Novel Coding Technique for Depth Images using Quadtree Decomposition and Plane Approximation

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

[a] Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
[b] LogicaCMG, P.O. Box 7089, 5605 JB Eindhoven, The Netherlands
y.morvan@tue.nl

ABSTRACT

An efficient way to transmit multi-view images is to send the texture image together with a corresponding depth image. The depth image specifies the distance between each pixel and the camera. With this information, arbitrary views can be generated at the decoder. In this paper, we propose a new algorithm for the coding of depth images that provides an efficient representation of smooth regions as well as geometric features such as object contours. Our algorithm uses a segmentation procedure based on a quadtree decomposition and models the depth image content with piecewise linear functions. We achieved a bit-rate as low as 0.33 bit/pixel, without any entropy coding. The attractivity of the coding algorithm is that, by exploiting specific properties of depth images, no degradations are shown along discontinuities, which is important for perceived depth.

Keywords: Stereo image coding, 3D coding, depth image coding, platelets functions

1. INTRODUCTION

The upcoming 3-D display technology allows to present images with the illusion of depth. These displays show several views of the same scene, viewed from different positions at the same time. An independent transmission of these views has several disadvantages. First, it is inefficient, since there is a strong correlation between the set of images depicting the same scene. Second, different displays will support a varying number of views, which makes it impractical to prepare the displayed views at the encoder. Instead, the views should be synthesized at the decoder, knowing the display geometry. The independent transmission of several views can be avoided by transmitting the texture data independent from the geometry data. One approach is to send the texture data for the central view and a depth image that specifies the depth of each pixel in the texture image. This depth image can be represented by a gray-scale image where dark and bright pixels correspond to far and near pixels, respectively. Based on this depth image, arbitrary views can be synthesized at the decoder. For efficient transmission, the coding of depth images needs to be addressed.

The characteristics of depth images differs from normal video signals. For example, it can be seen in Figure 1*, that large parts of typical depth images usually show smooth objects and other approximately planar surfaces. As a result, the input depth image contains various areas of linear depth changes, corresponding to surfaces of objects. Furthermore, at the objects boundaries, the depth image shows step functions, i.e. sharp edges. Therefore, we are interested in a compression algorithm which can efficiently decompose a typical depth image into regions of linear depth changes, which are bounded by sharp edges.

In previous work, one of the approaches is to use a wireframe modeling technique[2] to code depth images. This technique divides the image into triangular patches, which are approximated by linear functions. If the data cannot be represented with a single linear function, smaller patches are used for such an area. However, the patch structure is not adapted to the image content, such that a large number of small patches is generated along edges. An alternative technique is based on a transform-based algorithm derived from MPEG-4 coders. Key advantage of using a standardized video coding algorithm to compress depth images is the backward compatibility with the existing technology. However, DCT transform coders have shown significant shortcomings for representing edges without deterioration. Consequently, such a coder generates ringing artifacts along edges that yield errors

*The image samples in this paper were taken from the Middlebury Stereo Vision Page "www.middlebury.edu/stereo" which were obtained using a structured light technique.[1]

Visual Communications and Image Processing 2005, edited by Shipeng Li, Fernando Pereira, Heung-Yeung Shum, Andrew G. Tescher, Proc. of SPIE Vol. 5960 (SPIE, Bellingham, WA, 2005) · 0277-786X/05/$15 · doi: 10.1117/12.631647


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

in pixel depth, and therefore appear as a blurry cloud of pixels along the object borders. For this reason, we focus on alternative coding algorithms. Required properties of the desired algorithm are that:

- regions of linear depth changes should be compactly represented and that
- it should efficiently capture and accurately approximate geometric features such as sharp edges.

This has brought us to the concept of modeling the signal using piecewise linear functions. The image is subdivided with a quadtree decomposition and an independent linear function model is used for each leaf of the tree. Additionally, we provide a mode that describes object contours, i.e. depth discontinuities, with a straight line separating two linear models. This last mode enables to code depth discontinuities, without introducing many small leafs along edges. Experimental results show that prior to entropy coding, the new proposal gives attractive low bit-rates for natural images, while simultaneously preserving the quality of the edges.

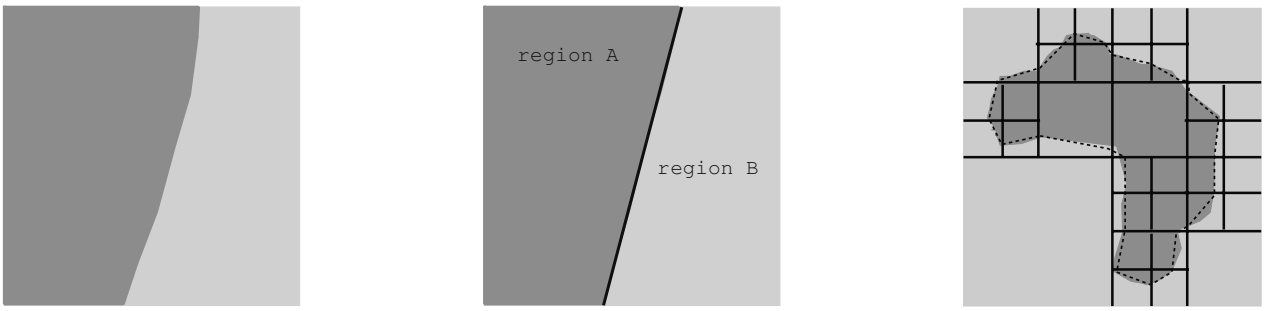
The sequel of this paper is structured as follows. Section 2 examines two compression algorithms and their potential efficiency for coding depth images. Section 3 describes the framework of our depth image coding algorithm, while Section 4 provides further details about the parameter estimation for each coding mode. In Section 5, we describe a rate-distortion optimization of the quadtree decomposition. Results are presented in Section 6 and the paper concludes with Section 7.

2. APPROXIMATION OF OBJECTS CONTOURS AND SURFACES

In this section, we focus on finding a compression scheme which can compactly represent smooth regions as well as boundaries (contours) of objects. We investigate two algorithm alternatives to standard transform coders, namely, the *wedgelet*[3] and *platelet*[4] decomposition.

2.1. Wedgelet decomposition

For efficient approximation of edges, the wedgelet decomposition provides a suitable framework. A wedgelet function is defined as two piecewise constant functions separated by a straight line (illustrated in Figure 2(b)). The wedgelet attempts to approximate a discontinuity (see Figure 2(a)) between two image areas. This discontinuity is captured by the wedgelet by aligning the subdivision line along the discontinuity. The wedgelet separates the image along this line into two regions A and B. These two regions, called *wedges*, are approximated by one gray level each, corresponding to the pixel average within the region. When the approximation quality is not sufficient, the image is subdivided into four sub-images. This results finally in a quadtree decomposition, where the image is segmented into a variable number of square sub-images, depending on the image details. As seen in



(a) (b) (c)

Figure 2. An image presenting an edge (Fig. 2(a)) can be approximated by a wedgelet function (Fig. 2(b)). When the image cannot be approximated with sufficient quality, it is recursively subdivided into four squares, resulting in a quadtree decomposition (Fig. 2(c)) of the image.

Figure 2(c), each leaf of the tree is approximated by a wedgelet function. Because edges are modeled by straight lines, the object contours remain sharp. However, object surfaces typically show a slowly varying gray level, i.e. image gradient, in the depth image. Considering that a wedgelet function approximates the image content by a single gray level, object surfaces are therefore poorly represented. For this reason, we investigate an alternative modeling that concentrates on the approximation of planar surfaces.

2.2. Platelet decomposition

As previously discussed, objects surfaces typically show a gray level gradient in the depth image. Because a gradient can be modeled by a planar surface, we discuss the so-called platelet function, which approximates an image by a linear function. A platelet function $f_S(x, y)$ defines on a square image or a wedge in the covering area S (support) can be described by a linear function

$$f_S(x, y) = (a_S x + b_S y + c_S), \quad \text{for all } (x, y) \in S, \tag{1}$$

where (a_S, b_S, c_S) are parameters which are adjusted to the image content and S indicates the support of the platelet function. Based on this definition, an image $I(x, y)$ presenting one grayscale gradient can be approximated by a single platelet function (illustrated in Figure 3(a)). Since this platelet function approximates a square image $I(x, y)$, it is called *a square platelet* and denoted as f_I. A square platelet function may be written as Equation 1 with subscripts I instead of S. Furthermore, an image depicting an edge is approximated by two platelet functions $f_A(x, y)$, $f_B(x, y)$, separated by a straight line, and is called a *wedged platelet*. Equivalently to the wedgelet decomposition, the line defines two regions A and B corresponding to the support of a platelet function. Accordingly, a wedged platelet is specified by

$$f_{wedged}(x, y) = \begin{cases} f_A(x, y) = a_A x + b_A y + c_A & (x, y) \in A, \\ f_B(x, y) = a_B x + b_B y + c_B & (x, y) \in B. \end{cases} \tag{2}$$

When the platelet function does not provide a sufficient quality, the image is subdivided into four smaller squares. Each leaf of the resulting quadtree is then approximated with either a square platelet or a wedged platelet.

A platelet is potentially more efficient than a wedgelet, since it offers both, smooth gradients and sharp edges. In the remainder, we will therefore focus on the platelet decomposition.

3. DEPTH IMAGE CODING ALGORITHM

In this section, we present a novel approach for depth coding using the above-mentioned platelet functions. With a single linear function, we can represent one planar surface of the scene, like the ground plane, walls, or

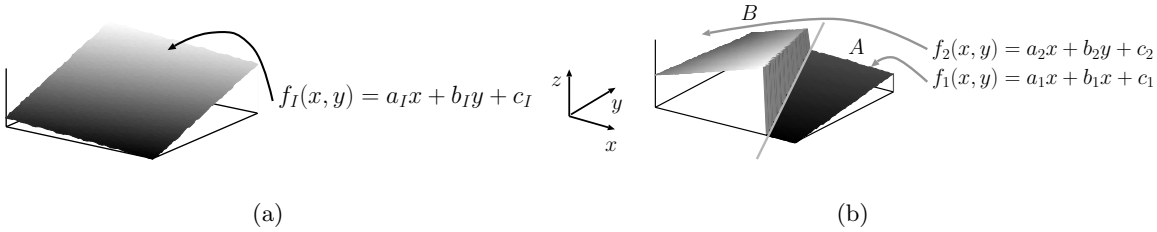


Figure 3. Fig. 3(a) and Fig. 3(b) depict a square platelet and a wedged platelet function, respectively.

small objects. Hence, a single function can cover areas of variable size in the image. To identify the location of these planar surfaces in the image, we employ a quadtree decomposition which recursively divides the image into squares of different size. In some cases, the depth image within one square can be approximated with a single linear approximation. If no suitable approximation can be determined for the square, it is subdivided into four smaller squares. Additionally to this standard quadtree subdivision, we apply a special coding mode if there are discontinuities in the depth image. To prevent that many small squares are required along a discontinuity, we separate the square along a straight edge into two regions. Each of these two regions is coded with a separate linear function. Consequently, the coding algorithm chooses between three possible modes for each node in the quadtree:

- *Mode 1*: Approximate the square content with a single linear function: a square platelet;
- *Mode 2*: Subdivide the square along a straight line into two regions and approximate each region with an independent linear function: a wedged platelet;
- *Mode 3*: If the previous two modes fail, then subdivide the square into smaller ones and recursively process these squares.

The decision for each coding mode is based on a rate-distortion optimization described in Section 5. To illustrate our approach, let us examine the natural depth image "Teddy" as an example (see Figure 4(b)). Three squares are highlighted that will be coded with the three described modes. The top square only comprises a wall and can thus be coded with a single model. The left square shows part of a planar object and the background, separated by an edge. Here, we can use *Mode 2* with one model for the first object and another one for the background. The square at the center of the image shows complex content and has to be subdivided. After this subdivision, each resulting node shows one or two regions, so that the node can be coded using either *Mode 1* or *Mode 2*. For an approximation of higher quality, a node can be subdivided with an additional recursive iteration.

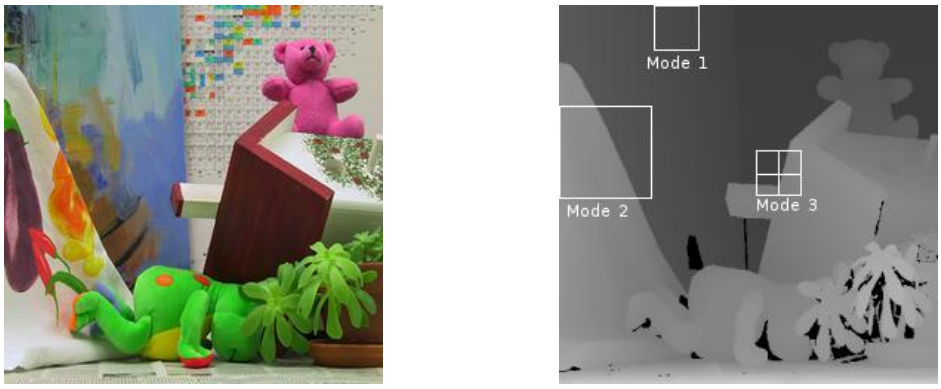
4. PARAMETER ESTIMATION FOR CODING MODES 1 AND 2

This section explains the computations of the parameters that are used in the two coding modes. For *Mode 1*, a single linear function is used for which the three parameters a, b, c with subscript I of Equation 1 should be calculated. For *Mode 2*, the location of the separation line and the two sets of parameters a, b, c from Equation 2 should be computed.

4.1. Parameter estimation for *Mode 1*

In order to determine the three parameters a_I, b_I, c_I of the linear function $f_I(x,y) = a_I x + b_I y + c_I$, a least-squares fitting is used. This optimization minimizes the sum of squared differences between the depth image $I(x,y)$ and the proposed linear model. Accordingly, parameters a_I, b_I, c_I are determined such that the error

$$E(a_I, b_I, c_I) = \sum_{x=1}^{n} \sum_{y=1}^{n} (a_I x + b_I y + c_I - I(x,y))^2$$



(a) (b)

Figure 4. Example input image "Teddy" (Fig. 4(a)) and the corresponding depth image (Fig. 4(b)). Three squares are marked for which different coding modes are used.

is minimized, where n denotes the node size in term of pixels. This error function $E(a_I, b_I, c_I)$ is minimized when the gradient satisfies $||\nabla E|| = 0$. When taking the partial derivatives with respect to a_I, b_I and c_I of this equation, we find a set of linear equations specified by

$$\begin{pmatrix} t & u & v \\ u & t & v \\ v & v & n^2 \end{pmatrix} \begin{pmatrix} a_I \\ b_I \\ c_I \end{pmatrix} = \begin{pmatrix} \sum_{x=1}^{n} \sum_{y=1}^{n} xI(x,y) \\ \sum_{x=1}^{n} \sum_{y=1}^{n} yI(x,y) \\ \sum_{x=1}^{n} \sum_{y=1}^{n} I(x,y) \end{pmatrix},$$

with

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

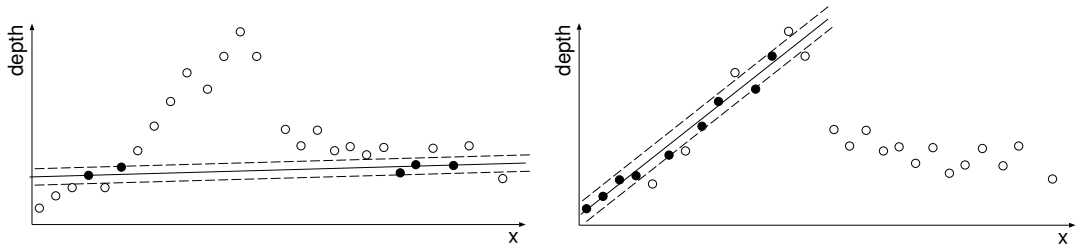
Since the matrix on the left side of the previous equation system is constant, it is possible to pre-compute and store the inverse for each size of the square (quadtree node). This enables to compute the parameters a_I, b_I, c_I with a simple matrix multiplication.

4.2. Parameter estimation for *Mode 2*

For *Mode 2*, we have to determine not only the model parameters, but also the separating line. This cannot be solved with a least-squares minimization for the following reason. Without knowing the subdivision boundary between the two regions, it is not possible to determine uniquely their model parameters. Similarly, it is not possible to identify the subdivision line as long as the region parameters are unknown.

4.2.1. Platelet parameters estimation

For this reason, we apply a robust estimation using the RANdom SAmple Consensus (RANSAC[5]) algorithm. This algorithm can estimate parameters even if the data is disturbed by a large number of outliers. In our case, we assume that the data comprises pixels from two regions with different models. We consider the data corresponding to the larger region as the inlier data, while the data in the smaller region is considered as outliers. Our algorithm starts with estimating the parameters for the larger region in a first run of the RANSAC algorithm. To this end, the algorithm randomly selects three pixels, which are used to compute candidate parameters $\hat{a}_A, \hat{b}_A, \hat{c}_A$. The algorithm then counts the number of pixels that fit to this model (inliers). This procedure is repeated several times (see a one-dimensional example in Figure 5) and finally, the parameters with the largest support are selected as the parameters for region A (see Figure 6). Using the obtained parameters for region A, we remove all inlier pixels from the data. On this reduced data set, we carry out a second RANSAC parameter estimation. As a result, we obtain the parameters for the smaller region B from the second RANSAC estimation.



(a) (b)

Figure 5. Two samples are randomly selected to compute a candidate model indicated by the line and input data close to the line candidate is considered as inliers (black dots). Fig. 5(a) and Fig. 5(b) show two candidate models with a small and a high number of inliers, respectively. The model with the larger number of inliers Fig. 5(b) is selected.

4.2.2. Estimation of the subdivision line parameters

The next phase is the determination of the parameters of the subdivision line. In the sequel, the subdivision line is seen as an edge in the depth image. To estimate the line parameters, we investigate two approaches: the first is based on the RANSAC estimation algorithm, whereas the second approach uses on a dictionary of edges.

Solution 1: Edge detection. To perform a fast estimation of the edge orientation and location, we mark the pixels with the model label A or B and we detect edges in this classification map (see Figure 6). Since this classification can contain errors, we use a RANSAC procedure to detect the best fitting edge. However, despite the robustness of the RANSAC algorithm, the estimated edge model may not always result as the best (mathematical) fit to the depth image. Consequently, when an unfavorable edge parameters are estimated, geometric artefacts appear in the reconstructed depth image. For this reason, we investigate a second algorithm which yields the best fit to the image content, according to a well-defined criterion.

Solution 2: Dictionary of edges. To obtain a robust estimation of edge parameters, we build a dictionary of wedges. The wedges in the dictionary are generated with all possible subdivision lines that divide the square into two areas. Subsequently, an exhaustive search through the dictionary is performed and the best fitting wedge is selected. To evaluate the wedge fit, we use the quality of the proposed model, i.e. the L_2 distance between the model and the image data. More formally, the best edge model minimizes the approximation error

$$\min_{(A,B)\in D} \|f_{wedged}(x,y) - I(x,y)\| = \min_{(A,B)\in D} \left(\sum_{(x,y)\in A} (f_A(x,y) - I(x,y))^2 + \sum_{(x,y)\in B} (f_B(x,y) - I(x,y))^2 \right),$$

where the parameters for f_A and f_B are computed according to Section 4.2.1.

5. RATE DISTORTION ANALYSIS

This section describes the subdivision criterion of the node in the quadtree that we apply to optimize the rate-distortion behavior of the quadtree. We will only outline the algorithmic principle and omit detailed discussions about optimality. Furthermore, we have omitted the use of entropy coding, so that the bit-rate calculation is relatively simple.

The guiding principle is the application of a Lagrangian cost function[6] $D+\lambda R$ where D denotes the distortion resulting from the quadtree decomposition in the pixel domain and R stands for the bit-rate required to code the sub-image with the corresponding parameters (e.g. platelets parameters). For each coding mode, a corresponding

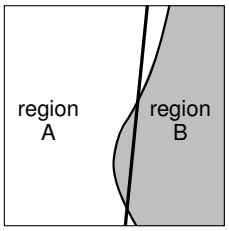


Figure 6. Regions A and B are identified with a classification map of inliers and outliers pixels, the boundary between both regions being approximated with a straight line.

cost (distortion and rate) can be defined. If the node is subdivided in smaller squares, the cost of this node includes the sum of the children node costs. To make an optimal decision for one node, we select the coding mode that gives the lowest Lagrangian cost. Because the cost depends on the cost of children nodes, an optimal solution can be found by computing the coding modes by a bottom-up traversal of the tree. However, this optimal solution is too computationally expensive. Instead, we propose an approximation that can be computed by a top-down traversal of the tree. The algorithm can be summarized as follows.

- **Step 1:** Compute the Lagrangian cost for coding *Mode 1* and *2*.
- **Step 2:** Compute the Lagrangian costs for the children nodes and select temporarily the modes of the children with the minimum cost.
- **Step 3:** For the current node, choose the best coding mode based on the cost of *Mode 1*, *Mode 2* and the sum of the children costs.

This top-down traversal of the tree is more computationally efficient since we can stop the subdivision as soon as *Mode 1* or *Mode 2* is selected.

6. EXPERIMENTAL RESULTS

For evaluating the performance of the algorithm, experiments were carried out using the depth images of the "Teddy" and "Cones" scenes. Experiments have revealed that the proposed algorithm can code large areas of the image with a single node. Examples are the top left nodes in Figure 7(c) and Figure 7(f).

With respect to the obtained bit rate, the following can be concluded. We have made a conservative estimate of the bit rate, assuming that the coding of the quadtree and the choice of coding modes 1-3 requires two bits per node. Moreover, we assume that model and line parameters require 8 bits each. In total, we get 24 bits for *Mode 1* and 64 bits for *Mode 2*. This conservative bit-rate estimation results in a bit-rate of 0.33 bit/pixel at a PSNR of 32.6 dB for the image "Teddy", and 0.47 bit/pixel at a PSNR of 33.62 dB for the image "Cones". The improvements that can be obtained by entropy coding are under study.

7. CONCLUSION

We have presented an algorithm for coding depth images that exploits the smoothness properties of such a signal. Regions are approximated using piecewise linear functions and they are separated with straight lines along their boundaries. The length of a line segment (and thus the distortion) depends on the level of the quadtree decomposition in the actual area of the depth image. The algorithm allows the coding of small details as well as large regions within a single node. The bit-rate range, without entropy coding, is between 0.33 bit/pixel and 0.47 bit/pixel for the "Teddy" and "Cones" image, respectively. The performance of the algorithm is controlled by a computationally efficient top-down approach in which Lagrangian cost functions for individual coding modes are evaluated and coding modes giving minimum costs are selected at each step of the decomposition.

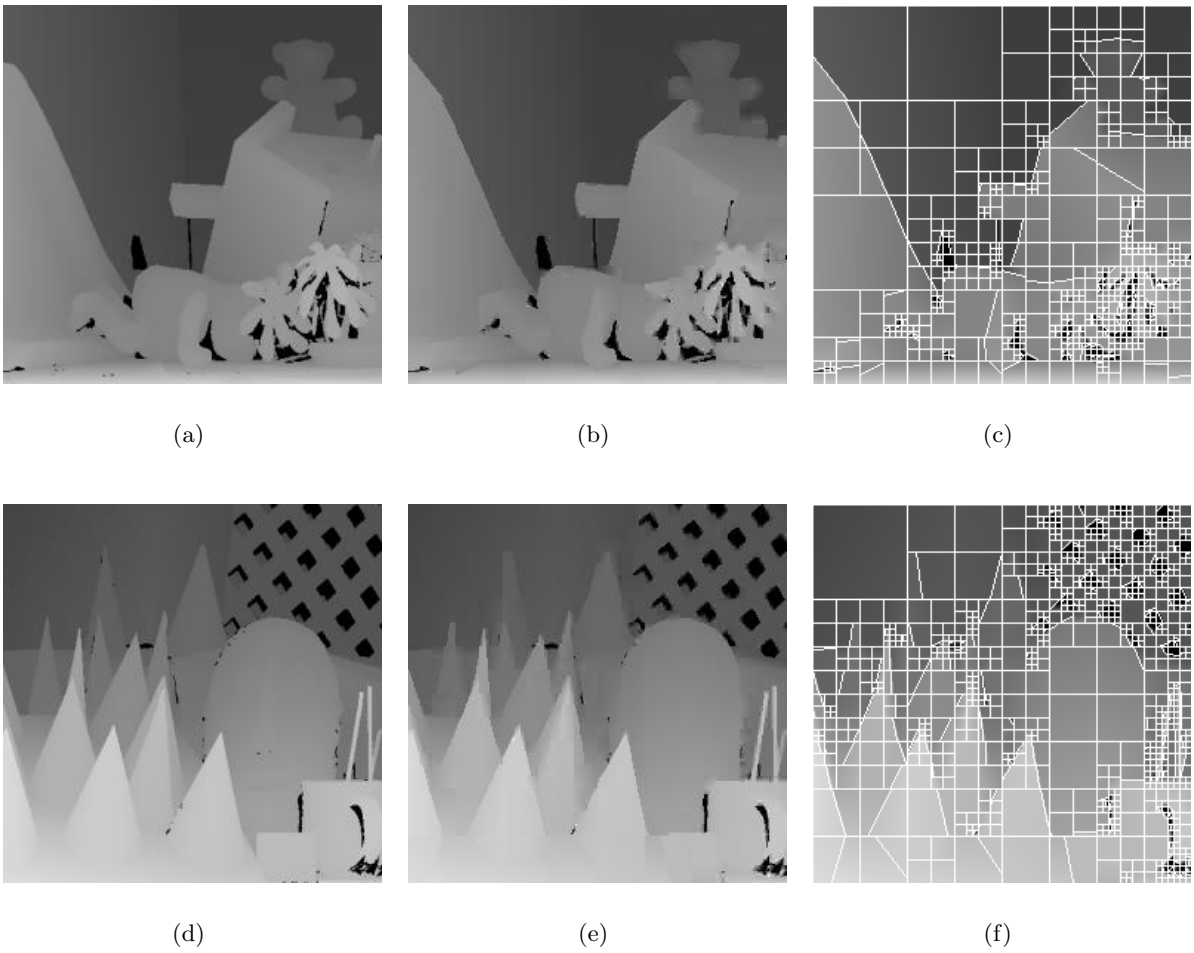


Figure 7. Fig. 7(a) and Fig. 7(d) show the original depth images "Teddy" and "Cones", the corresponding reconstructed depth images are Fig. 7(b) and Fig. 7(e), respectively. The superimposed nodes of the quadtree are portrayed by Fig. 7(c) and Fig. 7(f). Coding for the depth images "Teddy" and "Cones" is achieved at a bit rate of 0.33 bit/pixel and 0.47 bit/pixel for a PSNR of 32.6 dB and 33.62 dB, respectively.

REFERENCES

1. D. Scharstein and R. Szeliski., "High-accuracy stereo depth maps using structured light.," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, **1**, pp. 195–202, June 2003.
2. D. Tzovaras, N. Grammalidis, and M. Strintzis, "Disparity field and depth map coding for multiview image sequence," in *Proceedings on Int. Conf. Image Processing*, **2**, pp. 887–890, 1996.
3. D. Donoho, "Wedgelets: nearly minimax estimation of edges," *Annals of Statistics* **27**, pp. 859–897, March 1999.
4. R. Willett and R. Nowak, "Platelets for Multiscale Analysis in Photon-Limited Imaging," in *IEEE International Conference on Image Processing*, **1**, pp. 365–368, (Rochester, NY), September 2002.
5. M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM* **24**(6), pp. 381–395, 1981.
6. A. Ortego and K. Ramchandran, "Rate-distortion methods for image and video compression," *IEEE Signal Processing Magazine* **15**, pp. 23–50, 1998.