Platelet-based coding of depth maps for the transmission of multiview images

Yannick Morvan[a], Peter H. N. de With[a,b] and Dirk Farin[a]

[a] Eindhoven University of Technology, P.O. Box 513, The Netherlands;
[b] LogicaCMG, P.O. Box 7089, 5605 JB Eindhoven, The Netherlands
y.morvan@tue.nl

ABSTRACT

Emerging 3-D displays show several views of the scene simultaneously. A direct transmission of a selection of these views is impractical, because various types of displays support a different number of views and the decoder has to interpolate the intermediate views. The transmission of multiview image information can be simplified by only transmitting the texture data for the central view and a corresponding depth map. Additional to the coding of the texture data, this technique requires the efficient coding of depth maps. Since the depth map represents the scene geometry and thereby covers the 3-D perception of the scene, sharp edges corresponding to object boundaries, should be preserved. We propose an algorithm that models depth maps using piecewise-linear functions (platelets). To adapt to varying scene detail, we employ a quadtree decomposition that divides the image into blocks of variable size, each block being approximated by one platelet. In order to preserve sharp object boundaries, the support area of each platelet is adapted to the object boundary. The subdivision of the quadtree and the selection of the platelet type are optimized such that a global rate-distortion trade-off is realized. Experimental results show that the described method can improve the resulting picture quality after compression of depth maps by $1-3$ dB when compared to a JPEG-2000 encoder.

Keywords: Depth image coding, model based coding, piecewise-linear function, platelets-based coding

1. INTRODUCTION

Upcoming 3-D display technology allows the presentation of images with the illusion of depth. Such display technology is based on showing several views of the same scene at the same time, exploring the different eye positions of the human visual system. An independent transmission of these views has several disadvantages. First, it is inefficient since there is a strong correlation between images. Second, different types of displays are supporting a different number of views, which makes it impractical to prepare the displayed views at the encoder. Instead, the views should be synthesized at the decoder where the display characteristics may be known. These disadvantages can be eliminated by transmitting the texture data independently from the geometry data. One approach is to send the texture data for the central view and a depth map[1] that specifies the depth of each pixel in the texture image. This depth map, also called depth image, can be represented by a gray-scale image where dark and bright pixels correspond to far and near pixels, respectively (see Figure 1). Based on this depth map, arbitrary views can be synthesized at the decoder. For efficient transmission of 3-D image data, the coding of depth maps needs to be addressed.

Previous work on depth image coding has used a transform-based algorithm derived from JPEG-2000[3] and MPEG encoders.[4] Key advantage of using a standardized video coding algorithm to compress depth images is the backward compatibility with already existing technology. 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 appears as a blurred depth signal along the object borders. An alternative approach is to use triangular meshes[5,6] to code depth maps. This investigated technique divides the image into triangular patches and approximates each patch by a linear function. If the data cannot be represented with a single linear function, smaller patches are used for that area. However, the placement of the patches is usually based on a regular grid, such that a large number of small patches are generated along edges.

Figure 1. Example texture image (left) and the corresponding depth map (right).[2] A typical depth image contains regions of linear depth changes bounded by sharp discontinuities.

The characteristics of depth maps differ from normal textured images. For example, since a depth map 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 map shows step functions, i.e. 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 extracts and compactly approximates these geometrical structures.
Several contributions have considered the use of geometrical models to approximate images. One modelling function, the "Wedgelet"[7] function, is defined as two piecewise-constant functions separated by a straight line. This concept was originally introduced as a mean of detecting and recovering edges from noisy images, and was later extended to piecewise-linear functions, called "Platelet"[8] functions. To define the area of support of each modeling function, a quadtree segmentation of the image is used. The concept is to recursively subdivide the image into blocks of variables size and approximate each block with an appropriate model.

Considering our compression framework, we adopt the "Wedgelet" and "Platelet" signal decomposition paradigm. Afterwards, the decomposition accuracy and the subsequent coding is subjected to a rate-distortion trade-off. In this way, we follow the idea developed to code image texture using piecewise polynomials[9] as modeling functions. More particularly, we consider four different piecewise-linear functions as modeling functions. The first and second modeling functions, a constant and linear function, respectively, are suitable to approximate smooth regions. The third and fourth modeling functions attempt to capture depth discontinuities, 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 variable-sized blocks 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 employ an equivalent cost function that determines the optimal block sizes of the quadtree. Our results show that, the 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 briefly illustrates the proposed acquisition and transmission system of multiview images. Section 3 introduces the framework of our depth image coding algorithm, while Section 4 provides further details about the modeling functions and coefficient estimation techniques. In Section 5, we describe a bit-allocation strategy that balances both rate and distortion. Experimental results are presented in Section 6 and the paper concludes with Section 7.

2. ACQUISITION AND TRANSMISSION SYSTEM FOR MULTIVIEW IMAGES

This section aims at describing our system (see Figure 2) for acquisition, coding/decoding and rendering of multiview images. Before describing the processing step performed on multiview images, let us first examine the acquisition sub-system.

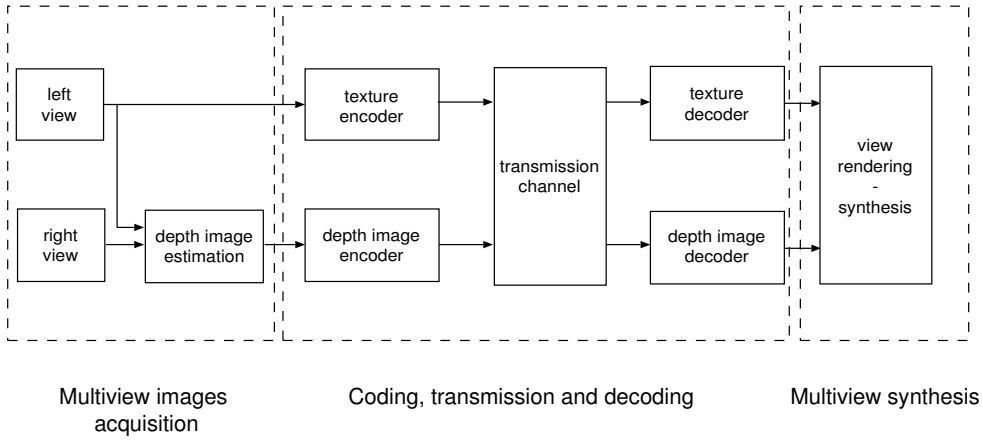


Figure 2. Overview of our multiview image transmission system.

2.1. Multiview image acquisition

To acquire multiview images, one possible approach is to capture a texture image and the corresponding 3-D geometry of the scene. The 3-D geometry can be acquired by recording the scene from several viewpoints. In practice, two points of view corresponding to a left and right camera are usually employed. By comparing differences between the two captured images, the depth (that corresponds to the 3-D geometry) can be estimated and represented in a so-called depth image. This depth image is represented by a gray-scale image: dark and bright pixels correspond to background and foreground distance, respectively.

2.2. Displaying multiview images

To synthesize a new view for display, we first convert the texture image into a set of 3-D points. We assume that the focal-length of the recording camera is a fixed parameter F and the principal point of the image is at (o_x, o_y). This lets us calculate the 3-D position $\boldsymbol{p}$ of each pixel (x, y) in *homogeneous* coordinates by

$$\boldsymbol{p} = ((x - o_x)z/F),\ (y - o_y)z/F,\ z,\ 1)^\top,$$

where $z = F + D(x, y)$ is the distance of the pixel to the recording camera and $D(x, y)$ is the depth map value at position (x, y). To synthesize a *new* view of the scene at an arbitrary camera position $\boldsymbol{t}$, we employ a 3×3 camera rotation matrix $\boldsymbol{R}$, and a *new* corresponding focal-length f, to obtain the projection of each pixel as

$$\boldsymbol{p'} = \begin{pmatrix} f & 0 & o_x \\ 0 & f & o_y \\ 0 & 0 & 1 \end{pmatrix} \left(\boldsymbol{R}|\boldsymbol{Rt}\right) \boldsymbol{p}.$$

Here, $\boldsymbol{Rt}$ forms the translation component that adds to $\boldsymbol{R}$ so that the total matrix $\left(\boldsymbol{R}|\boldsymbol{Rt}\right)$ involves the complete rotation and translation to the new position. For the case that the observation camera is identical to the capturing camera, it follows that $\boldsymbol{t} = \boldsymbol{0}$, $f = F$, and $\boldsymbol{R}$ is equal to the identity matrix, such that we obtain exactly the input image. Example views that were synthesized from a single texture image and the depth information are depicted in Figure 3.

2.3. Transmission system

For an efficient transmission of multiview images, the problem of compression of the texture and depth image should be addressed. In past years, numerous video coding standards such as MPEG-1, MPEG-2 and MPEG-4 have been developed to perform efficient compression of video signals. However, there are only a few proposals suited for an efficient compression of depth images. Therefore, we propose a novel depth image coder which we embed in our transmission system.

(a) (b) (c) (d) (e)

Figure 3. Various views of a scene. All views were synthesized using only a single texture image and a corresponding depth map.

3. 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, flat surfaces that show smooth regions in the depth image can be approximated by a piecewise-constant function. Secondly, planar surfaces of the scene like the ground plane and walls, 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 identify the location of these surfaces in the image, we employ a quadtree decomposition (see Figure 4) which recursively 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 separated by a straight line and approximate each region with a constant function (a wedgelet function);
- *Modeling function* $\hat{f}_4$*:* Subdivide the block into two regions separated by a straight line and approximate each region with a linear function (a platelet function);

The decision for each modeling function is based on a rate-distortion decision criterion that is described in Section 5.1.

4. MODELING FUNCTIONS

In this section, we first define the four different modeling functions $\hat{f}_j$ $j \in \{1, 2, 3, 4\}$ used to approximate each quadtree block. Secondly, we describe a set of techniques that computes the coefficients of the modeling functions. In the sequel of this paper, we denote the depth value at pixel position $(x, y) \in S$ by $f(x, y)$, where S is the area support of an $n \times n$ pixels quadtree block.

4.1. Modeling functions

To model the depth signal within each block, we focus on finding a compression scheme that can compactly represent smooth blocks as well as boundaries (contours) of objects. First, to approximate smooth blocks, we

(a) (b) (c)

Figure 4. Example of quadtree decomposition. Each block, i.e. nodes, of the quadtree is approximated by one modeling function.

employ a simple constant function $\hat{f}_1(x, y) = \alpha_0$. Second, to approximate smooth object surfaces, we use a piecewise-linear function $\hat{f}_2$ defined by

$$\hat{f}_2(x, y) = (\beta_0 + \beta_1 x + \beta_2 y), \quad \text{for } (x, y) \in S.$$

Third, to properly approximate blocks with sharp edges, we separate those blocks into two regions A and B divided by a straight line. Each region is subsequently approximated by one modeling function. To define the subdivision-line, we connect two pixels, P_1 and P_2, located at a different sides of the block. For coding this line, instead of saving the coordinates of both marker points, we use two indexes I_1 and I_2 addressing the location of P_1 and P_2. The index is defined such that all pixels directly surrounding the block are scanned in a clockwise fashion (see Figure 5). To determine if a pixel belongs to region A or B, we use a numerical test known as the

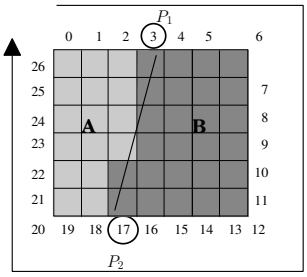


Figure 5. An example edge line that divides a quadtree block into two regions A and B, using pixels $P1$ (index 3) and $P2$ (index 17) as marker points.

orientation test in computational geometry applications.

The two resulting regions A and B can be approximated by either a wedgelet function, or a platelet function. A wedgelet function is composed of two piecewise-constant functions over two regions A and B, separated by a straight line and can be defined as

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

Similarly, a platelet function is defined by

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

Figure 6 depicts example patterns for each modeling function $\hat{f}_1$, $\hat{f}_2$, $\hat{f}_3$ and $\hat{f}_4$.

(a) (b) (c) (d)

Figure 6. Figures 6(a)–6(d) depict examples of functions $\hat{f}_1$, $\hat{f}_2$, $\hat{f}_3$ and $\hat{f}_4$, respectively.

4.2. Estimation of model coefficients

The objective of this processing step is to provide the model coefficients that minimize the approximation error between the original depth signal in a block and the corresponding approximation.

4.2.1. Coefficient estimation for modeling functions $\hat{f}_1$ and $\hat{f}_2$

For $\hat{f}_1(x, y)$, only one coefficient α_0 has to be computed. Practically, the coefficient α_0 that minimizes the error between f and $\hat{f}_1$ simply corresponds to the mean value of the original data, so that $\alpha_0 = \frac{1}{n \times n} \sum_{x,y \in S} f(x, y)$.

Secondly, it was indicated earlier that we use a linear function $\hat{f}_2$ to approximate blocks that contain a gradient. In order to determine the three coefficients β_0, β_1 and β_2 of the linear function $\hat{f}_2(x, y) = \beta_0 + \beta_1 x + \beta_2 y$, a least-squares optimization is used. This optimization minimizes the sum of squared differences between the depth image $f(x, y)$ and the proposed linear model. Accordingly, coefficients β_0, β_1 and β_2 are determined such that the error

$$e_2(\beta_0, \beta_1, \beta_2) = \sum_{x=1}^{n} \sum_{y=1}^{n} (\beta_0 + \beta_1 x + \beta_2 y - f(x, y))^2$$

is minimized, where n denotes the node size expressed in pixels. This error function $e_2(\beta_0, \beta_1, \beta_2)$ is minimal when the gradient satisfies $||\nabla e_2|| = 0$. When taking the partial derivatives with respect to β_0, β_1 and β_2 of this equation, we find a set of linear equations specified by

$$\begin{pmatrix} t & u & v \\ u & t & v \\ v & v & n^2 \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_0 \end{pmatrix} = \begin{pmatrix} \sum_{x=1}^{n} \sum_{y=1}^{n} x f(x, y) \\ \sum_{x=1}^{n} \sum_{y=1}^{n} y f(x, y) \\ \sum_{x=1}^{n} \sum_{y=1}^{n} f(x, y) \end{pmatrix},$$

with

$$t = \frac{n^2(n+1)(2n+1)}{6}, \quad u = \frac{n^2(n+1)^2}{4}, \text{ and } v = \frac{n^2(n+1)}{2}.$$

Since the matrix at the left side of the equation system is constant, it is possible to pre-compute and store the inverse for each size of the square (quadtree node). This enables to compute the coefficients $\beta_0, \beta_1, \beta_2$ with a simple matrix multiplication.

4.2.2. Coefficient estimation for functions $\hat{f}_3$ and $\hat{f}_4$

For the wedgelet $\hat{f}_3$ and platelet $\hat{f}_4$ functions, we have to determine not only the model coefficients, but also the separating line. For each model, we aim at approximating the depth value $f(x, y)$ with two functions in the supporting areas A and B. Each area is delineated by a subdividing line P_1P_2, defined by points P_1 and P_2 (see Figure 5). The best set of coefficients for the wedgelet $\hat{f}_3$ and platelet $\hat{f}_4$ functions are those coefficient values that minimize the approximation errors

$$e_3 = ||f - \hat{f}_{3A}||^2 + ||f - \hat{f}_{3B}||^2 \quad \text{and} \quad e_4 = ||f - \hat{f}_{4A}||^2 + ||f - \hat{f}_{4B}||^2,$$

respectively. These optimization problems cannot be solved directly for the following reason. Without knowing the subdivision boundary between the two regions, it is not possible to determine the model coefficients. On the other hand, it is not possible to identify the subdivision line as long as the region coefficients are unknown. We

propose to first identify the separation line. For this reason, the coefficient estimation is initialized by testing every possible line that divides the block into two areas. This step provides a candidate subdivision line $P_1^c P_2^c$ and two candidate regions A^c and B^c. Subsequently, the wedgelet and platelet coefficients are computed over the candidate regions using the average pixel value and a least-squares minimization, respectively. At the end of the exhaustive search for each function $\hat{f}_3$ and $\hat{f}_4$, the best fitting model is selected. The major advantage of this technique is that it provides the optimal set of coefficients minimizing the least-squares error between the model and the image data. However, such a full search is computationally expensive. Therefore, we also investigated a fast sub-optimal coefficient-estimation algorithm.[10]

4.3. Summary of modeling functions and estimation of model coefficients

Table 1 shows a summary of possible modeling functions to approximate each block of the quadtree and their corresponding coefficient-estimation techniques.

Modeling function	Coefficient-estimation technique
$\hat{f}_1(x,y) = \alpha_0$	Average value of pixel $(x,y) \in S$.
$\hat{f}_2(x,y) = \beta_0 + \beta_1 x + \beta_2 y$	Compute least-squares minimization.
$\hat{f}_3(x,y) = \begin{cases} \hat{f}_{3A}(x,y) = \gamma_{0A} & (x,y) \in A \\ \hat{f}_{3B}(x,y) = \gamma_{0B} & (x,y) \in B \end{cases}$	Average pixel values over regions A and B where A and B are obtained through an exhaustive search.
$\hat{f}_4(x,y) = \begin{cases} \hat{f}_{4A}(x,y) = \theta_{0A} + \theta_{1A} x + \theta_{2A} y & (x,y) \in A \\ \hat{f}_{4B}(x,y) = \theta_{0B} + \theta_{1B} x + \theta_{2B} y & (x,y) \in B \end{cases}$	Compute least-squares minimization over regions A and B where A and B are obtained through an exhaustive search.

Table 1. Summary of modeling functions and their corresponding coefficient-estimation techniques.

5. BIT-ALLOCATION STRATEGY

In this section, we aim at providing details about the bit-allocation strategy that optimizes the coding in a rate-distortion sense. Lossy compression is a trade-off between rate and distortion. Considering our lossy encoder/decoder framework, our aim is to optimize the compression of a given image to satisfy an objective Rate-Distortion (R-D) constraint. In our practical case, there are three parameters that influence this trade-off.

- **Quadtree decomposition.** To accurately approximate a block that shows fine details, the quadtree decomposition recursively subdivides the block into smaller blocks. However, additional subdivisions lead to a higher bit-rate. Therefore, an appropriate subdivision criterion which prevents that many small blocks are created, is required.
- **Selection of modeling functions.** There are four different modeling functions $\hat{f}_j$ $j \in \{1,2,3,4\}$ to approximate a given block. Each modeling function has a different approximation accuracy, but also different complexity (thus rate). Therefore, an appropriate model that balances both rate and distortion has to be selected.
- **Quantization step-size.** Each function coefficient is quantized so that the quantization step-size should be adapted to the required R-D constraint.

Thus, our central problem is to adjust each of the above parameters such that the objective R-D constraint is satisfied. To optimize these parameters in an R-D sense, a possible 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

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

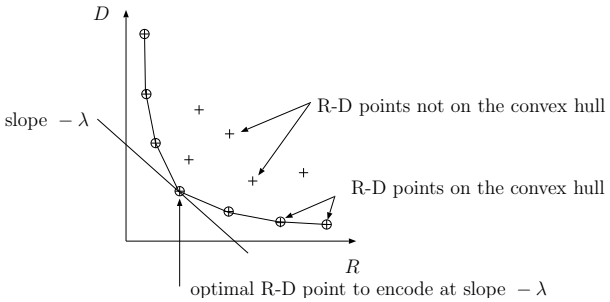


Figure 7. Rate-distortion plane and convex hull of the R-D points.

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. For example, for $\lambda = 0$, we obtain a signal reconstructed at the lowest distortion and highest bit-rate achievable. At the opposite, for $\lambda = \infty$, the signal is compressed at lowest rate and highest distortion. To achieve an optimal encoding, the Lagrangian cost function $J(R_i)$ should be minimized. This cost function is minimized when the derivative is set to 0, which yields

$$\lambda = -\frac{\partial D(R_i)}{\partial R_i}.$$

Thus, the optimal encoding is obtained when compressing the image at an operation point which has a constant slope $-\lambda$ in the convex hull of the R-D points (see Figure 7). Additionally, it can be shown that for an optimal encoding, not only the image must be encoded at constant slope $-\lambda$, but also each independent sub-image. As a consequence, the key principle followed in the discussed bit-allocation strategy is to adjust parameters (i.e. the quadtree decomposition, the model selection and the quantizer step-size), such that the image and also each quadtree block is encoded at a constant slope $-\lambda$.

However, optimizing all parameters in a single step is complicated. For example, without a given quadtree segmentation, it is not possible to select an appropriate modeling function for each quadtree block. To simplify the optimization problem, let us first assume that an optimal quadtree segmentation and quantizer step-size is provided. Using this information, the selection of modeling functions for each quadtree block can be performed.

5.1. Modeling function selection

In Section 4, it was explained that a quadtree block can be approximated by four different modeling functions $\hat{f}_1$, $\hat{f}_2$, $\hat{f}_3$ and $\hat{f}_4$. The objective is not only to select the most appropriate modeling function, but also to incorporate this choice in the overall R-D constraint. The function mentioned in Equation (1) represents the global cost function for the image i. However, the objective is now to select for *each block* the modeling function $\hat{f}_{j,\ j\in\{1,2,3,4\}}$ that minimizes the global coding cost $D_i(R_i) + \lambda R_i$.

Since the rate and distortion are additive functions over all blocks, an independent optimization can be performed for each block. 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

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

where $R_m(\hat{f}_j)$ and $D_m(\hat{f}_j)$ represent the rate and distortion resulting from using one modeling function $\hat{f}_j$. In the implementation, the applied distortion measure is the squared error and the rate for each modeling function is derived from the quantizer step-size.

5.2. Quadtree decomposition

Let us now provide more details about the quadtree decomposition. The objective is to obtain an optimal quadtree decomposition of an image in the R-D sense. Because the quadtree segmentation is not known in

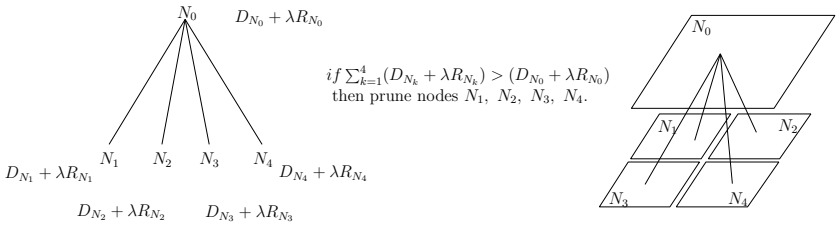


Figure 8. Illustration of a bottom-up tree pruning procedure.

advance, the algorithm first segments the image into a full quadtree decomposition up to the pixel level. From this full tree, the algorithm aims at deriving a sub-tree that optimally segments and approximates the image in the R-D sense. To obtain an optimal tree decomposition of the image, a well-known approach is to perform a so-called *bottom-up* tree-pruning technique.[11] 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. We have adopted from earlier work, to base the decision criterion on the Lagrangian cost function.

The algorithm is illustrated by Figure 8 and 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. More formally written, the children nodes are pruned whenever the following condition holds

$$\sum_{k=1}^{4}(D_{N_k} + \lambda R_{N_k}) > (D_{N_0} + \lambda R_{N_0}). \tag{3}$$

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. It has been proved[11] 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.

To estimate the bit-rate at block level, we also include the side information required to code the node of the quadtree, the modeling function index and the bit-rate resulting from using the modeling function. We adress all three elements briefly. For the side information, we assume that the quadtree structure can be coded using one bit per node (one bit set to "1" or "0" to indicate that the node is subdivided or not). For the index, two bits per leaf are necessary to indicate which modeling function is used. The previous two aspects contribute to the bit-rate. Hence, the rate R_{N_k} that is required to code one given node N_k can be written as

$$R_{N_k} = 1 + 2 + R_m(\tilde{f}),$$

where $R_m(\tilde{f})$ is the bit-rate necessary to code the best modeling function $\tilde{f}$ (see Equation (2)). The distortion D_{N_k} for one node N_k corresponds to the distortion introduced by the selected modeling function $\tilde{f}$.

5.3. Quantizer selection

A criterion that selects the optimal quantizer is now described. Up till now, we have described a scheme that subdivides the image into a full quadtree decomposition. For each node of the full tree, an appropriate modeling function $\tilde{f}$ is selected using the criterion of Equation (2). The full quadtree is then pruned by applying a bottom-up tree pruning technique. 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 yields the desired R-D trade-off. We propose to select the quantizer $\tilde{q}$ out of a given set of possible scalar quantizers $\{q_2, q_3, q_4, q_5, q_6, q_7, q_8\}$, operating at 2, 3, 4, 5, 6, 7 and 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 i

$$\tilde{q} = \underset{q_l \in \{q_2, q_3, q_4, q_5, q_6, q_7, q_8\}}{\arg\min} D_i(R_i, q_l) + \lambda R_i(q_l). \tag{4}$$

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 (4), the image is encoded using all possible quantizers $\{q_2, q_3, q_4, q_5, q_6, q_7, q_8\}$ 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.

5.4. 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 four modeling functions $\hat{f}_0$, $\hat{f}_1$, $\hat{f}_2$, $\hat{f}_3$. 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 5.2, the full tree is then pruned in a bottom-up fashion. The second step of the algorithm is repeated for all quantizers $q_l \in \{q_2, q_3, q_4, q_5, q_6, q_7, q_8\}$ and the quantizer that leads to the lowest global coding cost is selected (see Equation (4)). Subsequently, for each leaf of the tree, the quantized zero-order coefficients $(\alpha_0, \beta_0, \gamma_{0A}, \gamma_{0B}, \theta_{0A}, \theta_{0B})$ are saved and because they are uniformly distributed, they are fixed-length coded. However, the first-order coefficients (β_1, β_2, θ_{1A}, θ_{2A}, θ_{1B}, θ_{2B}) satisfy a Laplacian distribution. For this reason, we encode these coefficients using an adaptive arithmetic encoder. The pseudo-code of the algorithm is summarized in Algorithm 1.

Algorithm 1 Encode depth image - algorithm summary

Require: A depth image and a weighting factor λ to control the R-D trade-off.

Step 1:

1. Recursively subdivide the image into a full quadtree decomposition.
2. Approximate each node of the full tree by four modeling functions $\hat{f}_0, \hat{f}_1, \hat{f}_2, \hat{f}_3$ (see Table 1).

Initialize l $\leftarrow$ 2, $\tilde{q} \leftarrow q_2$.

Step 2:

1. For each node of the full tree, quantize coefficients of the modeling functions with quantizer q_l, and select a best modeling function $\tilde{f}$ using the criterion of Equation (2).
2. Prune the tree in a bottom-up fashion.
3. If the global coding cost $J_i(R_i, q_l) \leq J_i(R_i, \tilde{q})$, then update the best R-D quantizer, i.e. $\tilde{q} \leftarrow q_l$.

Step 3: l $\leftarrow$ l+1, if $l \leq 8$, go to Step 2.

6. EXPERIMENTAL RESULTS

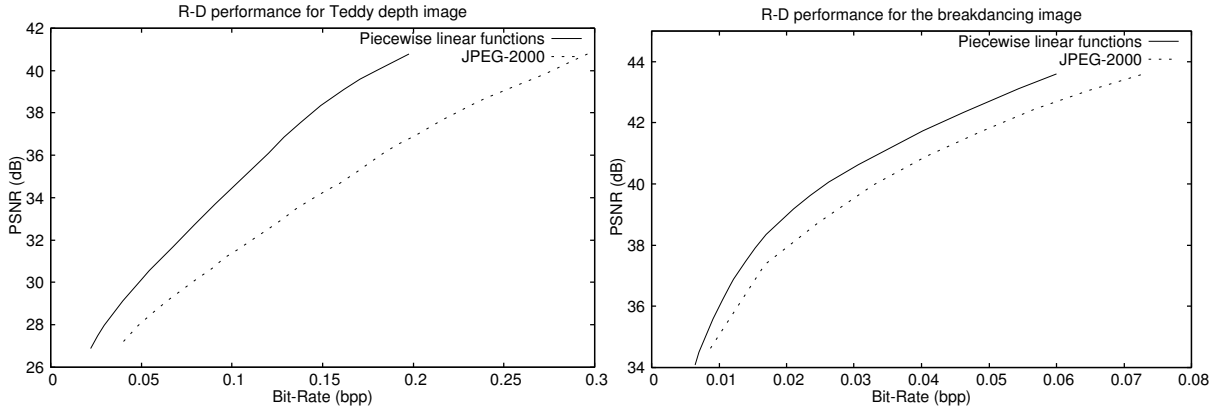
Prior to discussing the coding performance of the algorithm, it is important to notice that the coding efficiency is affected by the quality of the depth image. For example, a noisy depth image results in a low compression factor. For this reason, we first conducted experiments using a high-quality ground-truth depth image "Teddy" (450 × 375 pixels). Experiments have revealed that the proposed algorithm can approximate large smooth areas as well as sharp edges with a single node (see Figure 9). Examples are the top center node and the left vertical edge in Figure 9(c). Furthermore, the approximation capabilities were evaluated with the depth image "Breakdancing"[12] (1024 × 768 pixels) *. Subsequently, the resulting R-D performances were compared to a

* "Breakdancing" depth image number 0 of camera 0.

(a) (b) (c)

Figure 9. Figure 9(a) shows the original depth image "Teddy". The corresponding reconstructed depth image is depicted on Figure 9(b) and the superimposed nodes of the quadtree are portrayed by Figure 9(c). Coding for the depth image "Teddy" is achieved at a bit-rate of 0.12 bit/pixel for a PSNR of 36.1 dB.

JPEG-2000 encoder.[13] Figures 10(a) and 10(b) show that the described encoder consistently outperforms the JPEG-2000 encoder. For example, 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. Perceptually, our algorithm reconstructs edges of higher quality than the JPEG-2000 encoder. For example, the horizontal edge in Figure 11(d) is accurately approximated when compared to the JPEG-2000 encoded signal (see Figure 11(c)).

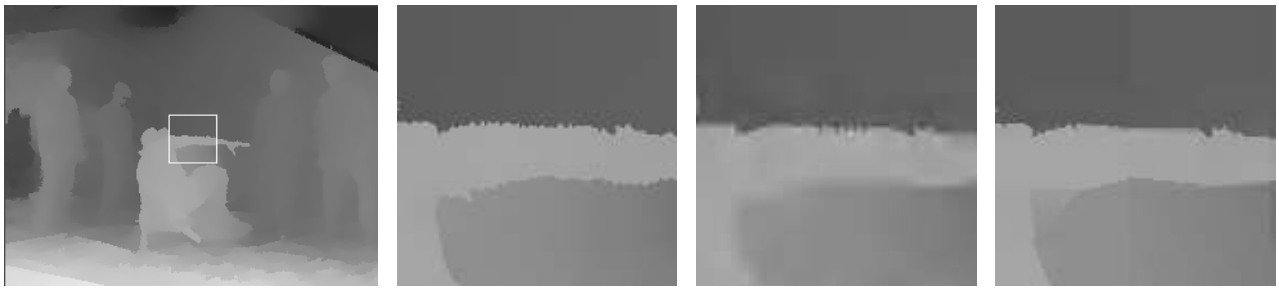


(a) (b)

Figure 10. Rate-distortion curves for the "Teddy" 10(a) and "Breakdancing"10(b) depth images, both for our algorithm and JPEG-2000.

7. CONCLUSIONS

We have presented a new algorithm for coding depth maps that exploits the smooth properties of depth signals. Regions are modeled by 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: the level of the quadtree segmentation (variable block size), the overall coefficient-quantization setting for the image and the choice the modeling function. All three aspects are controlled by a Lagrangian cost function, in order to impose a global rate-distortion constraint for the complete image. For typical bit-rates (i.e. between



(a) (b) (c) (d)

Figure 11. (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.

0.01 bit/pixel and 0.25 bit/pixel), experiments have shown that the coder outperforms a JPEG-2000 encoder by $1-2$ *dB*. The proposed algorithm is intended to be used in a 3D 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.

REFERENCES

1. S. Roy and I. J. Cox, "A maximum-flow formulation of the n-camera stereo correspondence problem," in *IEEE Int. Conf. on Comp. Vision*, pp. 492–502, 1998.
2. D. Scharstein and R. Szeliski, "High-accuracy stereo depth maps using structured light.," in *IEEE Comp. Society Conf. on Comp. Vision and Pattern Recognition (CVPR)*, **1**, pp. 195–202, June 2003.
3. 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.* **3**, pp. 828–831, October 2001.
4. C. Fehn, K. Schuur, P. Kauff, and A. Smolic, "Coding results for EE4 in MPEG 3DAV." ISO/IEC JTC 1/SC 29/WG 11, MPEG03/M9561, March 2003.
5. D. Tzovaras, N. Grammalidis, and M. Strintzis, "Disparity field and depth map coding for multiview image sequence," in *IEEE Int. Conf. on Image Proc.*, **2**, pp. 887–890, 1996.
6. B.-B. Chai, S. Sethuraman, and H. S. Sawhney, "A depth map representation for real-time transmission and view-based rendering of a dynamic 3D scene," in *First Int. Symposium on 3D Data Proc. Visualization and Transmission*, pp. 107–114, June 2002.
7. D. Donoho, "Wedgelets: nearly minimax estimation of edges," *Annals of Statistics* **27**, 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 Transactions on Medical Imaging* **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 Transactions on Image Proc.* **14**, pp. 343–359, March 2005.
10. Y. Morvan, D. Farin, and P. H. N. de With, "Novel coding technique for depth images using quadtree decomposition and plane approximation," in *Visual Communications and Image Proc.*, July 2005.
11. P. A. Chou, T. D. Lookabaugh, and R. M. Gray, "Optimal pruning with applications to tree-structured source coding and modeling.," *IEEE Transactions on Information Theory* **35**(2), pp. 299–315, 1989.
12. 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* **23**(3), pp. 600–608, 2004.
13. M. Adams and F. Kossentini, "Jasper: A software-based JPEG-2000 codec implementation," in *IEEE Int. Conf. on Image Proc.*, **2**, pp. 53–56, October 2000.