Adaptive Context Modeling for Efficient Image and Elevation Data Compression

Egor Yusov
Intel Corporation
Nizhny Novgorod, Russia
egor.yusov@gmail.com

Abstract

In this paper, a new wavelet zero tree-based image compression algorithm that is based on exploitation of smart adaptive context models is proposed. The models are derived from wavelet transform properties and are exploited to improve efficiency of arithmetic coding. They are intuitively clear and do not require any preliminary training. To author's knowledge, the proposed algorithm is comparable to or surpasses all previous zero-tree based encoders in terms of R-D performance. At the same time, the computational complexity of the algorithm remains low because it bypasses bit-plane coding and processes each coefficient in only one pass. The near-lossless algorithm extension that is based on lossy plus residual coding and provides a guaranteed maximum absolute error bound is presented. A new hierarchical compressed multiresolution terrain model designed for efficient elevation data storage and retrieval that exploits the presented compression technique is proposed. Special care is taken to guarantee seamless stitching of neighboring patches.

Keywords: *Compression, Wavelet Transform, Context Modeling, Terrain.*

1. INTRODUCTION

Rapid evolution of digital data acquisition technologies in the past years led to the exponential growth of digital content size. As a result, efficient compression techniques that reduce the storage requirements with no or minimal information loss have becoming more and more important. Image compression applications are well known and include digital camera, medical imaging, internet browsing, to name a few.

Compact representation of digital elevation data sets is another area where efficient compression techniques are required. This problem is especially important for such applications as geographical information systems, flight simulators, virtual environments, computer games etc. Satellite scans of a terrain region can contain billions elevation samples potentially requiring storage of up to terra bytes. Processing such huge uncompressed data sets is a very complex task, especially in the context of real-time terrain visualization, because the data size can exceed not only the main memory, but even the disk capacity.

In this paper, we present a new image compression method that improves the compression performance of previous algorithms such as SPIHT [SP96] and LTW [OM03] by using smart adaptive context modeling for more efficient arithmetic coding. At the same time, since algorithm is zero-tree based and bypasses bit-plane coding, its temporal complexity remains low. We applied the presented compression technique to construct hierarchical compressed multiresolution terrain representation that can be exploited in real-time terrain rendering applications. Special care is taken to assure seamless patch connection across borders.

2. RELATED WORKS

2.1 Wavelet-based image compression methods

During last years, wavelet-based image compression techniques have becoming more and more popular since they provide better compression performance and at the same time do not suffer from artifacts typical to the Discrete Cosine Transform – based image compression algorithms. All wavelet-based image coders have the same workflow. To remove correlation between neighboring pixels, at the first stage of the compression process, the image is transformed from spatial domain to a combined spatial-frequency wavelet domain [ABM92]. After the first step of wavelet decomposition, the image is transformed into the lower resolution representation (LL subband) and three detail subbands called: horizontal (LH), vertical (HL) and diagonal (LL) details (fig. 1). The same transform is further applied to the LL subband and so on.

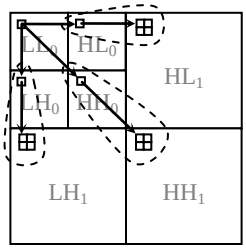

Fig. 1: Subbands of two-stage wavelet decomposition.

We denote $s_{i,j}^{(l)}$ to be the coefficient of subband s where $s \in \{ll, lh, hl, hh\}$ of the level l located at spatial position (i,j). In the literature, coefficient $s_{i,j}^{(l)}$ is called significant with respect to threshold τ, if its magnitude exceeds τ, i.e. $|s_{i,j}^{(l)}| \geq \tau$ and it is called insignificant otherwise.

Wavelet coefficients resulting from full wavelet decomposition are quantized at the next stage (here some information is lost) and are encoded. Due to hierarchical nature of wavelet transform, quantized wavelet coefficients can be organized into three quad trees growing through the LH, HL and HH subbands (see fig. 2).

The fundamentally new method for wavelet coefficient trees encoding was presented by J.Shapiro [Sha93] in his EZW algorithm. The main contribution of [Sha93] is introduction of a zero tree structure. Wavelet coefficients tree is called *zero tree* with respect to threshold τ if all its nodes are insignificant with respect to τ. If we define $T_{i,j}^{(l)} = \{(i_t, j_t, l_t)\}$ to be the set of positions of all nodes belonging to the tree rooted at node (i,j) on level l, including the root, then the node $s_{i,j}^{(l)}$ ($s \in \{lh, hl, hh\}$) is called the *zero*

tree root with respect to threshold τ, if it does not yet belong to the zero tree and $\forall (i_t, j_t, l_t) \in T_{i,j}^{(l)} \rightarrow | s_{i_t,j_t}^{l_t} | < \tau$.

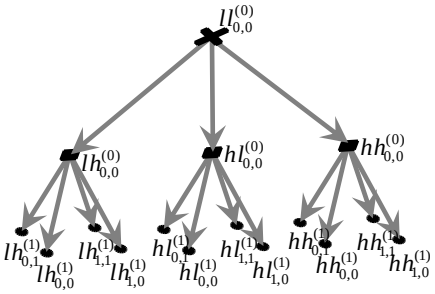

Fig. 2: Wavelet coefficients quad trees.

The main idea behind zero tree coding is the observation that if a coefficient is insignificant at a coarse scale in a multiresolution representation, then it is likely that its descendants at the finer scales are also insignificant. Thus single symbol is enough to encode all zero coefficients in a zero tree. One important feature of EZW algorithm is that it generates embedded bit stream, in which all encodings of the same image at lower bit rates are embedded in the beginning of the bit stream at higher bit rate.

EZW laid down the foundations for the new class of wavelet-based image compression techniques – the zero tree coders. The next method in this trend is called SPIHT and was presented by A. Said and W. Pearlman [SP96]. SPIHT is a highly refined version of EZW and due to smart set partitioning rules exploited it achieves remarkably higher compression ratios compared to EZW.

Though code embedding is a very useful feature, it significantly complicates the algorithm since each coefficients is reconstructed in several passes, one for each bit plane. However many applications (such as the digital camera or elevation data storage) do not require that feature and the data need to have pre-defined quality. So a number of algorithms tried to give up quality scalability in favor of higher execution speed.

One of such methods named SWEET was presented by J. Andrew [And97]. SWEET is solely based on energy clustering in transformed image in frequency and space and exploits block partitioning as an alternative to zero-tree coding to separate significant coefficients from large sets of insignificant ones. In contrast to SPIHT and EZW, SWEET does not produce embedded code: it outputs all bits of the coefficient magnitude up to some minimum bit plane number n_{min} as well as its sign as soon as the coefficient is identified as significant. Since SWEET avoids complicated list processing, it is much simpler and faster than SPIHT and at the same time it demonstrates comparable compression efficiency.

To improve temporal performance of the wavelet encoders/decoders, J. Oliver and M. Malumbres [OM03] proposed the algorithm that is based on a structure called wavelet coefficients lower tree (LTW), which is actually the zero-tree of pre-quantized wavelet coefficients with respect to threshold $2^{rplanes}$ (*rplanes* is the number of least significant bit planes to drop). The main speed improvement in LTW is achieved by eliminating bit-plane encoding (similar to SWEET). As a result, the compressed bit stream is not embedded, but it is resolution scalable, which means that the information corresponding to coarser image representation goes first. The encoding process consists of two passes. On the first pass, wavelet coefficients are labeled using special three labels: LOWER, ISOLATED_LOWER and LOWER_COMPONENT. The first two labels directly correspond to Zero Tree Root and Isolated Zero labels used by Shapiro in EZW [Sha93]. The last label indicates that the coefficient belongs to a lower tree. On the second pass, the coefficients are scanned from the lowest resolution to the highest resolution subbands, and for each coefficient its label as well as the number of bits required to represent its magnitude are arithmetically coded. They are then followed by the least significant bits representing the coefficient magnitude (MSB is omitted) and its sign. The main advantage of the LTW algorithm is its simplicity. LTW does not exploit lists and reconstructs coefficients in one pass which leads to significant speed improvement.

Previous zero-tree coders do not fully take advantage of adaptive context modeling. EZW exploits simple Markov conditioning based on significance of previous coefficient in scan and parent coefficient significances. SPIHT uses adaptive context modeling to jointly encode significance of 4 sibling coefficients. LTW also exploits simple arithmetic coding algorithm, however no details are presented in [OM03]. It was shown in other works [CO97], [Wu97] that exploitation of high-order context modeling can yield significant compression performance improvement. However, algorithms presented in [CO97], [Wu97] are not zero-tree based. Besides ECECOW algorithm [Wu97] requires extensive trainings to initialize its coding structures. In this work, we tried to take best from both worlds – combine adaptive context modeling with zero-tree coding to improve compression performance keeping at the same time the algorithmic complexity low.

2.2 Compressed multiresolution terrain models

Though large-scale terrain visualization has long history, only a few recent works concentrate on efficient elevation data compression methods. The geometry clipmaps approach [LH04] exploits regular grid pyramid data structure that enables applying the lossy image compression technique [Mal00] to the terrain height map. However, this method cannot provide a guaranteed error bound, which becomes especially apparent on high-variation terrains such as the Grand Canyon. Another method that utilizes terrain elevation data compression is presented by E. Gobbetti, et al., [GMC+06] and is called C-BDAM. It exploits a wavelet-based two stage near-lossless compression technique presented by S. Yea and W. Pearlman [YP06]. A problem of seamless stitching neighboring patches is not covered by C-BDAM. Another method presented by C. Dick, et al. [DSW09], mainly focuses on compressing adaptive triangulation in a way that enables GPU-based decompression. The method achieves a moderate compression factor of 8-9. Elevation data compression techniques are not considered by C. Dick, et al.

The remained of this paper is organized as follows. In section 3, our image compression technique is described. In section 4, we present compressed multiresolution terrain representation based on the proposed image compression algorithm. Section 5 presents experimental results. Section 6 concludes the paper.

3. COMPRESSION ALGORITHM DESCRIPTION

The compression method we developed belongs to the class of zero-tree coders and extends the ideas of EZW [Sha93], SPIHT [SP96] and LTW [OM03] algorithms. As in EZW, the quantized wavelet coefficients are scanned from the lowest frequency subband to the highest frequency in our algorithm, and in each subband, coefficients are scanned in zigzag order. While EZW and SPIHT perform multiple bit-plane passes, our method encodes

each coefficient in only one pass. It encodes each coefficient magnitude using at most n_{bp} bits. It thus is similar to SWEET [And97] and LTW [OM03] algorithms, which discard a fixed number of least significant bits in quantized wavelet coefficients and generate non-embedded but resolution scalable bit stream.

3.1 Quantization

We use a uniform quantizer with a dead zone (fig. 3). The quantization step q is determined as $q = M / 2^{n_{bp}}$ where $M = \max_{s\in\{lh,hl,hh\},i,j,l} |s_{i,j}^{(l)}|$ is the maximum wavelet coefficient magnitude of all detail subbands (LH, HL and HH) across all scales and n_{bp} is the maximum number of bits allotted to represent the coefficient magnitudes. The quantizer divides the range of all magnitudes [0, M] onto $N = 2^{n_{bp}}$ uniform quantization bins B_n such that $B_n = \{s : nq \le |s| < (n+1)q\}$.

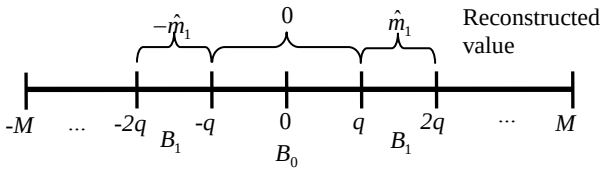

Fig. 3: Uniform quantizer with a dead zone.

We denote $n_B(s) = \lfloor |s| / q \rfloor$ to be the number of the bin which coefficient s falls into ($\lfloor \cdot \rfloor$ denotes the integer part). We will refer to this number later as quantized magnitude. Binary representation of $n_B(s)$ can be treated as a sequence of n_{bp} binary decisions where the first bit indicates whether the coefficient magnitude falls into the $[0, M/2)$ range (0 bit) or into the $[M/2, M]$ range (1 bit). The next bit refines the uncertainty interval to length $M/4$ (00: $[0, M/4)$, 01: $[M/4, M/2)$, 10: $[M/2, 3M/4)$, 11: $[3M/4, M]$) and so on. Magnitudes of all coefficients $s_{i,j}^{(l)}$ falling into the same quantization bin B_n are reconstructed equally as $\hat{m}_n$ and at the decoder, the coefficients are restored as follows:

$$\hat{s}_{i,j}^{(l)} = sign(s_{i,j}^{(l)}) * \hat{m}_{i,j}^{(l)}$$

where $sign(s_{i,j}^{(l)}) = +1$ if $s_{i,j}^{(l)} > 0$ and $sign(s_{i,j}^{(l)}) = -1$ if $s_{i,j}^{(l)} < 0$; $\hat{m}_{i,j}^{(l)}$ is the reconstruction level for the quantization bin which $s_{i,j}^{(l)}$ falls into (with the number $n_B(s_{i,j}^{(l)})$). The question is what value to use as a reproduction level $\hat{m}_n$ for the bin B_n. If the probability distribution function (PDF) $f(t)$ of the coefficient magnitudes was known, the optimal reproduction levels for each bin would be placed at the centroid of the distribution for that bin:

$$\hat{m}_n = E(nq \le |s_{i,j}^{(l)}| < (n+1)q) = \int_{nq}^{(n+1)q} t f(t) dt \Big/ \int_{nq}^{(n+1)q} f(t) dt \qquad (1)$$

Exact distribution is unknown; however it is usually assumed that wavelet coefficients have Laplacian distribution with zero mean, that is the coefficient magnitudes distribution can be well approximated by the following PDF:

$$f(t) = \alpha e^{-\alpha t} \qquad (2)$$

Given that assumption, the reconstruction levels $\hat{m}_n$ can be calculated as follows:

$$\hat{m}_n = \int_{nq}^{(n+1)q} \alpha t e^{\alpha t} dt \Big/ \int_{nq}^{(n+1)q} \alpha e^{\alpha t} dt$$

Reducing this formula gives the following equation for $\hat{m}_n$:

$$\hat{m}_n = nq + \frac{1}{\alpha} - \frac{q}{e^{q\alpha} - 1} \qquad (3)$$

The unknown parameter α in (3) can be estimated based on the magnitudes of wavelet coefficients. This could be done using one of the methods from mathematical statistics. Since our goal is to minimize reconstructed image error, we exploit different approach. It is difficult to find exact α that minimizes the distortion in image space without performing extensive calculations. We instead calculate α such that it minimizes the distortion in wavelet domain. Due to bi-orthogonality of wavelet transform, this method also gives close to optimal solution in image space. We thus minimize the following error function that gives mean square error in wavelet domain:

$$D(\alpha) = \sum_{n=1}^{N-1} \sum_{s\in B_n} (|s| - \hat{m}_n)^2 \to \min$$

It can be seen from (3) that all reproduction levels $\hat{m}_n$ are equally shifted from the quantization bin B_n lower bound nq:

$$\hat{m}_n = nq + \beta \text{ where} \qquad (4)$$

$$\beta = \frac{1}{\alpha} - \frac{q}{e^{\alpha q} - 1}. \qquad (5)$$

We thus need to find the optimal shift β that minimizes the following function:

$$D(\beta) = \sum_{n=1}^{N-1} \sum_{s\in B_n} (|s| - (nq + \beta))^2 \to \min$$

The minimum of this function can be derived from the root of the following equation: $\frac{dD(\beta)}{d\beta} = 0$

which gives the following formula for optimal β:

$$\beta = \frac{1}{N_s} \sum_{n=1}^{N-1} \sum_{s\in B_n} (|s| - nq) \qquad (6)$$

Where N_s is the total number of significant coefficients. It follows from (6) that the optimal shift β for reproduction levels is the average magnitude shift from the quantization bins' lower bounds. This is intuitively reasonable result.

Given β, the distribution parameter α can be calculated by reverting equation (5), however in fact, we do not need α since we only interested in optimal shift β. Thus $\beta^{(l)}$ parameters are calculated separately for all decomposition levels in our algorithm and are encoded as side information. Since β falls in the range $[0, q]$, the following symbol is encoded using $nbits_\beta$ bits:

$$\tilde{\beta}^{(l)} = \left\lfloor \beta^{(l)} / q \cdot 2^{nbits_\beta} \right\rfloor.$$

At decoder, the parameter is reconstructed as follows:

$$\hat{\beta}^{(l)} = \left\lfloor (\tilde{\beta}^{(l)} + 0.5) / 2^{nbits_\beta} \cdot q \right\rfloor.$$

We use 7 bits ($nbits_\beta = 7$) for encoding $\beta^{(l)}$.

Wavelet coefficients of the lowest frequency LL subband in our algorithm are quantized and are arithmetically encoded using separate model for each bit position. Since large values are less possible, arithmetic coding reduces the compressed data size.

The compression performance of different wavelet-based algorithms is primarily determined by how efficiently the quantized coefficients are encoded. In the next subsections, we will describe our adaptive context models that improve compression efficiency of arithmetic coding. To distinguish our method from others, we call it ACMZW (Adaptive Context Modeling Zero-tree Wavelet coder).

3.2 Utilizing adaptive context modeling to predict the coefficient magnitude

Similar to LTW, our method encodes number of bits $nbits(n_B(s))$ required to represent each significant coefficient *s* quantized magnitude $n_B(s)$ using arithmetic coding [WNC87] and then transmits the magnitude bits followed by the sign. To exploit information carried by already encoded neighboring coefficients, we utilize adaptive context modeling. For this purpose, we first predict the reconstructed coefficient magnitude. Wavelet transform localize energy in both frequency and spatial domains: WT coefficients of similar magnitudes statistically cluster in frequency subbands and in spatial locations. As a result, the reconstructed coefficient $\hat{s}_{i,j}^{(l)}$ estimated magnitude can be well derived from the magnitudes of the coefficients in the context, containing four neighbors (one to the left ($\hat{s}_{i-1,j}^{(l)}$)) and three from above ($\hat{s}_{i-1,j-1}^{(l)}$, $\hat{s}_{i,j-1}^{(l)}$ and $\hat{s}_{i+1,j-1}^{(l)}$)) and one direct ancestor ($\hat{s}_{i/2,j/2}^{(l-1)}$) (see fig. 4).

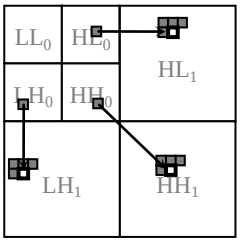

Fig. 4: 5-coefficients contexts used to estimate the coefficient's magnitude.

We use the following expression to calculate the expected coefficient magnitude $\hat{\hat{m}}_{i,j}^{(l)}$:

$$\hat{\hat{m}}_{i,j}^{(l)} = \sqrt{w_0(\hat{m}_{i-1,j}^{(l)})^2 + w_1(\hat{m}_{i-1,j-1}^{(l)})^2 + w_2(\hat{m}_{i,j-1}^{(l)})^2 + w_3(\hat{m}_{i+1,j-1}^{(l)})^2} + w_4\sqrt{(\hat{m}_{i/2,j/2}^{(l-1)})^2} \quad (7)$$

In the expression above, the first summand represents the energy localization property in space: it is expected that neighboring coefficients will have a similar energy. The second summand represents the energy localization property in frequency: the ancestor's energy is distributed among its descendants.

Weights w_i can be optimized for each subband for particular image. However, this would require additional computations and a lot of side information to be sent, so we use constant weights instead, which are optimized for a test set of images. Since LH subband exhibits predominantly horizontal structures, we use the following coefficients for this subband:

$(w_0, w_1, w_2, w_3, w_4) = (0.4,\ 0.15,\ 0.3,\ 0.15,\ 0.125)$.

HL subband exhibits predominantly vertical structures, so we apply the following weights for this subband:

$(w_0, w_1, w_2, w_3, w_4) = (0.3,\ 0.15,\ 0.4,\ 0.15,\ 0.125)$.

HH subband does not exhibit explicit structures, and we exploit the following weights:

$(w_0, w_1, w_2, w_3, w_4) = (0.25,\ 0.25,\ 0.25,\ 0.25,\ 0.125)$.

Due to the zig-zag scanning order, all coefficients in the context are evaluated first, so $\hat{\hat{m}}_{i,j}^{(l)}$ is always properly calculated. The number of bits $nbits(n_B(\hat{\hat{m}}_{i,j}^{(l)}))$ required to represent the quantized predicted magnitude $n_B(\hat{\hat{m}}_{i,j}^{(l)})$ gives the adaptive context model number to be used in arithmetic coder. To further improve compression performance, we encode the maximum number of bits required in each level of wavelet decomposition and use separate context sets (consisting of *max_bits(level)* models) for each level. After the number of bits $nbits(n_B(s_{i,j}^{(l)}))$ required to represent the exact quantization bin number $n_B(s_{i,j}^{(l)})$ is encoded with the appropriate model (which is given by $nbits(n_B(\hat{\hat{m}}_{i,j}^{(l)}))$ and decomposition level), all the coefficient magnitude refinement bits (which are the bits in binary representation of the $n_B(s_{i,j}^{(l)})$) excepting the most significant one are transmitted. We determined that exploiting arithmetic coding for refinement bits also improves compression performance. This is achieved by using a separate arithmetic model for each bit position and for each number of bits required to represent bin number.

3.3 Utilizing adaptive context modeling to predict insignificant coefficient sets

While the LTW algorithm exploits the EZW-like coding style to encode large sets of insignificant coefficients, with additional symbols corresponding to degree-1 zero tree, we adopt the more advanced SPIHT-style coding method. Since SPIHT is a degree-two zero tree coder (see [CP07]), while LTW is a degree-one zero tree coder, the exploitation of SPIHT-style coding gives additional benefit [CP07]. In [SP96], the set of coordinates of all offsprings (direct descendants) of the node is called *set type A*. The set of all descendants of the node excepting its offsprings is called *set type B*. Set is defined to be significant if it contains at least one significant coefficient. SPIHT defines smart set partitioning rules that are used to efficiently separate significant and insignificant coefficients (see [SP96] for more details).

SPIHT jointly encodes the significance of four neighboring sets of type A and uses conditioning on the significance of set type A to encode the significance of set type B. In our algorithm, we implemented different and more efficient approach. As with magnitudes, we predict the significance of sets of type A and B based on already encoded information (fig. 5) and taking into account space and frequency localization properties of wavelet transform.

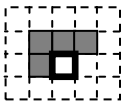

Fig. 5: 4-coefficients context used to predict set type A and set type B significance.

Due to the space localization property, it can be expected that the more significant neighboring sets of type A (B) the coefficient has, the more probable its set type A (B) is significant. Due to the frequency localization property, it is expected that the more energy is concentrated in this spatial location, the more probable the energy spread to higher frequency and as a result the coefficient's set type A (B) is significant. Combining these two considerations, we derive the following expressions to calculate context model number to encode the set type A (B) significance:

$$CM_A = A^{(l)}_{i-1,j} + A^{(l)}_{i-1,j-1} + A^{(l)}_{i,j-1} + A^{(l)}_{i+1,j-1} + (4,\ if\ |s^{(l)}_{i,j}| > 2q)$$

$$CM_B = B^{(l)}_{i-1,j} + B^{(l)}_{i-1,j-1} + B^{(l)}_{i,j-1} + B^{(l)}_{i+1,j-1} + (4,\ if\ |s^{(l)}_{i,j}| > q)$$

where $A^{(l)}_{i,j} = 1$ ($B^{(l)}_{i,j} = 1$) if set type A (B) is significant and $A^{(l)}_{i,j} = 0$ ($B^{(l)}_{i,j} = 0$) otherwise. Thresholds $2q$ and q were found out empirically for test set of images. The presented equations have the meaning that the greater context model number, the more probable the set is significant.

3.4 Encoding coefficient signs

In [BP01], the sign/significance information is encoded using 3^m symbols for m yet insignificant coefficients in a group of 2x2 sibling coefficients. We figured out that it is more efficient to encode sign separately using one out of 27 context models for each subband. The context model number is defined depending on the sign and significance of three already encoded neighbors in the context shown in fig. 6. There can be 3 possible states for each neighbor: positive, negative or insignificant (zero), which gives 27 possible combinations for three coefficients. Since signs tend to produce different patterns in each subband, we use 3 separate context model sets for each subband (LH, HL and HH).

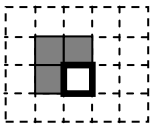

Fig. 6: 3-coefficients context used to encode the sign.

The close approach was implemented in ECECOW algorithm [Wu97], however contexts of ECECOW contain much more coefficients and thus they require much more data to provide reliable probability estimations.

3.5 Lossy plus residuals coding that guarantees the specified tolerance of the reconstructed data

In many applications, such as medical imaging, the decompressed image must satisfy some pre-defined error tolerance in L^∞ sense. For terrain compression, the L^2 error bound is also clearly inappropriate. The L^∞ error bound can not be guaranteed in wavelet domain, so many methods exploit lossy plus residual coding approach [AMC98]. In this scheme, the data is first compressed using the lossy coder and it is then supplemented by the encoded quantized residuals that guarantee a given L^∞ error bound δ (refer to [AMC98, YP06] for more details). For this purpose, the residual layer R which is the difference between the decompressed image $\hat{I}$ and original image I is calculated: $R = I - \hat{I}$. Since pixels of 8-bit images can only be integer values, the residual layer is quantized using the following rule [AMC98]:

$$\tilde{r}_{i,j} = \lfloor (|r_{i,j}| + \delta)/(2\delta + 1) \rfloor \quad (8)$$

At decompression, the residuals are reconstructed as follows: $\hat{r}_{i,j} = sign(r_{i,j}) \cdot \tilde{r}_{i,j} \cdot (2\delta + 1)$. Note that before calculating the residual layer, the decompressed image $\hat{I}$ must be rounded to integer values. The resulting decompressed image is then obtained as $\hat{I} + \hat{R}$ and differs from original image I by at most δ.

Quantized residuals $\tilde{r}_{i,j}$ as well as their signs $sign(r_{i,j})$ are arithmetically encoded in our method using adaptive context modeling. We first estimate average variation in the lossy decompressed image around the residual using the following equation: $V_{i,j} = \sqrt{\frac{1}{8}\sum_{m,n=-1}^{1} (\hat{i}_{i+m,j+n} - \hat{i}_{i,j})^2}$. After that we exploit the same context as that used for encoding set type A/B significance (fig. 5) to take into account already encoded neighboring residuals. Finally, the context model number is determined as follows:

$$CM_{r_{i,j}} = \lfloor V_{i,j}/\delta \rfloor + \tilde{r}_{i-1,j} + \tilde{r}_{i-1,j-1} + \tilde{r}_{i,j-1} + \tilde{r}_{i+1,j-1}.$$

Note that [AMC98] exploits much more complex method to determine context model number; however our method being much simpler provides comparable compression results as shown in section 5. Residual signs are encoded in the same way as described in section 3.4. In our current implementation, the optimal lossy bitrate is determined iteratively, however it can be estimated during the encoding process as described in [YP06].

4. COMPRESSED MULTIRESOLUTION TERRAIN MODEL

4.1 Hierarchically encoding height map using ACMZW algorithm

In this section, we describe compressed multiresolution terrain representation that exploits the presented compression method. At first, the initial height map is prefiltered into a mipmap pyramid much like it is done in the geometry clipmaps framework [LH04] and our earlier approach [YT08]. Each level of the pyramid has two times fewer samples in each direction and a two times longer sample spacing interval and thus approximates the original height map with diminishing accuracy. For pyramid construction, we use normalized Daubechies 9/7 low-pass wavelet filter.

At the next stage, the patch quad tree data structure is constructed by subdividing each level of the pyramid into square blocks having an equal number of samples (64x64, 128x128, 256x256 etc.) as shown in fig. 7. The resulting hierarchy is compressed in a top-down order starting from the coarsest resolution (patch at level 0). During that process, for each patch, special refinement information is encoded that enables reconstructing patch descendant's height maps. For this purpose, the difference layer $D = \{d_{i,j}\}^{2n \times 2n}$ is calculated as the discrepancy between the four child patches' predicted height maps ($H^P = \{h^P_{i,j}\}^{2n \times 2n}$) and their exact height maps ($H = \{h_{i,j}\}^{2n \times 2n}$) as shown in fig. 8: $D = H - H^P$.

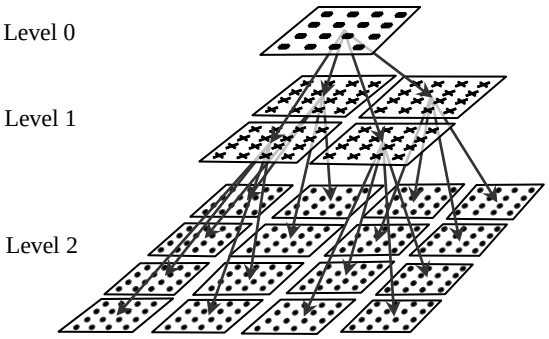

Fig. 7: Patch quad tree data structure.

The predicted height maps H^P are obtained by applying Daubechies 9/7 synthesis low-pass wavelet filter to the parent patch height map. For the coarsest resolution patch (level 0), the predicted height map is defined to be zero. An important aspect here is that the parent patch height map is extracted from a compressed representation instead of using the exact data from the multiresolution representation. This eliminates error propagation from coarser to finer levels.

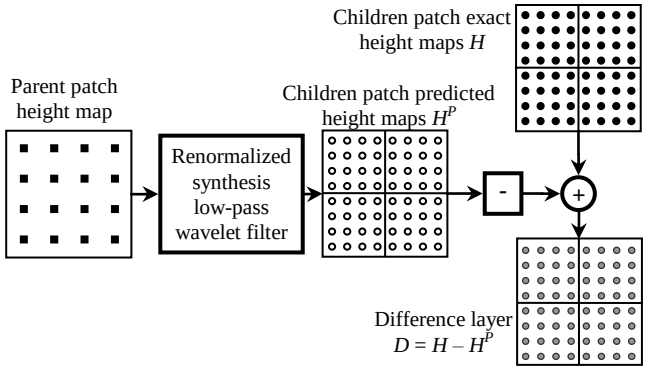

Fig. 8: Calculating the difference layer.

The difference layer D for each group of four sibling patches is compressed using a two-stage lossy plus residuals coding scheme described in section 3.5 with another residual quantization rules.

At first, we attempted to compress levels with exponentially increasing error tolerance such that patches at the finest level are compressed with a user-defined world space error threshold δ, patches at the next-coarser level are compressed with threshold 2δ and so on. However, later we found out that it is more efficient to compress each level of the pyramid using the same tolerance δ. In the latter case, the prediction turns out to be more accurate and as a result, the difference layer is compressed more efficiently. Besides, we use another quantization rule when compressing patches at all coarser levels excepting the finest one: $\tilde{r}_{i,j} = \lfloor | r_{i,j} | / \delta \rfloor$. Residual reconstruction is performed as follows:

$$\hat{r}_{i,j} = \begin{cases} sign(r_{i,j}) \cdot (\tilde{r}_{i,j} + 0.5) \cdot \delta, & \tilde{r}_{i,j} \neq 0 \\ 0, & \tilde{r}_{i,j} = 0 \end{cases}$$

This quantizer has the dead zone of length 2δ around zero, but the rest quantization intervals are two times shorter. This also results in a more precise prediction and leads to more compact representation. Besides, since elevation data samples are not integer values, we use more accurate quantizer for the finest level than that given by (8):

$$\tilde{r}_{i,j} = \lfloor (| r_{i,j} | + \delta) / (2\delta + p) \rfloor \quad (9)$$

where p is the height map samples representation precision.

We found out that exploiting two-stage compression yields better results compared to one-stage encoding, where the difference layer is directly compressed.

For each patch in the compressed hierarchy, we store its upper approximation error bound. We recursively calculate it as the sum of three terms: 1) the maximum distance between patch's interpolated height map and the vertices at the next finer resolution; 2) the maximum reconstruction error of the patch descendants and 3) elevation data reconstruction error. This value is used to construct adaptive view-dependent block-based terrain approximation.

4.2 Seamlessly stitching neighboring patches

None of previous elevation data compression techniques take care of seamless elevation data connection and normal map generation. In our representation, neighboring patches have common elevation data samples and each patch is "responsible" for seamlessly stitching transition with its right and top neighbors. The two-stage compression technique presented in section 3.5 guarantees that each elevation data sample's reconstruction error is within the predefined threshold. However, it is not guaranteed, that the common samples of neighboring patches both being within the tolerance are reconstructed equally. To cope with this problem, we introduced special boundary area around four sibling patches' height maps. We call this area "matching boundary" (see fig. 9).

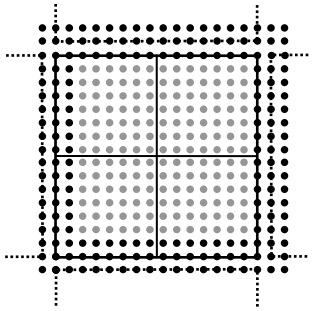

Fig. 9: Matching boundary.

Matching boundary consists of several rings (3 in example). Note that it is sufficient to have one ring to assure seamless geometric stitching, however for seamless normal map stitching more rings are required. Each ring is compressed separately without wavelet transform using the following algorithm. The ring is treated as a one dimensional sequence of elevations h_t, which are quantized using (9). For each quantized elevation $\tilde{h}_t$ (excepting $\tilde{h}_0$), the difference $d_t = \tilde{h}_t - \tilde{h}_{t-1}$ is encoded using arithmetic coding. We use simple context modeling based on previous quantized difference d_{t-1} to improve compression performance. Signs are also arithmetically encoded using conditioning on previous sign.

Matching boundary consisting of 3 rings assures that the normal maps calculated for neighboring patches perfectly connect.

In terrain rendering systems, sometimes it is required to generate elevation data finer than the original data set define. In our test system, we implemented procedural height map generation algorithm that exploits local surface elevation and slope. The presented matching boundary also assures that procedurally generated height maps for neighboring patches perfectly match.

5. EXPERIMENTAL RESULTS AND DISCUSSION

5.1 Image compression performance

In our tests, we used Daubechies 9/7 bi-orthogonal wavelet filters [ABM92] with 5 decomposition levels. The compression performance gain obtained by applying each modification described in sections 3.2-3.4 for Lena image is presented in Table 1. The table shows compression ratio of the not optimized algorithm and compressed bit stream size after applying each modification as a fraction (in percents) of the bit stream size generated by the basic algorithm.

# bit planes	PSNR, dB	Basic (no opts), bpp	Magn. predict. (sec. 3.2)	Set type A/B signif. prediction (sec. 3.3)	Sign coding (sec. 3.4)	All opts
6	29.936	0.098	96.4%	98.0%	98.9%	93.2%
7	33.187	0.212	95.6%	97.9%	98.2%	91.7%
8	36.343	0.436	94.6%	98.0%	97.9%	90.5%
9	39.652	0.931	92.8%	98.3%	98.2%	89.3%
10	44.413	1.946	92.3%	99.3%	98.9%	90.5%

Table 1: Compression ratio improvements resulting from exploiting presented context models for Lena image.

The results of compressing standard test images, Lena and Barbara in comparison with LTW [OM03] and SPIHT [SP96] methods are presented in Tables 2 and 3. To have fair comparison, we also implemented LTW and SPIHT-AC methods as it is described in original papers. The exact bit rate in our method is achieved by tuning the *M* value and thus by adjusting the quantization step *q*.

codec\ rate	LTW [OM03]	LTW (Ours)	SPIHT-AC [SP96]	ACMZW
0.125	31.27	31.01	31.10	31.20
0.25	34.31	33.98	34.11	34.28
0.5	37.35	37.07	37.21	**37.39**
1.0	40.50	40.13	40.41	**40.55**
2.0	45.46	44.82	45.07	**45.67**

Table 2: PSNR (dB) values at various rates for Lena image.

codec\ rate	LTW [OM03]	LTW (Ours)	SPIHT-AC [SP96]	ACMZW
0.125	25.52	25.17	25.23	**25.55**
0.25	28.33	27.89	27.83	**28.33**
0.5	31.78	31.34	31.45	**31.9**
1.0	35.88	35.41	35.69	**35.94**
2.0	40.74	39.97	40.40	**40.9**

Table 3: PSNR (dB) values at various rates for Barbara image.

Our implementation of the LTW encoder based on exact formal description of the algorithm presented in [OM03] incorporates all improvements mentioned in [OM03] and a few our modifications that also improve the compression ratio. In Tables 2 and 3, we presented the best results we obtained for our LTW implementation. However despite all modifications, our implementation of the LTW method demonstrated significantly poorer compression performance compared to the results reported in [OM03] (up to 0.77 dB below the reported results (see tables 2 and 3) and 0.43 dB below in average). This is not an issue of the implementation since LTW algorithm is rather simple from one hand, and our implementation of the SPIHT encoder demonstrates exactly the same compression ratios as stated in [SP96], from the other. The lack between real compression performance and reported in [OM03] is probably the result of adaptive context modeling implemented in real LTW which is only mentioned in [OM03] but no details are presented in the paper. In this work, we thoroughly described adaptive context models we used in our algorithm that enables us to improve the compression performance and achieve the same or higher compression rates compared to LTW. As tables 2 and 3 show, our method demonstrates better compression performance (up to **0.6 dB**) on all bitrates compared to SPIHT-AC [SP96]. It also demonstrates comparable or better (up to **0.21 dB**) performance than reported in [OM03] and significantly better performance than our exact implementation of LTW.

Table 4 shows compression performance for Lena image for different maximum absolute error thresholds compared to other methods (the data is taken from [YP06]). These results are obtained using quantizer (8).

Method\abs err	1	2	4	6	7
JPEG-LS (bpp)	2.72	2.09	1.54	1.24	1.14
CALIC (bpp)	2.59	1.95	1.29	0.96	0.85
[ACM98] (bpp)	2.69	2.02	1.28	0.86	0.73
ACMZW (bpp)	2.68	2.02	1.30	0.88	0.75
lossy + residual	0.47+2.21	0.39+1.63	0.45+0.85	0.39+0.49	0.39+0.36

Table 4: Comparing compression performance for various maximum absolute errors with other methods for Lena image.

As table 4 shows, though we used much simpler context to compress residuals, our method demonstrates compression performance comparable to [ACM98] and comparable to or superior than other methods such as JPEG-LS [WSS00] and CALIC [WM97]. Our method also demonstrates **1.39** to **2.9** times higher compression ratios compared to FBTR method [Zhe04].

5.2 Elevation data compression performance

To test the performance of our terrain compression algorithm, we used two different elevation data sets. The first data set is the Puget Sound being 16384x16384 in size and sampled at 10 m spacing. This data set is used as the common benchmark and is available at [PS]. The second one is the Grand Canyon being 8192 x 8192 in size and sampled at 30 m spacing. The elevation data precision is 0.1 m, so we use p=0.1 in (9). The compression results for patch size 256x256 and 64x64 and matching boundary width 3 are presented in Tables 5 and 6. The compression and run-time experiments were done on a workstation with the following hardware configuration: CPU: Intel Core i7 @2.67 GHz (4 cores with 2 hyper threads each); 6.0 GB RAM; GeForce GTX275 graphics card.

Tolerance (m)	Puget 256M			Grand Canyon 64M		
	rms (m)	Compr. rate (bps)	Compr. time (s)	rms (m)	Compr. rate (bps)	Compr. time (s)
1	0.449	0.636	294	0.539	1.963	124
3	1.120	0.287	235	1.445	1.031	80
10	3.099	0.116	223	3.962	0.414	61

Table 5: Compressing Puget 256M and Grand Canyon 64M data sets with patch size 256x256 and matching boundary size 3.

Tolerance (m)	Puget 256M			Grand Canyon 64M		
	rms (m)	Compr. rate (bps)	Compr. time (s)	rms (m)	Compr. rate (bps)	Compr. time (s)
1	0.423	0.837	255	0.531	2.282	93
3	1.077	0.431	229	1.407	1.237	68
10	2.931	0.21	221	3.898	0.54	56

Table 6: Compressing Puget 256M and Grand Canyon 64M data sets with patch size 64x64 and matching boundary size 3.

The compression method presented is up to 2.5 times more efficient than our previous algorithm [YT08]. The same Puget Sound data set was compressed by C-BDAM with 1 m max error to 0.61 bps, as reported in [GMC+06]. Without matching boundary and using patch size 256x256 our algorithm achieved 0.588 bps, which proves high compression potential of the proposed method.

During an interactive flight over the Grand Canyon and Puget Sound data sets with procedural terrain surface texturing and atmospheric effects rendered at 1920x1200 resolution, the frame rates never dropped below 120 fps. The decompression is performed significantly faster (less than 0.1 s to decompress one patch) than compression and in conjunction with the asynchronous rendering algorithm, it provides steady frame rates. These results show that our method can be successfully used in real-time terrain rendering applications.

6. CONCLUSION AND FUTURE WORK

In this paper, we presented a new image compression algorithm called ACMZW that combines smart adaptive context modeling with zero-tree coding and demonstrates top compression performance in the class of zero tree wavelet coders such as EZW, SPIHT and LTW. At the same time, the algorithmic complexity of the algorithm is comparable to that of LTW since it bypasses multiple bit-plane coding.

Our experiments showed that exploiting proposed context models reduces the compressed bit stream size by more than 10%. Our ACMZW method demonstrates up to **0.6 dB** compression performance superiority over SPIHT with arithmetic encoding [SP96] and up to **0.21 dB** superiority over LTW [OM03] on standard test images (Lena, Barbara).

A new hierarchical compressed multiresolution terrain model is proposed that is based on presented image compression technique and can be exploited in high-quality real-time terrain rendering systems. The model introduces special matching boundary region that is compressed in a way that guarantees the borders of neighboring patches perfectly match. The algorithm demonstrates compression performance that is on par with the best known methods.

Our future work will be aimed at exploiting the recent graphics hardware for improving temporal performance of the algorithms. The most computation-intensive parts of the presented compression technique can be implemented entirely on GPU using compute shader, the new capability exposed by DX11-class hardware.

7. REFERENCES

[WNC87] I.H.Witten, R.M.Neal, and J.G.Cleary, "Arithmetic coding for data compression". Comm. ACM, vol. 30, No. 6, pp.520-540, June 1987.

[ABM92] M. Antonini, M. Barlaud, P. Mathieu, I. Daubechies, "Image coding using wavelet transform," IEEE Trans. Image Proc., Vol. 1, No. 2, pp. 205-220, 1992.

[Sha93] J.M. Shapiro, "Embedded Image Coding Using Zerotrees of Wavelet Coefficients," IEEE Transactions on Signal Processing, vol. 41, no 12, pp. 3445-3462, Dec. 1993.

[SP96] A. Said and W. Pearlman, "A new, fast and efficient image codec based on set partitioning in hierarchical trees," IEEE Trans. Circuits Syst. Video Technol., vol. 6, no 3, pp. 243–250, June 1996.

[CO97] C. Chrysafis, A. Ortega, "Efficient Context-Based Entropy Coding for Lossy Wavelet Image Compression", Proceedings of the Conference on Data Compression, pp.241-250, March 25-27, 1997.

[Wu97] X. Wu, "High-order context modeling and embedded conditional entropy coding of wavelet coefficients for image compression," Proc. ACSSC, Nov. 1997.

[WM97] X. Wu and N.D.Memon, "Context-based, adaptive, lossless image coding," *IEEE Trans. Commun.*, vol. 45 (4), pp. 437-444, Apr. 1997.

[And97] J. Andrew, "A simple and efficient hierarchical image coder," ICIP'97, vol. 3, pp 658-661, 1997.

[AMC98] R. Ansari, N.Memon, and E. Ceran, "Near-lossless image compression techniques," J. Electronic Imaging, vol. 7, pp. 486-494, Jul. 1998.

[Mal00] H. Malvar. "Fast Progressive Image Coding without Wavelets." In Data Compression Conference (DCC '00), pp. 243–252, 2000.

[WSS00] M. Weinberger, G. Seroussi, and G. Sapiro,"The LOCO-I lossless image compression algorithm: principles and standardization into JPEGLS," IEEE Trans. Image Processing, vol. 9, pp. 1309-1324, Aug. 2000.

[BP01] U. Bayazit and W. A. Pearlman, "Algorithmic Modifications to SPIHT". IEEE Int. Conf. on Image Processing (ICIP 2001), Thessaloniki, Greece., Oct. 2001.

[OM03] J. Oliver and M. P. Malumbres, "Fast and efficient spatial scalable image compression using wavelet lower trees," in Proc. 2003 IEEE Data Compression Conference, Mar 2003, pp. 133–142.

[LH04] F. Losasso, H. Hoppe, "Geometry Clipmaps: Terrain Rendering Using Nested Regular Grids". ACM Transactions on Graphics (Proceedings of SIGGRAPH 2004) 23(3), pp. 769-776.

[Zhe04] S.V. Zherzdev. "Development of adaptive video information compression algorithms based on hierarchical structures for operation visualization". PhD Thesis. Nizhny Novgorod State University, Nizhny Novgorod, Russia.

[YP06] S. Yea and W. Pearlman. "A wavelet-based two-stage near-lossless coder". In Proc. ICIP (2004), pp. 2503-2506.

[GMC+06] E. Gobbetti, F. Marton, P. Cignoni, M. Di Benedetto, and F. Ganovelli. "C-BDAM - Compressed Batched Dynamic Adaptive Meshes for Terrain Rendering". Computer Graphics Forum, Volume 25(2006), Number 3.

[Cp07] Y. Cho and W. A. Pearlman, "Quantifying the Performance of Zerotrees of Wavelet Coefficients: Dgree-k Zerotree Model", IEEE Trans. on Signal Processing , Vol. 55, Part 1, pp. 2425-2431, June 2007.

[YT08] Egor Yusov, Vadim Turlapov. "JPEG2000-based compressed multiresolution model for real-time large scale terrain visualization". Proc. of the 18th international Conference on Computer Graphics and Vision "GraphiCon'2008" pp. 164-171, Moscow, June 23-27, 2008.

[DSW09] C. Dick, J. Schneider, and R. Westermann. "Efficient Geometry Compression for GPU-based Decoding in Realtime Terrain Rendering". In Computer Graphics Forum, Vol. 28, No 1, pp. 67–83, 2009.

[PS] Puget Sound elevation data set is available at http://www.cc.gatech.edu/projects/large_models/ps.html