DEPTH-IMAGE COMPRESSION BASED ON AN R-D OPTIMIZED QUADTREE DECOMPOSITION FOR THE TRANSMISSION OF MULTIVIEW IMAGES

Yannick Morvan[1], *Dirk Farin*[1] *and Peter H. N. de With*[1,2]

[1] University of Technology Eindhoven, P.O. Box 513
5600 MB Eindhoven, Netherlands
{y.morvan,d.s.farin}@tue.nl

[2] LogicaCMG, RTSE, P.O. Box 7089
5605 JB Eindhoven, Netherlands
P.H.N.de.With@tue.nl

ABSTRACT

This paper presents a novel depth-image coding algorithm that concentrates on the special characteristics of depth images: smooth regions delineated by sharp edges. The algorithm models these smooth regions using piecewise-linear functions and sharp edges by a straight line. To define the area of support for each modeling function, we employ a quadtree decomposition that divides the image into blocks of variable size, each block being approximated by one modeling function containing one or two surfaces. The subdivision of the quadtree and the selection of the type of modeling function is optimized such that a global rate-distortion trade-off is realized. Additionally, we present a predictive coding scheme that improves the coding performance of the quadtree decomposition by exploiting the correlation between each block of the quadtree. Experimental results show that the described technique improves the resulting quality of compressed depth images by 1.5–4 dB when compared to a JPEG-2000 encoder.

Index Terms— Stereo vision, image coding, geometric modeling.

1. INTRODUCTION

The multiview video technology is based on video data recorded by multiple synchronized cameras. This technology enables applications such as 3D-TV and free-viewpoint video. A free-viewpoint video system provides the user the ability to interactively select a viewpoint in the video scene. As a first step toward standardization of multiview video technologies, algorithms addressing the problem of transmission and rendering of 3-D data are currently investigated by the Multiview Video Coding (MVC) group within MPEG. One technique, i.e. the Depth Image Based Rendering (DIBR) algorithm, has been recognized as a promising tool to perform not only the 3-D rendering, but also the compression of multiview images. For 3-D rendering, the DIBR technique can be employed to synthesize a virtual camera view at an arbitrary position. The idea followed is to take a reference texture and a corresponding depth image as input data (see Figure 1). Given the depth information, 3-D image-warping techniques can subsequently be employed. For compression, the DIBR algorithm can be used to perform predictive coding of views [1]. The predictive coding of views exploits the fact that neighboring camera-views are highly correlated, resulting to a possible coding gain. Similarly, the prediction of the neighboring camera-views is carried out using a reference texture and the corresponding depth image. For an efficient transmission of the above-mentioned reference texture and depth image, the coding of depth images needs to be addressed.

Previous work on depth image coding has used a transform-based algorithm derived from JPEG-2000 [2] and H.264 encoders [3]. However, transform coders have shown a significant shortcoming for representing edges without deterioration at low bit-rates. Perceptually, such a coder generates ringing artifacts along edges that lead to errors in pixel positions, which leads to fuzzy object borders in the synthesized images. This remark is similar to recent conclusions [4] related to depth compression results for multiview video coding.

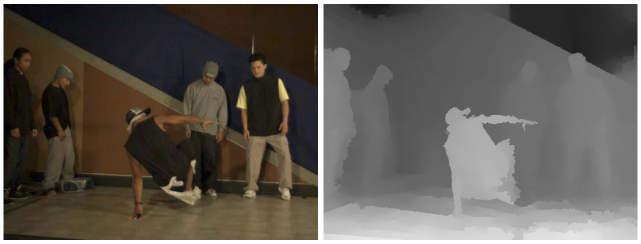


Fig. 1. Example texture image (left) and the corresponding depth image (right). A typical depth image contains regions of linear depth changes bounded by sharp discontinuities.

The characteristics of depth images differ from normal textured images. For example, since a depth image explicitly captures the 3-D structure of a scene, large parts of typical depth images depict object surfaces. As a result, the input depth image contains various areas of smoothly changing gray levels. Furthermore, at the object boundaries, the depth image typically shows sharp edges. Following these observations, we propose to model depth images by piecewise-linear functions separated by straight lines. For this reason, we are interested in an algorithm that captures and compactly approximates these geometrical structures.

The problem of capturing image discontinuities has been extensively studied. Example of coding algorithms are Bandelets [5] or Contourlets [6]. Two alternative contributions based on Wedgelet [7] and Platelet [8] are attractive for depth image coding. The Wedgelet function, is defined as two piecewise-constant functions separated by a straight line. This concept was later extended to piecewise-linear functions and called Platelet functions. To define the area of support of each modeling function, a quadtree segmentation which recursively subdivides the image into blocks of variable size is employed. The advantages of both approaches, wedgelet and platelet are twofold. First, edges location are explicitly transmitted to the decoder and therefore available to the warping algorithm as well. Special treatment of pixel-boundaries can thus be performed while rendering the 3-D images. Second, the Wedgelet and Platelet-based algorithms were specifically designed for denoising images, while preserving the edges. Therefore, depth images are inherently denoised while performing the compression.

For our compression framework, we have adopted the Wedgelet and Platelet signal-decomposition technique. In this way, we follow the concept developed to code image texture using piecewise polynomials [9] as modeling functions. More particularly, we consider

1-4244-1437-7/07/$20.00 ©2007 IEEE

four different piecewise-linear functions for modeling. The first and second modeling function, a constant and a linear function, respectively, are suitable to approximate smooth regions. The third and fourth modeling function attempt to capture depth discontinuities (the sharp edges), using two constant functions or two linear functions separated by a straight line. These modeling functions are used to approximate the image. To this end, the image is subdivided into blocks of variable size using a quadtree decomposition. An independent modeling function is subsequently selected for each node. The selection of the most appropriate function is performed using a cost function that balances both rate and distortion. In a similar way, we determine the optimal decomposition of the image into a quadtree with differing block sizes and the most appropriate quantizer step-size. Finally, to reduce the redundancy between nodes in the quadtree, a predictive coding technique is introduced. In more detail, this means that we decorrelate the remaining dependencies between each block to further enhance coding efficiency. Experimental results show that our proposal yields up to 1–3 dB PSNR improvement over a JPEG-2000 encoder.

The sequel of this paper is structured as follows. Section 2 introduces the framework of our depth-image coding algorithm. In Section 3, we describe a bit-allocation strategy that balances both rate and distortion and Section 4 introduces a predictive coding technique. Experimental results are presented in Section 6 and the paper concludes with Section 7.

2. DEPTH-IMAGE MODELING

In this section, we present a novel approach for depth-image coding using the piecewise-linear functions mentioned in Section 1. The concept followed is to approximate the image content using modeling functions. In our framework, we use two classes of modeling functions: a class of piecewise-constant functions and a class of piecewise-linear functions. For example, regions of constant depth (e.g. flat surfaces not oriented in perspective) show smooth regions in the depth image and can be approximated by a piecewise-constant function. Secondly, planar surfaces of the scene like the ground plane and walls of e.g. a room, appear as regions of gradually changing gray levels in the depth image. Hence, such a planar region can be approximated by a single linear function. To specify the 2D-support of the modeling functions in the image, we employ a quadtree decomposition that hierarchically divides the image into blocks, i.e. nodes of different size. In some cases, the depth image within one block can be approximated with one modeling function. If no suitable approximation can be determined for the block, it is subdivided into four smaller blocks. To prevent that many small blocks are required along a discontinuity, we divide the block into two regions separated by a straight line. Each of these two regions is coded with an independent function. Consequently, the coding algorithm chooses between four modeling functions for each leaf in the quadtree:

- *Modeling function $\hat{f}_1$:* Approximate the block content with a constant function;
- *Modeling function $\hat{f}_2$:* Approximate the block content with a linear function;
- *Modeling function $\hat{f}_3$:* Subdivide the block into two regions A and B separated by a straight line and approximate each region with a constant function (a wedgelet function);

$$\hat{f}_3(x,y) = \begin{cases} \hat{f}_{3A}(x,y) = \gamma_{0A} & (x,y) \in A \\ \hat{f}_{3B}(x,y) = \gamma_{0B} & (x,y) \in B \end{cases}$$

- *Modeling function $\hat{f}_4$:* Subdivide the block into two regions A and B separated by a straight line and approximate each region with a linear function (a platelet function);

$$\hat{f}_4(x,y) = \begin{cases} \hat{f}_{4A}(x,y) = \theta_{0A} + \theta_{1A}x + \theta_{2A}y \\ \hat{f}_{4B}(x,y) = \theta_{0B} + \theta_{1B}x + \theta_{2B}y \end{cases}$$

The decision for each modeling function is based on a rate-distortion decision criterion that is described in Section 3.

3. R-D OPTIMIZED BIT-ALLOCATION

In this section, we aim at providing details about the bit-allocation strategy that optimizes the coding in a rate-distortion sense. Considering our lossy encoder/decoder framework, our aim is to optimize the compression of a given image to satisfy a Rate-Distortion (R-D) constraint. In our practical case, there are three parameters that influence this trade-off: (1) the selection of modeling functions, (2) the quadtree decomposition and (3) the quantization step-size of the modeling-function coefficients. Thus, the problem statement is to adjust each of the previous parameters such that the objective R-D constraint is satisfied. The three aspects will be individually addressed below.

To optimize these three parameters in an R-D sense, the adopted approach is to define a cost function that combines both rate R_i and distortion D_i of the image i. Typically, the Lagrangian cost function

$$J(R_i) = D_i(R_i) + \lambda R_i \tag{1}$$

is used, where R_i and D_i represent the rate and distortion of the image, respectively, and λ is a weighting factor that controls the rate-distortion trade-off. Using the above Lagrangian cost function principle, the algorithm successively performs three independent parameters optimizations: (1) an independent selection of modeling functions, (2) a quadtree structure optimization and (3) the quantizer step-size selection. Let us address all three aspects now.

(1) Modeling function selection. First, we assume that an optimal quadtree segmentation and quantizer step-size is provided. Since the rate and distortion are additive functions over all blocks, an independent optimization can be performed within the blocks. Therefore, for each block, the algorithm selects the modeling function that leads to the minimum coding cost. More formally, for each block, the algorithm selects the best modeling function $\tilde{f}$ in an R-D sense according to

$$\tilde{f} = \arg\min_{\hat{f}_j \in \{\hat{f}_1, \hat{f}_2, \hat{f}_3, \hat{f}_4\}} (D_m(\hat{f}_j) + \lambda R_m(\hat{f}_j)), \tag{2}$$

where $R_m(\hat{f}_j)$ and $D_m(\hat{f}_j)$ represent the rate and distortion resulting from using one modeling function $\hat{f}_j$, respectively.

(2) Quadtree decomposition. To obtain an optimal quadtree decomposition of the image, a well-known approach is to perform a so-called *bottom-up* tree-pruning technique [10]. The guiding principle is to parse the initial full tree from bottom to top and recursively prune nodes (i.e. merge blocks) of the tree according to a decision criterion.

The algorithm can be described as follows. Consider four children nodes denoted by N_1, N_2, N_3 and N_4 that have a common parent node which is represented by N_0. For each node k, a Lagrangian coding cost $(D_{N_k} + \lambda R_{N_k})$ $k \in 0, 1, 2, 3, 4$ can be calculated. Using the Lagrangian cost function, the four children nodes should be pruned whenever the sum of the four coding cost functions is higher than the cost function of the parent node. When the children nodes are not pruned, the algorithm assigns the sum of the coding costs of the children nodes to the parent node. Subsequently, this tree-pruning technique is recursively performed in a bottom-up fashion.

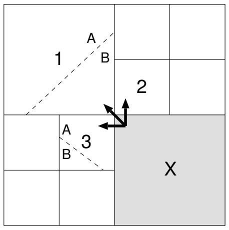


Fig. 2. Illustration for parameter encoding. The grey block X is currently being coded. DC-predictors are formed from the three numbered blocks (median of DC-coefficient of 1-B, 2, and 3-A). See Section 4 for more information.

It has been proved [10] that such a bottom-up tree pruning leads to an optimally pruned tree, thus in our case to an R-D optimal quadtree decomposition of the image.

(3) Quantizer selection. So far, we have assumed that the coefficients of the modeling functions are scalar quantized prior to model selection and tree-pruning. However, no detail has been provided about the selection of an appropriate quantizer. Therefore, the problem is to select the optimal quantizer, denoted $\tilde{q}$, that meets the desired R-D constraint. We propose to select the quantizer $\tilde{q}$ out of a given set of possible scalar quantizers $\{q_2, \ldots, q_8\}$, operating at 2–8 bits per level, respectively. To optimally select the quantizer, we re-use the application of the Lagrangian cost function and select the quantizer $\tilde{q}$ that minimizes the Lagrangian coding cost of the image.

$$\tilde{q} = \underset{q_l \in \{q_2,\ldots,q_8\}}{\arg\min} \quad D_i(R_i, q_l) + \lambda R_i(q_l). \tag{3}$$

Here, $R_i(q_l)$ and $D_i(R_i, q_l)$ correspond to the global rate R_i and distortion $D_i(R_i)$ in which the parameter q_l is added to represent the quantizer selection. To solve the optimization problem of Equation (3), the image is encoded using all possible quantizers and the quantizer $\tilde{q}$ that yields the lowest coding cost $J_i(R_i, \tilde{q}) = D_i(R_i, \tilde{q}) + \lambda R_i(\tilde{q})$ is selected.

4. PREDICTIVE CODING OF PARAMETERS

In this section, the entropy coding of our coding algorithm is described.

Quadtree structure. The quadtree is transmitted top-down, where for each node the binary decision is made if this node is subdivided or not. This decision is coded with an arithmetic encoder with fixed probabilities. The probability that the node is subdivided is depending on the number of neighboring nodes at the same tree-level that are also subdivided. The neighborhood context contains the left, the top/left, and the neighboring top block, such that the number of subdivided neighbors is between 0–3. In the example of Fig. 2, the number of subdivided neighbors is two.

Coding mode. The coding mode for each block in the quadtree is coded with an input-adaptive arithmetic coder. Fixed probabilities are not suitable here, since the selection of coding modes depends on the bit-rate (more complex models for higher bit-rates).

DC coefficients. The DC coefficients of all input blocks are highly correlated. For this reason, we predict their value from previously-coded blocks and only code the residual value. More specifically, we consider the three blocks that are adjacent to the top-left pixel in the current block. The predictor is formed by the median of the DC coefficients of these three blocks. If a block is subdivided into two models and hence has two DC coefficients, we use the DC-coefficient of the region that is adjacent to the top-left pixel of the current block. If the current block is represented with two functions, the predictor is used for both. Note that this scheme always works properly, whatever coding modes have been used for these blocks. In the example in Fig. 2, the DC coefficients of the functions 1-B, 2, and 3-A would be used to build the predictor for block X.

The distribution of the residual signal after prediction subtraction is non-uniform. Consequently, coding of the residual value is carried out with an input-adaptive arithmetic encoder. Note that the residual value for an n-bit quantized image requires $n + 1$ bits, but that this range can be folded onto itself such that only 2^n different symbols have to be considered.

AC coefficients The AC parameter corresponds to the first-order parameters of the functions $\tilde{f}_2$ and $\tilde{f}_4$. For $\tilde{f}_4$, this means θ_{1A}, θ_{2A}, θ_{1B} and θ_{2B}. The statistics for these four parameters and the two parameters for function $\tilde{f}_2$ are non-uniformly distributed. This implies the use of a variable-length coder as well. Therefore, the four parameters were coded with an input-adaptive arithmetic encoder as well.

5. ALGORITHM SUMMARY

The algorithm requires as an input the depth image and a weighting factor λ that controls the R-D trade-off. The image is first recursively subdivided into a full quadtree decomposition. All nodes of the tree are then approximated by the four modeling functions $\hat{f}_1$, $\hat{f}_2$, $\hat{f}_3$, $\hat{f}_4$. In a second step, the coefficients of the modeling functions are quantized using one scalar quantizer q_l. For each node of the full tree, an optimal modeling function $\tilde{f}$ can now be selected using Equation (2). Employing the tree-pruning technique described in Section 3, the full tree is then pruned in a bottom-up fashion. The second step of the algorithm (i.e. the coefficient quantization) is repeated for all quantizers $q_l \in \{q_2, \ldots, q_8\}$ and the quantization that leads to the lowest global coding cost is selected (see Equation (3)). Subsequently, for each leaf of the tree, the quantized zero-order coefficients are predicted using the neighboring values as explained in Fig. 2 and because the residual values are non-uniformly distributed, they are arithmetically coded. Note that the first-order coefficients satisfy a Laplacian distribution. If it would be a fixed Laplacian distribution, we could have "simply" used a fixed probability table. However, we do not know the exponential coefficient value, it is still valid to use an adaptive encoder. For this reason, we encode these coefficients using an adaptive arithmetic encoder.

Note that the rate-distortion optimization is based on fixed-length codes to enable a fast computation. The final bit-rate is lower since the parameters are further compressed with an arithmetic encoder. The gain of the prediction scheme with the subsequent arithmetic encoding yields between 0.5–1.5 dB of gain for a fixed bit rate.

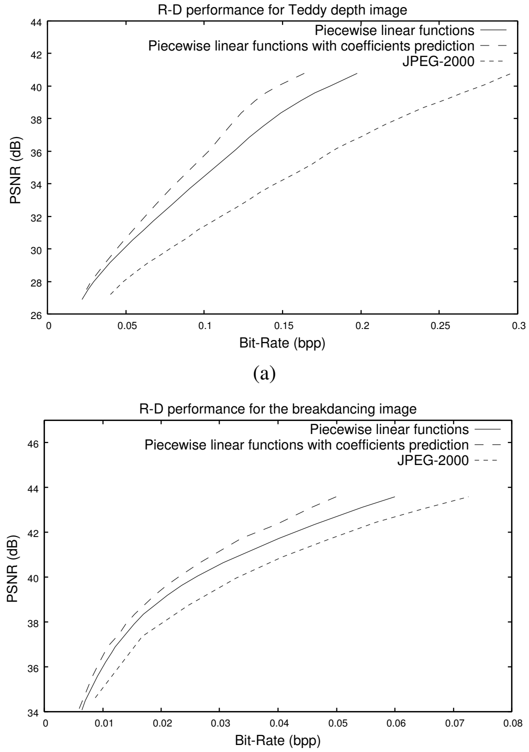
6. EXPERIMENTAL RESULTS

We first conducted experiments using the depth image "Teddy" [1]. Experiments have revealed that the proposed algorithm can approximate large smooth areas as well as sharp edges with a single node. Second, the approximation capabilities were evaluated with the depth image "Breakdancing" [11] [2]. Subsequently, the resulting R-D performances were compared to a JPEG-2000 encoder. Figures 3(a)

[1] available at http://www.middlebury.edu/stereo, accessed January 2007

[2] "Breakdancing" depth image number 0 of camera 0.

and 3(b) show that the described encoder consistently outperforms the JPEG-2000 encoder. For example, for the fixed-length coding without coefficient prediction, improvements over JPEG-2000 are as high as 2.8 dB at 0.1 bit per pixel for the "Teddy" image. For the "Breakdancing" image, a gain of 1.3 dB can be obtained at 0.025 bit per pixel. Using coefficient prediction, the figures are even clearly better. For the "Teddy" image, the additional improvement is about 1 dB and for the "Breakdancing" image, the extra improvement is approximately 0.3 dB, both measured at the previously mentioned bit rates. Perceptually, the algorithm "Piecewise linear functions" reconstructs edges of higher quality than the JPEG-2000 encoder (see Figure 4(c)).



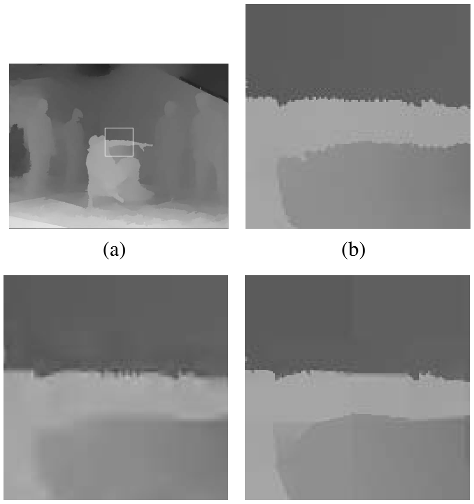
(a)

(b)

Fig. 3. Rate-distortion curves for the "Teddy" 3(a) and "Breakdancing" 3(b) depth images, both for our algorithm and JPEG-2000.

7. CONCLUSIONS

We have presented a new algorithm for coding depth images that exploits the smooth properties of depth signals. Regions are modeled by piecewise-constant and piecewise-linear functions and they are separated by straight lines along their boundaries. The algorithm employs a quadtree decomposition to enable the coding of small details as well as large regions with a single node. The performance of the full coding algorithm can be controlled by three different aspects: (1) the choice the modeling function, (2) the level of the quadtree segmentation (variable block size), and (3) the overall coefficient-quantization setting for the image. All three aspects are individually controlled by a Lagrangian cost function, in order to impose a global rate-distortion constraint for the complete image. A predictive coding of the modeling-function parameters is employed to reduce the remaining interblock correlation after quadtree decomposition. The residual signals (i.e. coefficients) are arithmetically coded. For typical bit-rates (i.e. between 0.01 bit/pixel and 0.25 bit/pixel), experiments have shown that the coder outperforms a JPEG-2000 encoder by $1.5 - 4\ dB$. The proposed algorithm is intended to be used in a 3-D video coding system. We think that this proposal is more suitable for handling the typical characteristics of a depth signal than conventional transform coders, because the modeling functions comply with the geometrical structures in depth images. However, further study is needed to reduce the algorithm complexity.



(a)

(b)

(c)

(d)

Fig. 4. (a) Original "Breakdancing" depth image, (b) Magnified view of the marked area, (c) Marked area coded with JPEG-2000 at 38.7 dB of PSNR, and (d) with piecewise-linear functions at 40.0 dB of PSNR Both results are obtained at 0.025 bit per pixel.

8. REFERENCES

[1] E. Martinian, A. Behrens, J. Xin, A. Vetro, and H. Sun, "Extensions of h.264/avc for multiview video compression," in *IEEE Int. Conf. on Image Proc.*, Atlanta, USA, October 2006.

[2] R. Krishnamurthy, B.-B. Chai, H. Tao, and S. Sethuraman, "Compression and transmission of depth maps for image-based rendering," *IEEE Int. Conf. on Image Proc.*, vol. 3, pp. 828–831, October 2001.

[3] A. Smolic and P. Kauff, "Interactive 3d video representation and coding technologies," *Proc. of the IEEE*, vol. 93, no. 1, Janunary 2005.

[4] C. Fehn, N. Atzpadin, M. Mller, O. Schreer, A. Smolic, R. Tanger, and P. Kauff, "An advanced 3DTV concept providing interoperabilty and scalabilty for a wide range of multi-baseline geometries," in *IEEE Int. Conf. on Image Proc.*, Atlanta, USA, October 2006.

[5] G. Peyre and S. Mallat, "Surface compression with geometric bandelets," in *ACM SIGGRAPH*, July 2005, vol. 24, pp. 601–608.

[6] M. N. Do and M. Vetterli, "The contourlet transform: an efficient directional multiresolution image representation," *IEEE Transactions Image on Processing*, vol. 14, no. 12, pp. 2091–2106, Dec 2005.

[7] D. Donoho, "Wedgelets: nearly minimax estimation of edges," *Annals of Statistics*, vol. 27, no. 3, pp. 859–897, March 1999.

[8] R. M. Willett and R. D. Nowak, "Platelets: a multiscale approach for recovering edges and surfaces in photon-limited medical imaging," *IEEE Trans. on Medical Imaging*, vol. 22, pp. 332–350, March 2003.

[9] R. Shukla, P. L. Dragotti, M. N. Do, and M. Vetterli, "Rate-distortion optimized tree-structured compression algorithms for piecewise polynomial images," *IEEE Trans. on Image Proc.*, vol. 14, pp. 343–359, March 2005.

[10] P. A. Chou, T. D. L., and R. M. Gray, "Optimal pruning with applications to tree-structured source coding and modeling.," *IEEE Trans. on Inf. Theory*, vol. 35, no. 2, pp. 299–315, 1989.

[11] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-quality video view interpolation using a layered representation," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 600–608, 2004.