Depth-Layer-based Multiview Image Synthesis & Coding for Interactive z- and x-dimension View Switching

Yu Mao[a], Gene Cheung[b], Yusheng Ji[b]

[a],The Graduate University for Advanced Studies, [b],National Institute of Informatics

ABSTRACT

In an interactive multiview image navigation system, a user requests switches to adjacent views as he observes the static 3D scene from different viewpoints. In response, the server transmits encoded data to enable client-side decoding and rendering of the requested viewpoint images. It is clear that there exists correlation between consecutive requested viewpoint images that can be exploited to lower transmission rate. In previous works, this is done using a pixel-based synthesis & coding approach for view-switch along the x-dimension (horizontal camera motion): given texture & depth maps of the previous view, texture pixels are individually shifted horizontally to the newly requested view, each according to its disparity value, via depth-image-based rendering (DIBR). Unknown pixels in the disoccluded region in the new view (pixels not visible in the previous view) are either inpainted, or intra-coded and transmitted by server for reconstruction at decoder.

In this paper, to enable efficient view-switch along the z-dimension (camera motion into / out of the scene), we propose an alternative layer-based synthesis & coding approach. Specifically, we first divide each multiview image into depth layers, where adjacent pixels with similar depth values are grouped to the same layer. During a view-switch into the scene, spatial region in a layer is enlarged via super-resolution, where the scale factor is determined by the distance between the layer and the camera. On the other hand, during a view-switch out of the scene, spatial region in a layer is shrunk via low-pass filtering and down-sampling. Due to high quality reconstruction of depth layers in the new view via rescaling, coding and transmission of a depth layer in the new view by server is necessary only in the rare case when the quality of layer-based reconstruction is poor, saving transmission rate. Experiments show that our layer-based approach can reduce bit-rate by up to 35% compared to previous pixel-based approach.

Keywords: Interactive multiview image streaming. Depth-image-based rendering (DIBR). Inpainting. Super-resolution.

1. INTRODUCTION

Among the many cues visually observed in a 3D scene by humans to form a depth perception, *motion parallax*[1]—a viewer's head movements triggering correspondingly shifted viewpoint images of the scene for observation—has been shown to be the strongest.[2] To create this motion parallax visual effect on a conventional 2D display, multiview imaging systems can capture images from different viewpoints of the same 3D scene using multiple closely spaced cameras.[3] Captured views corresponding to the viewer's most recently detected head positions can then be rendered interactively on a 2D display. In this paper, we consider a network streaming version where multiview images are stored at a remote server, and a client successively requests switches to adjacent views as he moves and observes the static 3D scene from different viewpoints. In response, the server transmits encoded data to enable client-side decoding and rendering of the requested viewpoint images.

It is clear that there exists correlation between consecutive requested viewpoint images that can be exploited to lower transmission rate. If depth maps (per-pixel distance between camera and physical objects) from the same camera viewpoints are available* and transmitted along with texture maps, then during a view-switch along the x-dimension (horizontal camera motion), texture pixels of previous view can be individually shifted horizontally to the newly requested view, each according to its disparity value, via depth-image-based rendering (DIBR).[5] Unknown pixels in the disoccluded region in the new view (pixels not visible in the previous view) can

*Depth maps can be captured directly through time-of-flight (ToF) cameras,[4] or indirectly through stereo-matching algorithms.

either be inpainted,[6] or intra-coded using conventional coding techniques like H.263[7] or H.264[8] and transmitted by server, for reconstruction at decoder.

The described *pixel-based* synthesis & coding approach[9] performs poorly for view-switch along the z-dimension (camera motion into / out of the scene), however. During a view-switch into the scene, objects closer to the camera increase in size more rapidly than objects further away. If decoder employs pixel-based synthesis, pixels of the same object in previous view will be scattered far apart in the new view, leaving the newly created pixels in-between difficult to fill using general inpainting techniques.[6]

In this paper, to enable efficient view-switch along the z-dimension, we propose an alternative *layer-based* synthesis & coding approach. Specifically, we first divide each multiview image into depth layers, where adjacent pixels with similar depth values are grouped to the same layer. During a view-switch into the scene, spatial region in a layer is enlarged via super-resolution (SR), where the scale factor is determined by the distance between the layer and the camera. On the other hand, during a view-switch out of the scene, spatial region in a layer is shrunk via low-pass filtering and down-sampling. Due to high quality reconstruction of depth layers in the new view via rescaling (e.g., SR algorithms like Li et al.[10] can preserve edge sharpness), coding and transmission of a depth layer in the new view by server is necessary only in the rare case when the quality of layer-based reconstruction is poor, resulting in saving in transmission rate. Experiments show that our layer-based approach can reduce bit-rate by up to 30% compared to previous pixel-based approach at the same objective quality.

The outline of the paper is as follows. We first briefly discuss related work in Section 2. We then describe our multiview image navigation system in Section 3. We describe how an image is decomposed into depth layers in Section 4. Given the derived depth layers, we describe how coding is performed offline to enable interactive view-switching along z- and x-dimension in Section 5. Experiment results and conclusions are presented in Section 6 and 7, respectively.

2. RELATED WORK

Coding of correlated images for interactive navigation has been studied previously in the context of light field,[11] where correlation between adjacent viewpoint images is exploited using motion-compensated P-frames[7] and distributed source coding (DSC) frames.[12] However, Daribo et al.[9] demonstrated that for view-switches along the x-dimension, the correlation can often be better exploited by first using DIBR to project known pixels from one view to another, followed by inpainting at decoder or intra-coded blocks sent from server to fill in pixels in disoccluded regions. View-switch along the x-dimension corresponds to movement of the observer's head to the left or right, which is very natural. View-switch along the z-dimension, however, which corresponds to forward or backward movement of the observer's head into or out of the observed 3D scene—also natural for a sitting observer—is not handled in Daribo et al.[9]

We stress that *camera motion into / out of the 3D scene is not the same as camera zoom.* During a camera zoom, all objects in the observed scene increase in size at the same rate, and conventional SR algorithms would work well. In contrast, during a camera motion into / out of the scene (also called a *dolly shot* in cinematography), closer objects increase in size faster than objects further away. This motivates the depth-layer-based approach we propose in this paper.

3. SYSTEM OVERVIEW

We first overview our server-client multiview image navigation system. We assume texture and disparty maps at different camera coordinates (x, z), as shown in Fig. 1, were previously captured and are available at server for compression prior to the start of any streaming session. At the start of a streaming session, a user begins his multiview image navigation at a designated coordinate $c_i = (x_i, z_i)$. Subsequently, the user can interactively request a view-switch $c_i \rightarrow c_j$ to a neighboring camera coordinate c_j. Correspondingly, the streaming server will transmit necessary image data (to be discussed in Section 5), so that correct decoding and display of image at coordinate c_j is possible, given image at coordinate c_i is available at decoder buffer as reference.

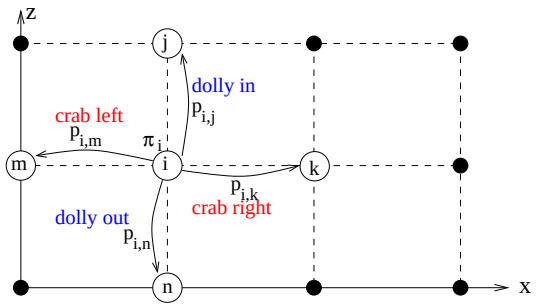


Figure 1. View-switches along x- and z-dimension correspond to different camera motions initiated by viewer.

3.1 View-switching along x- and z-dimensions

More specifically, if the requested view-switch $c_i \rightarrow c_j$ is to coordinate $c_j = (x_i \pm \theta, z_i)$, for some constant θ, then view-switch $c_i \rightarrow c_j$ corresponds to camera *crab* motion to the left or right. Alternatively, if the requested view-switch $c_i \rightarrow c_j$ is to coordinate $c_j = (x_i, z_i \pm \phi)$, for given constant ϕ, then view-switch $c_i \rightarrow c_j$ corresponds to camera *dolly* motion into and out of the 3D scene. Previous multiview image coding work[9] focuses on view synthesis and coding supporting camera crab motion along the x-dimension. In this paper, we instead focus on view synthesis and coding supporting camera dolly motion along the z-dimension.

3.2 Objective Function

We now describe the objective function for the view synthesis & coding problem in our multiview image navigation system. We first assume that the quality of the rendered images at all coordinates is controlled by a common and fixed quantization parameter (QP), which specifies the required image quality during navigation. The goal is then to minimize the expected transmission rate R during user's image navigation. Let π_i be the *steady state probability* of a user landing at image coordinate c_i. Let $p_{i,j}$ be the *view-switch probability* of a user switching from coordinate c_i to c_j, given the user is initially in coordinate c_i. We can now write R simply as:

$$\min R = \sum_i \pi_i \sum_{j|c_i \rightarrow c_j} p_{i,j} R_{i,j} \tag{1}$$

where $R_{i,j}$ is the size of the image data transmitted by server to client to enable view-switch $c_i \rightarrow c_j$; i.e., so client can correctly decode image at c_j given image at c_i is available in the decoder buffer. In the sequel, we will discuss view synthesis & coding algorithms so that $R_{i,j}$ is minimized for all view-switches $c_i \rightarrow c_j$.

4. DEPTH LAYERING

We first overview the traditional *pixel-based* synthesis & coding approach using depth-image-based rendering (DIBR), as done in previous work. We then describe its shortcoming during camera dolly motion, and propose our alternative *layer-based* synthesis & coding approach. Finally, we describe how we divide an image into depth layers for separate synthesis and coding.

4.1 Pixel-based DIBR Synthesis & Coding Approach

During a camera crab motion, pixels of objects in the 3D scene close to the observer have larger horizontal displacements (disparity) in the new image relative to objects further away. As an example, the closer green doll in Fig. 2 has larger horizontal displacement than the further blue box when the camera switches from left to right view. The amount of disparity required per-pixel during a camera-crab view-switch is typically described in a disparity map (or conversely, in a depth map, where there is a one-to-one mapping between disparity and depth). By encoding both texture and depth maps from the same viewpoint,[13] the decoder can first construct the new horizontally shifted view using a DIBR-based synthesis method like 3D warping.[5] Pixels in disoccluded regions in the new shifted view (pixels not visible in the original view) can either be inpainted exploiting correlation to neighboring known pixels,[6] or intra-coded and sent from server for reconstruction at decoder.

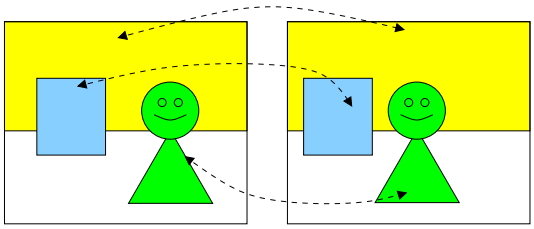


Figure 2. During a camera crab motion, objects shift horizontally by different amount depending on respective distances to observer.

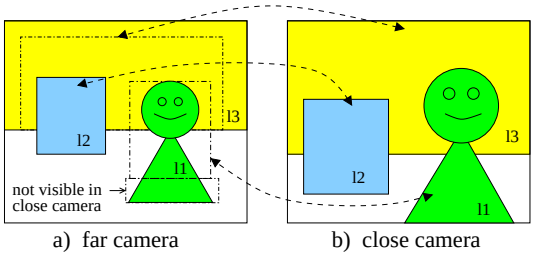


Figure 3. During a camera dolly motion, object sizes scale to different factors depending on respective distances to observer.

4.2 Layer-based DIBR Synthesis & Coding Approach

In contrast, during a camera dolly motion, objects closer to the observer have larger scaling factors than objects further away. In Fig. 3, the closer green doll increases in size more than the further blue box as the camera moves closer. If traditional pixel-based DIBR view synthesis as described previously is employed for camera-dolly view-switch as well, then adjacent pixels of the same object in the original view will be scattered far apart in the new view as the scale of the object increases, leaving newly created pixels in-between empty. Filling these pixels using general inpainting techniques like Oh et al.[6] would be difficult to achieve high quality.

The shortcoming of the pixel-based approach motivates us to derive an alternative layer-based approach: segment a 3D scene into *depth layers*, where each layer corresponds to a contiguous region of pixels with similar depth values. In Fig. 3, the green doll and blue box are segmented into separate depth layers. During a camera dolly-in motion, spatial region in the same layer will be enlarged via super-resolution (SR). (Pixels of same depth values will have the same scale factor.) Because SR is very likely performed on the same physical object, there will be textural consistency within the layer (object), leading to high quality reconstruction.

On the other hand, during a camera dolly-out motion, spatial region in the same layer will be shrunk via low-pass filtering and down-sampling. Server will transmit an intra-coded depth layer only if these operations fail to reconstruct a layer of high enough quality.

We note that pixels of the same depth will be shifted horizontally by the same amount during a camera crab motion, and hence our proposed layer-based approach for view synthesis and coding can also be used for view-switches in the x-dimension.

We next describe how an image can be segmented into depth layers efficiently.

4.3 Depth-based Segmentation Algorithm

We segment an image into K depth layers using available 8-bit per-pixel depth map as follows. First, we begin with an initial segmentation into K layers using disparity boundary values $D_1, D_2, \ldots, D_{K+1}$, where $D_1 = 0$ and $D_{K+1} = 255$. Pixels (i, j) with disparity value $d_{i,j}$ between D_l and D_{l+1} are initialized to layer l. Second, for each layer l, we compute a *centroid* disparity value v_l, which minimizes the mean square error between the disparity values $d_{i,j}$'s of the pixels in the layer and v_l, $MSE_l = \sum_{(i,j)\in l}(d_{i,j} - v_l)^2$. MSE for the whole depth map is then $MSE_{total} = \sum_{l=1}^{K} MSE_l$. Given centroids v_l's of all layers, a third step is to find new boundary values to repartition pixels in order to minimize MSE_{total}. We iterate between step two and three until we reach the local minimum of MSE_{total}.

The quality of the converged locally optimal solution clearly depends on the initial boundary values. We choose the initial boundary values in a heuristic way. We use Canny edge detector to find the edges of the depth map, and separately, construct a histogram of depth values of the image. The $K-1$ most prominent peaks in the histogram of the pixels along the detected edges will be used as the initial boundaries.

5. CODING & SYNTHESIS OF DEPTH LAYERS

5.1 Coding & Synthesis for Dolly-in View-switch

When a user requests a dolly-in view-switch $c_i \rightarrow c_j$, i.e., $c_j = (x_i, z_i + \phi)$, we perform the following coding & synthesis operations. Depending on depth layer l_i^k's distance δ to the observer at coordinate c_i, spatial region spanned by layer l_i^k will be increased by scale factor $s_\delta(z_i, z_i+\phi)$ to layer l_j^k at coordinate c_j. Using layer l_i^k, user can attempt this SR operation[†] at decoder by factor $s_\delta(z_i, z_i+\phi)$ to reconstruct l_j^k, resulting in distortion $d_{i,j,k}(0)$. Alternatively, the server can explicitly transmit layer l_j^k of image at coordinate c_j with distortion $d_{i,j,k}(1)$ and rate $r_{i,j,k}(1)$. Among these two options, the server will make a rate-distortion (RD) optimal decision $\rho_{i,j}(k) \in \{0, 1\}$ on whether to transmit or not, i.e.,

$$\rho_{i,j}(k) = \begin{cases} 1 & \text{if } d_{i,j,k}(0) > d_{i,j,k}(1) + \lambda r_{i,j,k}(1) \\ 0 & \text{o.w.} \end{cases} \tag{2}$$

where $\lambda > 0$ is a pre-defined Lagrangian multiplier weighting the cost of transmission rate against distortion.

Intuitively, the server will not send layer l_j^k during view-switch $c_i \rightarrow c_j$, if the spatial region can be easily super-resolved, e.g., the scale factor $s_\delta(z_i, z_i + \phi)$ is very close to one, or layer l_j^k has very little texture details and hence can be easily interpolated from pixels in l_i^k.

The transmission rate during view-switch $c_i \rightarrow c_j$ will be the sum of coding rates for all layers transmitted, *plus* the size W_j of a DSC frame. As done in Cai et al.,[11] DSC frame[‡] is used to merge the small coding differences among reconstructed frames from all view-switches to coordinate c_j, so that resulting rendered frame is bit-by-bit equivalent to the I-frame at coordinate c_j. We can now write $R_{i,j}$ as:

$$R_{i,j} = W_j + \sum_k \rho_{i,j}(k) r_{i,j,k}(1) \tag{3}$$

5.2 Coding & Synthesis for Dolly-Out View-switch

When a user requests a dolly-out view-switch $c_i \rightarrow c_j$, i.e., $c_j = (x_i, z_i - \phi)$, the required coding & synthesis operations are more complicated. The reason is that during a camera dolly-out motion, the field-of-view of a particular layer can become larger. As an example, in Fig. 3 we see that when moving from the close camera view in (b) to the far camera view in (a), the bottom part of the green doll that was not visible in close camera view becomes visible in far camera view. That means instead of performing SR to increase scale of a depth layer during view synthesis for dolly-in view-switch, during dolly-out view-switch a user needs to perform two operations:

1. reduce scale of spatial region in layer l_i^k by factor $s_\delta(z_i, z_i - \phi)$ to partially construct layer l_j^k; and
2. perform *inpainting* to fill in unknown pixels in spatial region visible in layer l_j^k but not in l_i^k.

The inpainting operation can be performed only if the boundary of the newly visible spatial region in layer l_j^k is known. We employ a method to extrapolate the boundaries of known spatial regions to unknown spatial regions for inpainting. If the extrapolation quality is poor, then the server has no other recourse but to explicitly transmit the newly visible spatial region in layer l_j^k at rate $r_{i,j,k}(1)$. The decision of whether to transmit newly visible spatial region in layer l_j^k is also made in a RD-optimal manner, as done previously. This also leads to a transmission rate $R_{i,j}$ as expressed in (3).

[†]Note that not all pixels in the super-resolved spatial regions of the layer may be visible in the image's field-of-view at new coordinate, as illustrated by the green doll in Fig. 3 when moving to a closer camera.

[‡]If the constructed frames from switches c_i's to c_j are of poor quality, the subsequent required DSC frame will be large. Hence the described RD-optimal selection of layers for transmission is important in minimizing $R_{i,j}$.

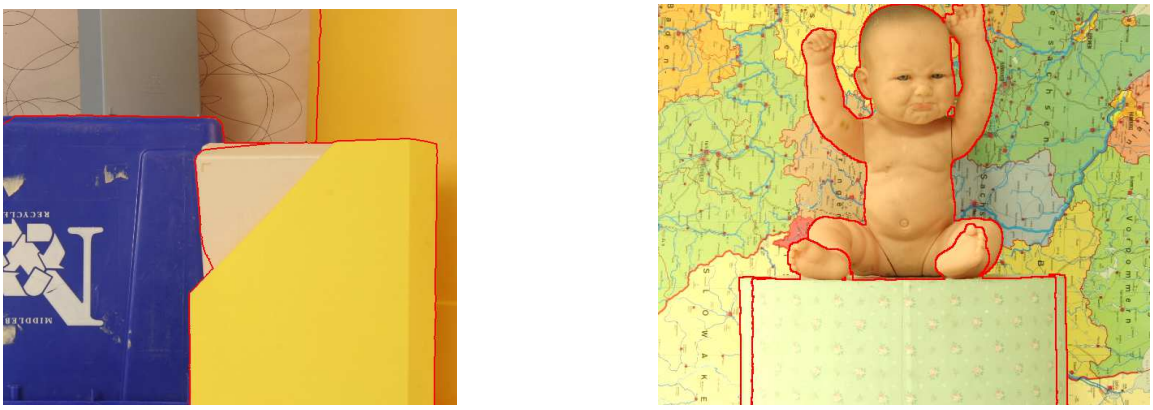
5.3 Coding & Synthesis for x-dimension View Switching

For a x-dimension view switch $c_i \rightarrow c_j$, i.e., $c_j = (x_i \pm \theta, z_i)$, there is no need to change the visual size of the depth layers. We shift the depth layers as a whole based on the observation that the pixels in the same depth layer have similar distance δ to the observer, which means their horizontal displacement during the view switch will also be similar. The depth layers with small variation in distance to the observer δ will result in high quality; otherwise, the synthesis quality stemming from uniform horizontal shift of the entire depth layer will not be good. Missing pixels in disocclusion holes and the horizontal margin that appear in the new field-of-view in the synthesized image will be filled with the same inpainting method as described earlier. Specifically, the following coding & synthesis operations are performed.

1. The user will attempt to shift each depth layer horizontally with a certain displacement, which is decided by the depth layer l_i^k's distance δ to the observer at coordinate c_i; and
2. perform *inpainting* to fill in unknown pixels in spatial region visible in layer l_j^k but not in l_i^k.

Once again, the server will decide whether to transmit new spatial region in the RD-optimal manner, and the transmission rate $R_{i,j}$ is expressed in (3).

6. RESULTS



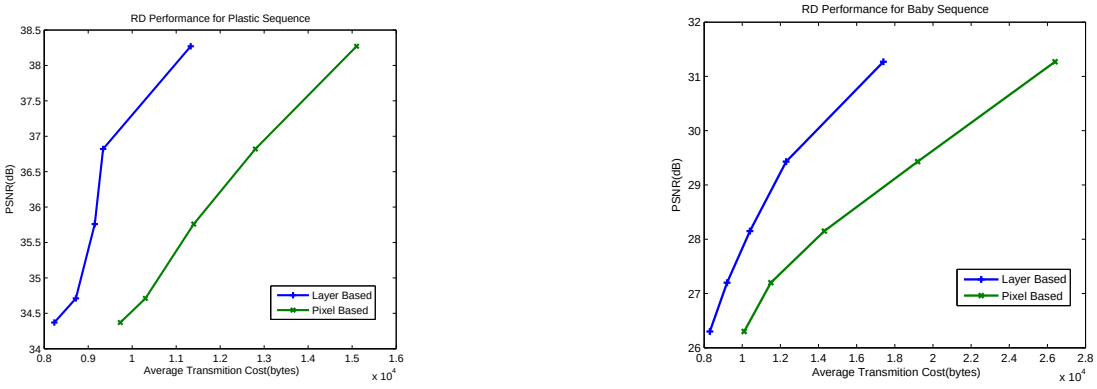
(a) Segmented image for `plastic` sequence

(b) Segmented image for `baby` sequence

Figure 4. Example of segmented image using per-pixel depth map

We used two test sequences of multiview images in our experiment: `plastic` and `baby` in Middlebury's 2006 datasets.[14] We resized the images of `plastic` and `baby` sequences to 624×544 and 608×496 respectively, so that pixel rows and columns were multiples of 16. For both sequences, we used seven and three multiview images along the x- and z-dimension respectively for the image navigation grid, resulting in 21 images for interactive browsing. We used the following view transition possibilities: a user can choose to dolly-in or out with probability 0.7 (in and out with equal probability), and a probability of 0.3 to crab left or right (left and right with equal probability). We used our proposed depth-based segmentation algorithm to divide each image into 3 layers. The result of a segmented image for both sequences is shown in Fig. 4. As shown, we were successful in dividing the image into roughly the same physical objects in the 3D scene. After the segmentation, we encoded individual depth layers using a H.263 encoder,[7] with QP varying from 10 to 30 with step-size of 5.

For comparison, we employed a pixel-based approach as follows. It first divides each image into 16×16 blocks. For each view-switch $c_i \rightarrow c_j$, it checks the quality of each block in the DIBR-synthesized image of coordinate c_j using image of coordinate c_i. If the quality is good, pixels of the original block in coordinate c_j would not be transmitted. Otherwise, the corresponding block in the original picture will be sent inside an additional frame. The additional frame thus contains all individual blocks that need to be sent, while the remaining part of the



(a) `plastic` sequence (b) `baby` sequence

Figure 5. RD performance of competing coding schemes

frame is filled with zeroes. Then we intra-code the additional frame and send it. As in our layer-based approach, a DSC frame is also transmitted thereafter to remove small coding differences of frames constructed from all view-switches to c_j.

We plotted the RD performance of both approaches in Fig. 5 for both sequences, where image quality is measured in PSNR. The PSNR value on the y-axis is determined by the QP, and the transmission rate is calculated using (1). We see that at the same quality, our proposed layer-based approach can reduce bit-rate by up to 25% and 35% for the `plastic` and `baby` sequences respectively compared to pixel-based approach.

7. CONCLUSION

Multiview image navigation system provides interactive view-switching during image browsing according to viewer's requests. To reduce transmission rate during a view-switch along the z-dimension (camera dolly motion), in this paper we propose a layer-based synthesis & coding approach, where pixels of similar depth values are grouped together as a depth layer. During a dolly view-switch, spatial region in a depth layer is either rescaled larger via super-resolution for camera dolly-in motion, or rescaled smaller via low-pass filtering and down-sampling for camera dolly-out motion, for reconstruction of the layer in the new view. A depth layer is intra-coded and transmitted by server only if the reconstructed depth layer via rescaling is not of high enough quality, saving transmission bits. Experiments show that our layer-based approach can reduce bit-rate by up to 30% compared to previous approach.

REFERENCES

[1] C. Zhang, Z. Yin, and D. Florencio, "Improving depth perception with motion parallax and its application in teleconferencing," in *IEEE International Workshop on Multimedia Signal Processing*, Rio de Jeneiro, Brazil, October 2009.

[2] S. Reichelt, R. Haussler, G. Futterer, and N. Leister, "Depth cues in human visual perception and their realization in 3D displays," in *SPIE Three-Dimensional Imaging, Visualization, and Display 2010*, Orlando, FL, April 2010.

[3] T. Fujii, K. Mori, K. Takeda, K. Mase, M. Tanimoto, and Y. Suenaga, "Multipoint measuring system for video and sound—100 camera and microphone system," in *IEEE International Conference on Multimedia and Expo*, Toronto, Canada, July 2006.

[4] S. Gokturk, H. Yalcin, and C. Bamji, "A time-of-flight depth sensor—system description, issues and solutions," in *Conference on Computer Vision and Pattern Recognition Workshop (CVPRW)*, Washington, DC, June 2004.

[5] W. Mark, L. McMillan, and G. Bishop, "Post-rendering 3D warping," in *Symposium on Interactive 3D Graphics*, New York, NY, April 1997.

[6] K.-J. Oh, S. Yea, and Y.-S. Ho, "Hole-filling method using depth based in-painting fore view synthesis in free viewpoint television (FTV) and 3D video," in *Picture Coding Symposium*, Chicago, IL, May 2009.

[7] ITU-T Recommendation H.263, *Video Coding for Low Bitrate Communication*, February 1998.

[8] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," in *IEEE Transactions on Circuits and Systems for Video Technology*, July 2003, vol. 13, no.7, pp. 560–576.

[9] I. Daribo, G. Cheung, , and T. Maugey P. Frossard, "RD optimized auxiliary information for inpainting-based view synthesis," in *3DTV-Conference 2012*, Zurich, Switzerland, October 2012.
[10] X. Li and M. Ochard, "New edge-directed image interpolation," in *IEEE Transactions on Image Processing*, October 2001, vol. 10, no.10.
[11] W. Cai, G. Cheung, T. Kwon, and S.-J. Lee, "Optimized frame structure for interactive light field streaming with cooperative cache," in *IEEE International Conference on Multimedia and Expo*, Barcelona, Spain, July 2011.
[12] N.-M. Cheung, A. Ortega, and G. Cheung, "Distributed source coding techniques for interactive multiview video streaming," in *27th Picture Coding Symposium*, Chicago, IL, May 2009.
[13] P. Merkle, A. Smolic, K. Mueller, and T. Wiegand, "Multi-view video plus depth representation and coding," in *IEEE International Conference on Image Processing*, San Antonio, TX, October 2007.
[14] "2006 stereo datasets," http://vision.middlebury.edu/stereo/data/scenes2006/.