

# DEPTH-IMAGE REPRESENTATION EMPLOYING MESHES FOR INTERMEDIATE-VIEW RENDERING AND CODING

Dirk Farin<sup>1</sup>, Rick Peerlings<sup>1</sup>, and Peter H. N. de With<sup>1,2</sup>

<sup>1</sup> University of Technology Eindhoven  
5600 MB Eindhoven, Netherlands  
d.s.farin@tue.nl  
rjjpeerlings@gmail.com

<sup>2</sup> LogicaCMG Netherlands  
PO Box 7089, TSE Eindhoven  
5605 JB Eindhoven, Netherlands  
p.h.n.de.with@tue.nl

## ABSTRACT

A common way to represent 3-D images is to send the texture-image together with a corresponding depth-image. This paper presents a depth-image representation employing triangular meshes to benefit both in obtaining higher compression factors and directly enabling fast intermediate-view rendering on commodity graphics hardware. Previous mesh-based representations did not consider the fact that assigning wrong depth values along object boundaries results in well-observable artifacts in the intermediate-view rendering. We solve this problem by assigning *two* depth values per node to better model sharp depth discontinuities. Furthermore, we ensure that mesh-edges are placed along object borders, preventing that triangles span background as well as objects at the same time. The results show good intermediate-view rendering quality, object/background separation, and good rate-distortion performance up to about 40 dB PSNR at a bit rate of 0.05 bits/pixel.

**Index Terms**—mesh generation, image coding, stereo vision, three-dimensional displays.

## 1. INTRODUCTION

A *mesh* is a division of a 2-D domain into non-overlapping polygons [1]. Usually, the mesh is composed out of triangles to simplify the computations. Meshes are commonly used to represent depth-images by assigning a disparity<sup>1</sup> value to each node in the mesh. The values within the triangle area are then linearly interpolated from the disparity at its nodes. Because this significantly reduces the number of samples to be sent, high compression ratios can be achieved. Furthermore, the triangulation of the depth-image allows an intermediate-view rendering implementation on commodity graphics hardware, thereby providing real-time rendering as a bonus.

There are different ways to construct a mesh. Most techniques base the mesh construction on the texture images, thereby implicitly computing the depth during mesh construction [2]. Because depth-estimation by itself is a complicated problem,

the disadvantage of these mesh-construction techniques is that the resulting depth-image has a low quality. A better approach is an algorithm based on two stages. First, a depth-image is generated from the multiview images with some specialized algorithm [3]. In a second step, this depth-image is represented with a triangular mesh [4]. Using this two-step approach, the depth-image representation can be more accurate and depth discontinuities can be sharply modeled, because they are readily discernible in the depth-image.

The algorithm proposed in this paper also employs the two-step approach. However, we extend the usual triangular-mesh representation with two new contributions:

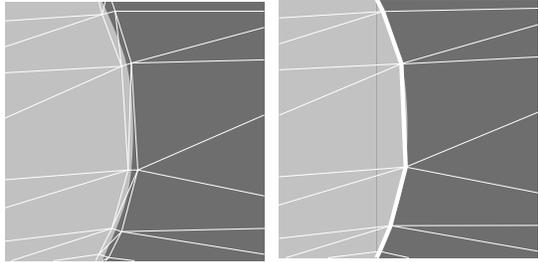
- **Two disparities per node:** It is essential to maintain disparity discontinuities in order to avoid connecting independent objects to each other. This effect cannot be achieved with a fully connected mesh. Hence, we allow for two disparity values per node, such that triangles connected to a node can choose to use either of the disparity values.
- **Edge placement along discontinuities:** It must be ensured that the mesh is constructed such that triangles do not span across the discontinuity. We achieve this by constraining the triangulation such that it always includes edges along the discontinuity.

The paper is organized as follows. First, Section 2 presents the construction of the mesh based on the depth-image. Section 3 proposes a coding scheme for the obtained data, and we estimate the bit rate required to code a depth-image using our representation. Finally, Section 4 presents results and Section 5 draws conclusions.

## 2. ALGORITHM TO CONSTRUCT THE DEPTH-IMAGE MESH

We have observed that depth-images usually comprise planar gradients corresponding to objects, separated by sharp discontinuities corresponding to object borders. For the subject-

<sup>1</sup>In this paper, we use the terms *depth* and *disparity* interchangeably.



(a) Using narrow triangles. (b) Using polylines (bold line).

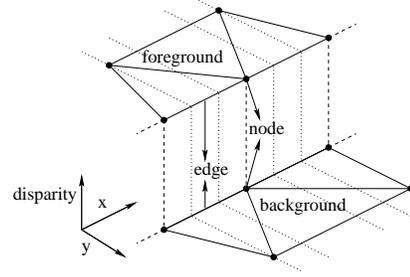
**Fig. 1.** Two methods to model discontinuities.

tive quality of the intermediate-view rendering, the exact reconstruction of the object borders is more critical than the exact disparities within the objects [5]. Therefore, our algorithm focuses on modeling the discontinuities and not on modeling the small detail within objects. Mesh nodes are placed exclusively on object borders.

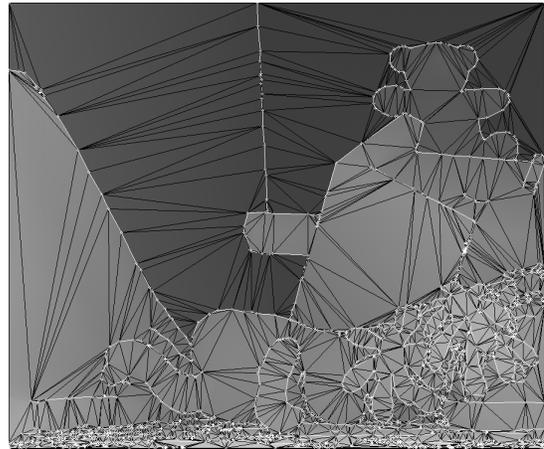
One way of handling disparity discontinuities is to place narrow triangles along them, as shown in Figure 1(a). Nodes are then placed closely along the foreground and background side of the discontinuity. The high gradient in the narrow triangles simulates the discontinuity. The narrower these triangles are, the better the approximation of the discontinuity. When their width is reduced, their gradients become steeper, until they reduce to lines and their gradients become infinity, as shown in Figure 1(b). The proposed algorithm uses these idealized polylines to represent the degenerated triangles.

These polylines are a fixed part of the triangulation and as such ensure that no triangles cross the discontinuities. The additional mesh-edges are added with a constrained Delaunay triangulation algorithm. Each polyline node receives two disparity values [6]: one for the foreground object and one for the background. Having two disparities per node, the foreground triangles can use the foreground disparity, while the background triangles can use the background disparity, as shown in Figure 2. This implicitly disconnects the mesh along the depth discontinuities, which prevents that triangles around them will stretch or squeeze in the intermediate-view rendering.

The algorithm proceeds in the following steps. First, a *discontinuity detector* finds depth discontinuities in the depth-image, resulting in an *edge-image*. Afterwards, a *chain detector* finds chains of connected edge pixels and a *polyline approximator* models them with a sequence of straight line segments. A *constrained Delaunay triangulation* starts with the line segments obtained in the last step and adds more edges to obtain a full triangulation. Finally, a *node-disparity computation* calculates the two disparity values for each node such that the modeled depth-image can approximate well the input depth-image. Furthermore, it decides for each node of a triangle whether to use the foreground or background disparity.



**Fig. 2.** Two disparities per node at discontinuities. The triangles are unconnected.



**Fig. 3.** Mesh superimposed on the ground-truth depth-image of the teddyH stereo data set. The white and black edges resulted from the polyline approximator and the constrained Delaunay triangulation, respectively.

## 2.1. Mesh construction

Mesh construction starts with finding the discontinuities in the depth-image. We have adopted the Canny edge detector for this purpose because it gives one-pixel wide lines. Connected edge-pixels are collected into chains and if needed, chains are separated at T-junctions. Note that now every edge-pixel has exactly two neighbors except at the start and end, where there is only one neighbor. Each chain is then approximated by a sequence of line segments, which then constitutes a *polyline* along the discontinuity. Starting from these polylines, the complete mesh is built using a constrained Delaunay triangulation. *Constrained* means that the polylines predefine some edges of the triangulation.

## 2.2. Node-disparity computation

After the mesh geometry is defined as described in the previous section, two disparities are computed for each node. First, a linear function  $d'(x, y) = \alpha x + \beta y + \gamma$  is fitted with a least-squares algorithm to the disparity values  $d(x, y)$  in each tri-

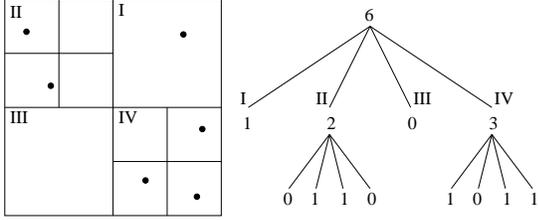


Fig. 4. Quad-tree representation of end-point positions.

angle area, where  $\alpha$ ,  $\beta$ , and  $\gamma$  are the function parameters [7]. Instead of using these function parameters, we usually work with the disparity values at the nodes. Note that the three disparity values at the node positions are sufficient to recompute the parameters.

Second, the disparities of all the triangles incident at one node are clustered into two sets. The two final disparities that are assigned to the node are defined as a weighted average value over each of the two sets. The weight for each disparity is proportional to the area of the corresponding triangle.

An example result of the obtained mesh geometry is depicted in Fig. 3.

### 3. COMPRESSION OF THE DEPTH-IMAGE MESH

The mesh representation is also attractive for compression because the nodes sample only a very limited number of disparity values. The remaining disparity values are filled-in nicely by the interpolated gradients.

Instead of sending the mesh node positions, our coder sends the chains of discontinuity edges. When employing the same polyline approximator as used in the encoder, the decoder can recover the node positions and afterwards the full mesh geometry, again using a constrained Delaunay triangulation. Sending the full edge-image has the advantage that the decoder can choose to approximate it with a polyline of higher accuracy, resulting in a more exact definition of the object boundary.

The discontinuity edges are coded as follows. First, the two end-points of all chains are transmitted, and afterwards, the pixel-accurate chains between a pair of points are sent.

The chain end-points are coded jointly using a quad-tree structure. The root node represents the entire image, the four child nodes subdivide the area of the parent node into four equally-sized sub-areas. Each quad-tree node is attributed with the number of end-points within its area. Starting with the root node, the quad-tree recursively sub-divides the image area into four sub-images until each leaf node contains either one or zero end-points. Figure 4 shows an example of such a quad-tree division.

To store the quad-tree, the number of end-points in each node of the quad-tree is saved, starting with the number of end-points in the root node, which is coded with a fixed num-



Fig. 5. Reconstructed depth-image of the teddyH image.

ber of bits. The number of end-points in each of its child nodes can be coded efficiently, since it is known that the total number of end-points in all childs is the same as the number of end-points in the parent. If there are  $n$  end-points in the parent's node area, there must be  $\sum_{i=1}^4 n_i = n$  end-points in the four child nodes. Hence, the number of end-points for the first child requires  $\log_2(n+1)$  bits to code, but for the second child, only  $\log_2(n+1-n_1)$  bits are required. The third child can be saved with  $\log_2(n+1-n_1-n_2)$  bits, and no bits are needed for the last child, because we know that the numbers  $n_i$  have to sum up to  $n$ .

After receiving the quad-tree, it is known in which area each end-point is, but not its exact position. Therefore, the position of each end-point relative to the sub-image is stored. The number of bits needed to store such a position is  $\log_2 w + \log_2 h$ , with  $w$  and  $h$  being the dimensions of the sub-image. Knowing the edge-image, the decoder can use the same algorithm as the encoder to approximate the polylines and finally to compute the mesh geometry.

The node disparities are transmitted differentially to the previous node along the polylines. Clearly, foreground and background node disparities are predicted independently. The assignment of the triangle nodes to background or foreground disparity is collected in a three-bit mask. Since the assignments are not statistically independent, this three-bit value is coded with an entropy-coder.

### 4. TEST RESULTS

An example result of the proposed algorithm applied to the teddyH image is shown in Figs. 5 and 6. The reconstructed depth-image of Fig. 5 resembles well the ground-truth depth-image (not shown), and the disparity discontinuities are sharp. Reconstruction of smoothly changing gradients inside objects is approximated by linear gradients, which does not lead to a significant degradation in the perceived quality of the inter-



Fig. 6. Left-view rendering of the teddyH stereo image.

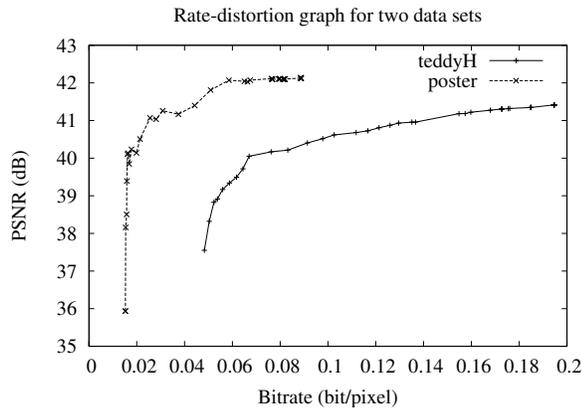


Fig. 7. Rate-distortion graphs for two data sets.

mediate views.

The view rendering of Fig. 6 is good since the foreground is separated from the background, leaving blank spaces in the occlusion areas. The rendering quality inside objects is also good, and only small imperfections of the mesh disconnection are visible.

Fig. 7 shows the resulting rate-distortion curves for two data sets. They were obtained by varying the accuracy of the polyline approximator, which affects the number of nodes generated along the chains in the edge-image. We measured the Peak Signal-to-Noise Ratio (PSNR) between the ground truth depth-image and the reconstructed depth-image, well knowing that the perceived quality would be higher than suggested by the PSNR values. The bit rate is computed using the compression technique of Section 3.

For teddyH, the algorithm is capable of reaching a PSNR of 40.1 dB at a rate of 0.056 bits/pixel. For comparison, JPEG compression reaches 40.6 dB at 0.056 bits/pixel, but the subjective quality of JPEG is lower. Moreover and more important, after decompressing the mesh representation, the

obtained data can be used directly for intermediate-view rendering, thereby providing an efficient framework for decoding and display.

## 5. CONCLUSIONS

This paper has presented a depth-image representation employing triangular meshes. It benefits both by higher compression and directly enables fast intermediate-view rendering on commodity graphics hardware.

Previous mesh-based representations overlooked the fact that assigning erroneous disparity values along object boundaries results in well-observable artifacts in the intermediate-view rendering. We have solved this problem by assigning two disparity values per node to better model depth discontinuities, and we ensured that mesh-edges are placed along object borders. This prevents that triangles span background as well as objects at the same time, which would lead to a wrong intermediate-view rendering.

The reconstruction results we obtained with the proposed method are good and outperform JPEG coding. The intermediate-view rendering quality is high and the object separation using mesh disconnection works well. The depth-image reconstruction is accurate, and depth discontinuities remain sharp. The rate-distortion performance is high, around 40 dB at 0.05 bits/pixel has been achieved. Future research may improve the method by using additional nodes within objects to improve the reconstruction of complex, bent object surfaces.

## 6. REFERENCES

- [1] Y. Wang and O. Lee, “Active mesh—a feature seeking and tracking image sequence representation scheme”, *IEEE Trans. Image Process.*, vol. 3, no. 5, pp. 610–624, September 1994.
- [2] M. Kardouchi, J. Konrad and C. Vázquez, “Estimation of large-amplitude motion and disparity fields: Application to intermediate view reconstruction”, *Proc. SPIE*, vol. 4310, pp. 340–351, 2001.
- [3] D. Scharstein and R. Szeliski, “A taxonomy and evaluation of dense two-frame stereo correspondence algorithms”, *International Journal of Computer Vision*, vol. 47, no. 1–3, pp. 7–42, April 2002.
- [4] B.-B. Chai, S. Sethuraman, H.S. Sawhney and P. Hatrack, “Depth map compression for real-time view-based rendering”, *Pattern Recognition Letters*, vol. 25, no. 7, pp. 755–766, May 2004.
- [5] R. Krishnamurthy, B.-B. Chai, H. Tao, S. Sethuraman, “Compression and transmission of depth maps for image-based rendering”, *Proc. ICIP*, vol. 3, pp. 828–831, 2001.
- [6] M. Jansen, R. Baraniuk, and S. Lavu, “Multiscale approximation of piecewise smooth two-dimensional functions using normal triangulated meshes”, *Appl. Comput. Harmonic Anal.*, vol. 19, no. 1, pp. 92–130, July 2005.
- [7] Y. Morvan, P.H.N. de With and D. Farin, “Platelet-based coding of depth maps for the transmission of multiview images”, *Proc. SPIE*, vol. 6055, pp. 93–100, January 2006.