^\bullet_{i+1}}$ by the straight-oriented order 2 Neville interpolating filter (fig. 4.2), just like the border values are predicted in C-BDAM. A similar computation of residuals $\boldsymbol{E_b}$ and $\boldsymbol{E^\bullet_b}$ then follows according to the target value b_t from $\boldsymbol{L_{i+1}}$ and b is assigned the final value $b\bullet$ (eq. 4.2).

$$\boldsymbol{E_b} = b_t - P_b(\boldsymbol{L^\bullet_{i+1}})$$

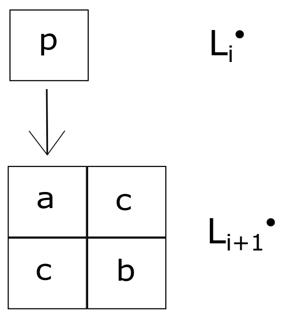

Figure 4.1: Substitution of a pixel p from $\boldsymbol{L^\bullet_i}$ by four children in $\boldsymbol{L^\bullet_{i+1}}$

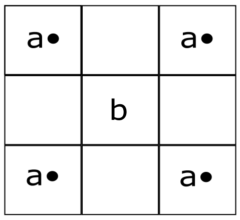

Figure 4.2: The prediction of b - $P_b(\boldsymbol{L}^{\bullet}_{i+1})$ - is the average of all the displayed $a\bullet$

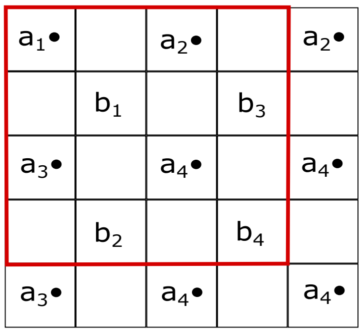

Figure 4.3: Handling of border cases in the computation of $P_b(\boldsymbol{L}^{\bullet}_{i+1})$ - the red line represents the border.

$$\begin{aligned} \boldsymbol{E}^{\bullet}_{b} &= Q_D(\boldsymbol{E}_{b}) \\ b\bullet &= P_b(\boldsymbol{L}^{\bullet}_{i+1}) + \boldsymbol{E}^{\bullet}_{b} \end{aligned} \tag{4.2}$$

The cases when the filter comes out of the image are handled by a specific mirror extension (fig. 4.3). Unlike C-BDAM, where the order 4 Neville filter is used for the interior values, in this method, the order 2 filter is used even for the interior values in order to increase the speed. As an additional optimization, the interpolation with the order 2 filter can be easily cached during horizontal traversal. Moreover, using the order 4 filter made the compression ratio slightly better - probably because it predicts hills and walleys more accurately, but made the quality of the reconstructed heightmap worse - it produced sharper artifacts on the borders of smooth gradient terrain blocks (Fig. 4.4) and near sharp terrain changes (Fig. 4.5).

The reason for these artifacts is that while the predictions are close enough to the real terrain (their quantized residuals are zeroes), the reconstructed values might be systematically above/under the terrain. But as soon as one prediction is a bit further from the terrain than those at the adjacent pixels, its residual is quantized to a non-zero value and the reconstructed value might flip to the other side of the terrain, producing a visual artifact. This often happens when smooth terrain is followed by a sharp change. The prediction operator might then predict different values near this change, as it reaches out to the area behind the change (Fig. 4.6, 4.7). This spike is then propagated to the next levels, but still within the maximum error bound. The mirroring at the borders produces such sharp changes, too, in a different, more complex way. However, some better form of mirroring might sort this out.

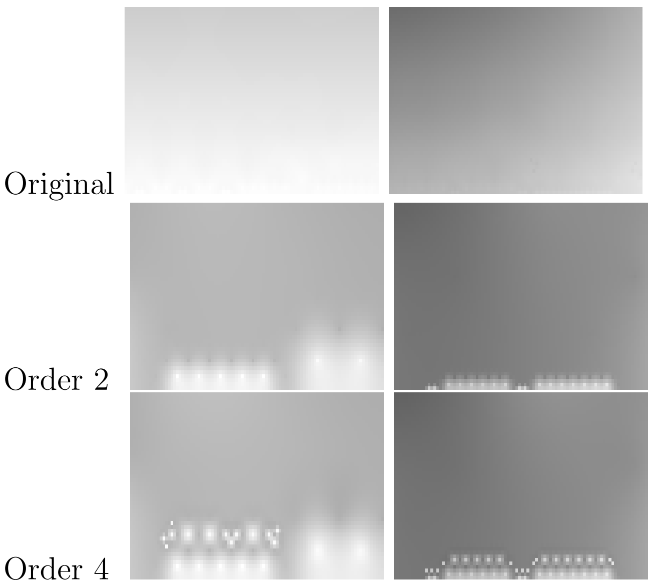

Figure 4.4: Two examples of different artifacts caused by order 2 and order 4 filters at the border of smooth gradient terrain - the first row shows the target heightmaps, the second row shows the same heightmaps compressed with the order 2 filter, the third row with the order 4 filter.

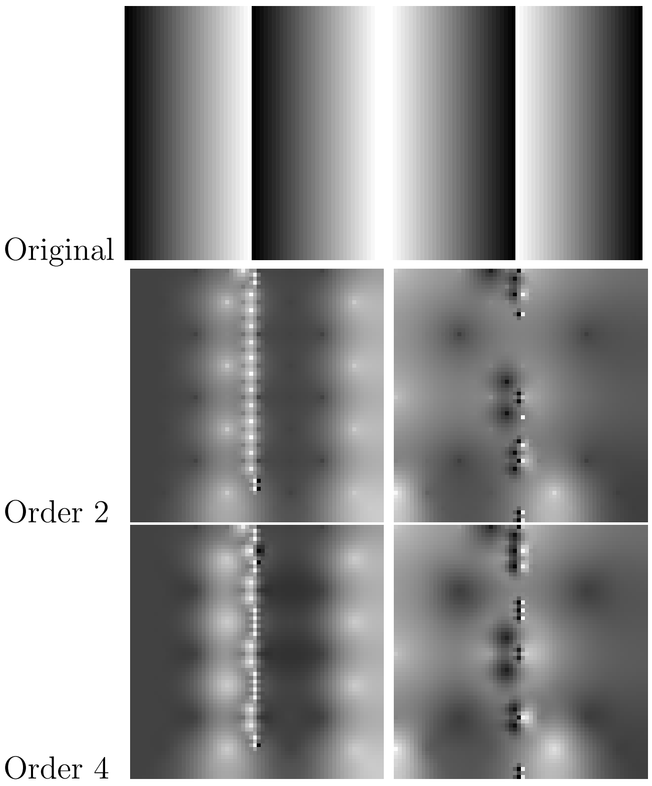

Figure 4.5: Two examples of different artifacts caused by order 2 and order 4 filters at a sharp terrain change - the first row shows the target heightmaps, the second row shows the same heightmaps compressed with the order 2 filter, the third row with the order 4 filter. The values in the original images range from 0 to 16 and the maximum deviation of the compression is 9.

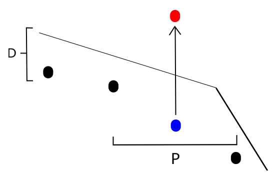

Figure 4.6: The illustration of how an artifact occurs - the black predictions are within the maximum-error bound D, so they are equal to the reconstructed values, but the blue one is not. Because a uniform quantizer is used, the blue prediction is shifted by $2D - 1$ to the top, creating an artifact.

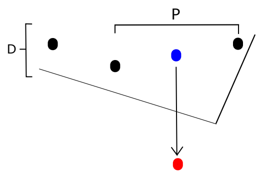

Figure 4.7: Another illustration of an artifact - the black predictions are within the maximum-error bound D, but the blue one is not. The blue prediction is shifted by $2D - 1$ to the bottom, creating an artifact.

While the prediction of the order 2 filter is the average of the four neighboring values, the prediction of the order 4 filter tends to differ from the neighboring values more, so this filter has the tendency to produce more disturbing artifacts.

The remaining pixels labeled c are predicted from the pixels $a\bullet$ and $b\bullet$ in $\boldsymbol{L^{\bullet}_{i+1}}$ by the diagonally-oriented order 2 Neville interpolating filter (fig. 4.8). The computation of residuals $\boldsymbol{E_c}$ and $\boldsymbol{E^{\bullet}_c}$ then follows according to the target value c_t from $\boldsymbol{L_{i+1}}$ and c is assigned the final value $c\bullet$ (eq. 4.2).

$$\begin{aligned} \boldsymbol{E_c} &= c_t - P_c(\boldsymbol{L^{\bullet}_{i+1}}) \\ \boldsymbol{E^{\bullet}_c} &= Q_D(\boldsymbol{E_c}) \\ c\bullet &= P_c(\boldsymbol{L^{\bullet}_{i+1}}) + \boldsymbol{E^{\bullet}_c} \end{aligned} \tag{4.3}$$

The cases when the filter comes out of the image are handled by a specific mirror extension (fig. 4.9). For the same reasons as in the prediction of b pixels, the order 2 filter is used for the prediction of all c pixels - both interior and

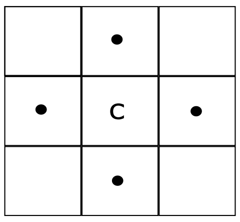

Figure 4.8: The prediction of c - $P_c(\boldsymbol{L^{\bullet}_{i+1}})$ - is the average of all the pixels marked with a dot - $\bullet$.

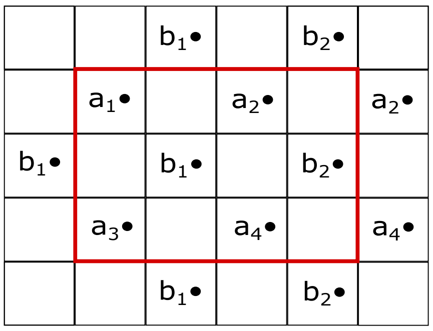

Figure 4.9: Handling of border cases in the computation of $P_c(\boldsymbol{L^{\bullet}_{i+1}})$ - the red line represents the border.

exterior ones. Similarly, the interpolation with such filter can be cached during diagonal traversal.

The residuals $\boldsymbol{E^{\bullet}_{a}}$, $\boldsymbol{E^{\bullet}_{b}}$ and $\boldsymbol{E^{\bullet}_{c}}$ are then encoded by an entropy codec and stored. The decompression is done in a similar manner with the only difference that the residuals are not computed anymore, but just decoded and read. So, we substitute every pixel from $\boldsymbol{L^{\bullet}_{i}}$ by four pixels in $\boldsymbol{L^{\bullet}_{i+1}}$, the value of which is computed in three passes of prediction followed by adding the read residual (the last lines of eq. 4.1, 4.2, 4.3).

5. Functional comparison to C-BDAM and wavelets

As we already mentioned, C-BDAM contains the whole rendering pipeline, whereas our method does not. However, it can be compared to C-BDAM in terms of how lifting is performed. As we already mentioned in the end of Section 2.1, C-BDAM omits a half of the samples while constructing a coarser LOD, whereas our method omits three fourths of the samples. This is spatially equivalent to two steps of lifting in C-BDAM (Fig. 2.1). The first step removes the pixels b and the second step removes the pixels c as seen in Fig. 4.1. Nevertheless, this equality is only spatial.

In our method, an analogy of the update operator of lifting is used to construct $\boldsymbol{L_i}$ from $\boldsymbol{L_{i+1}}$ (the averaging of four neighboring pixels - Sec. **??**). However, the lifting is not complete in our method as it does not contain the prediction operator - no residuals are computed there yet. In C-BDAM, also a prediction operator is used in the lifting to produce intermediate residuals. However, using just these residuals would not guarantee any maximum error bound, so C-BDAM makes another top-bottom pass to correct the residuals against the real values of samples produced in the first bottom-top pass. To make this correction fit into the original wavelet framework, several intricate computations need to be performed, including division, which is quite a large performance hit.

Our vision was that once it is needed to perform an extra top-bottom pass to correct the residuals so that the maximum error bound is guaranteed, it is not neccessary to compute any temporary values of the residuals during the lifting steps (the construction of the LOD pyramid). This is why we perform just an analogy of the update (the averaging of pixels) in the update-first scheme and let the following top-bottom pass compute suitable values of residuals. This is obviosly a major deviation from the wavelet scheme. In the top-bottom pass, we just predict the values in the finer LOD as accurately as possible, but these predictions have no linkage to the previous bottom-top pass, as they have not been used there at all. Then we directly compute the residuals with respect to the original values computed in the first bottom-top pass at the corresponding levels.

All in all, it can be said that the way the residuals are computed in this method is an extreme simplification of the way they are computed in C-BDAM. This way of computation does not even conform to the second-generation wavelet scheme - the lifting is not complete and the reconstruction is not the inverse of lifting. We think that without the residuals quantization or the per-level correction of residuals, respecting the wavelet scheme makes sense, as it ensures computational equivalency with the first-generation wavelets. However, in case the residuals need to be corrected at each level, we think that conforming to a wavelet scheme makes no sense, because this correction immediatelly destroys the mentioned equivalence - once a residual is cropped in order to get the resulting value closer to the actual data, it cannot be said that any of the following reconstruction is the inverse to the lifting performed before. Moreover, thanks to the mentioned deviation of C-BDAM from the classical update-first second-generation wavelet discussed in

Section 2.1, we question its computational equivalency with the first-generation wavelet even with no residuals quantization or cropping performed. Because of this, we think that the computations made in the second top-bottom pass can be optimized this way without any cost. Thus, this method would probably better be called wavelet-inspired than wavelet-based.

6. Results

This method has been applied in the real-time planet renderer mentioned in the introduction on height data of the whole Earth with the resolution of 90m (SRTM). Due to the redundancy of data in the applied LOD hierarchy, the size of the original data was 260GB. With the maximum error bound set to 5m, the size of the compressed data is 7GB, which yields the compression ratio of 37:1.

For a comparison, C-BDAM reached the compression ratio of 64:1 on the same dataset, but with the maximum error bound set to 16m. Thanks to the fact that the LOD hierarchy of C-BDAM contains no redundancy, the size of the original data was just 29GB and the size of the compressed data just 870MB. Under such circumstances, only the comparison in terms of the compression ratio is relevant.

In Fig. 6.1, you can see a part of a heightmap compressed by this method, together with the differences from the original.

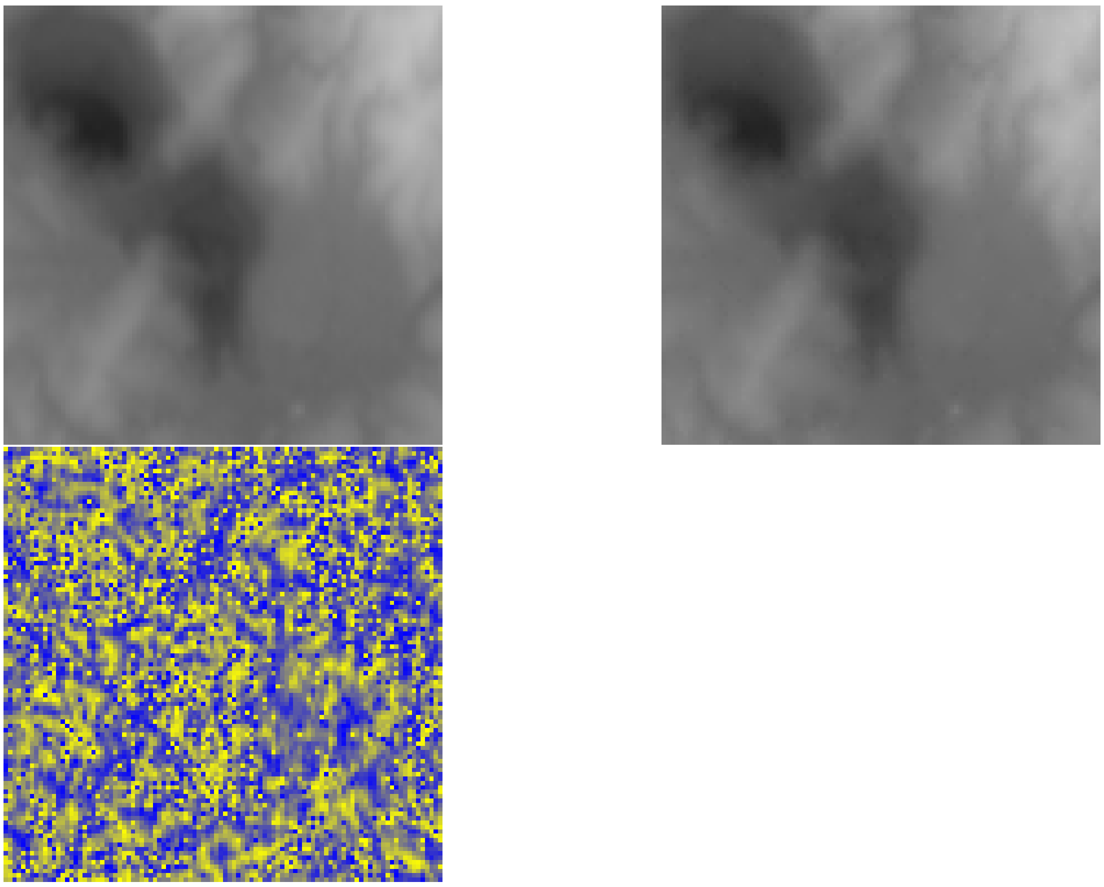

Figure 6.1: From the top to the bottom - the original terrain, the same terrain compressed with the maximum deviation of 5m, the difference between these two. The brighter the color, the greater the value. In the difference image, the yellow color means 4.5m, whereas the blue color means -4.5m.

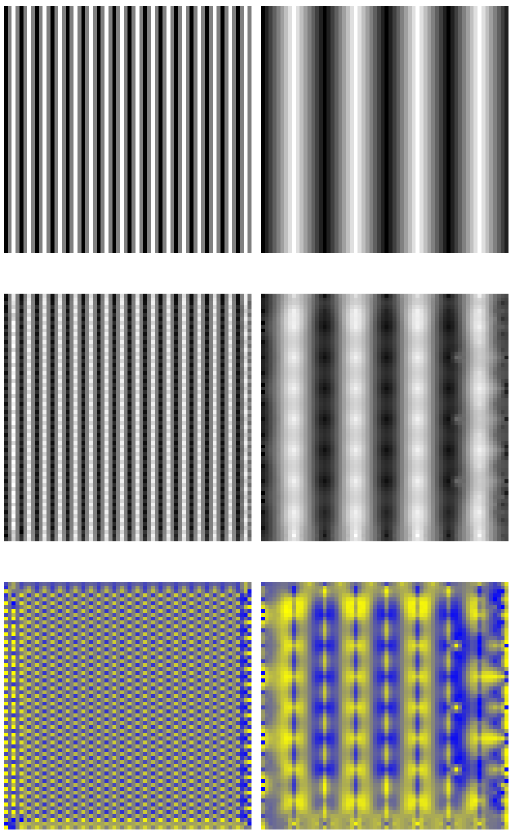

Figure 6.2: Two synthetic test images of size 64x64, each one containing spiky terrain with the heights ranging from -16 to 16. On the left, the longitude of spikes is 4, on the right, it is 16. From the top to the bottom - the original, compressed with the maximum deviation of 5, the difference between these two. The brighter the color, the greater the value. In the difference image, the yellow color means 4.5, whereas the blue color means -4.5.

7. Conclusion

In this paper, we described a heightmap compression method designed to be a plugin into an existing real-time planet renderer with its own rendering pipeline. The method proved to be convenient for the purpose, providing fast decompression (only about 1ms per block of data). Its compression ratio is comparable to C-BDAM, which is the method with the best compression ratio among the methods for the terrain compression, which guarantee a maximum error bound adjustable by the user.

Conclusion

[5]

Bibliography

[1] P. M. Bentley and J. T. E. McDonnell. Wavelet transforms: an introduction. *Electronics Communication Engineering Journal*, 6(4):175–186, Aug 1994.

[2] Paolo Cignoni, Fabio Ganovelli, Enrico Gobbetti, Fabio Marton, Federico Ponchio, and Roberto Scopigno. Planet–sized batched dynamic adaptive meshes (p-bdam). In *Proceedings IEEE Visualization*, pages 147–155, Conference held in Seattle, WA, USA, October 2003. IEEE Computer Society Press.

[3] Roger L. Claypoole, Geoffrey M. Davis, Wim Sweldens, and Richard G. Baraniuk. Nonlinear wavelet transforms for image coding via lifting. *IEEE Trans. Image Processing*, 12:1449–1459, 2003.

[4] Ingrid Daubechies and Wim Sweldens. Factoring wavelet transforms into lifting steps. *J. Fourier Anal. Appl*, 4:247–269, 1998.

[5] Enrico Gobbetti, Fabio Marton, Paolo Cignoni, Marco Di Benedetto, and Fabio Ganovelli. C-BDAM – compressed batched dynamic adaptive meshes for terrain rendering. *Computer Graphics Forum*, 25(3):333–342, September 2006. Proc. Eurographics 2006.

[6] Ricardo Olanda, Mariano Perez, Juan Manuel Orduna, and Silvia Rueda. Terrain data compression using wavelet-tiled pyramids for online 3d terrain visualization. *Int. J. Geogr. Inf. Sci.*, 28(2):407–425, February 2014.

[7] Renato Pajarola and Enrico Gobbetti. Survey of semi-regular multiresolution models for interactive terrain rendering. *The Visual Computer*, 23(8):583–605, 2007.

[8] Gregory K. Wallace. The jpeg still picture compression standard. *Communications of the ACM*, pages 30–44, 1991.

[9] Sehoon Yea and W.A. Pearlman. A wavelet-based two-stage near-lossless coder. *Image Processing, IEEE Transactions on*, 15(11):3488–3500, Nov 2006.

List of Tables

List of Abbreviations

Attachments