EXPANSION HOLE FILLING IN DEPTH-IMAGE-BASED RENDERING USING GRAPH-BASED INTERPOLATION

Yu Mao [*], *Gene Cheung* [#], *Antonio Ortega* [$] *and Yusheng Ji* [#]

[*] The Graduate University for Advanced Studies, [#] National Institute of Informatics,
[$] University of Southern California

ABSTRACT

Using texture and depth maps of a single reference viewpoint, depth-image-based rendering (DIBR) can synthesize a novel viewpoint image by translating texture pixels of the reference view to a virtual view, where synthesized pixel locations are derived from the associated depth pixel values. When the virtual viewpoint is located much closer to the 3D scene than the reference view (camera movement in the z-dimension), objects closer to the camera will increase in size in the virtual view faster than objects further away. A large increase in object size means that a patch of pixels sampled from an object surface in the reference view will be scattered to a larger spatial area, resulting in expansion holes. In this paper, we investigate the problem of identification and filling of expansion holes. We first propose a method based on depth histogram to identify missing or erroneously translated pixels as expansion holes. We then propose two techniques to fill in expansion holes with different computation complexity: i) linear interpolation, and ii) graph-based interpolation with a sparsity prior. Experimental results show that proper identification and filling of expansion holes can dramatically outperform inpainting procedure employed in VSRS 3.5 (up to 4.25dB).

Index Terms— depth-image-based rendering, image interpolation, graph-based transform

1. INTRODUCTION

With the advent of depth sensors such as time-of-flight (ToF) cameras [1] and Microsoft Kinect, availability of depth maps (per-pixel distance between captured objects in the 3D scene and capturing camera) has become commonplace. Armed with texture (conventional RGB or YUV images) and depth maps from the same camera viewpoint—a format known as *texture-plus-depth* [2], a user can synthesize a new virtual viewpoint image using *depth-image-based rendering* (DIBR) techniques [3] such as 3D warping [4]. In a nutshell, DIBR is a pixel-to-pixel mapping from reference to virtual view: each texture pixel in the reference view is mapped to a synthesized pixel in the virtual view, where the synthesized location is derived from the corresponding depth pixel in the reference view. Due to disocclusion (pixel locations that were not visible in the reference view), missing pixels in the virtual view are subsequently filled in using depth-based inpainting algorithms [5, 6]. For relatively small camera motion along the x- or y-dimension (camera moving left-right or top-down), this DIBR synthesis plus inpainting approach has been shown to work reasonably well [7], and is the conventional approach in the 3D view synthesis literature.

In immersive applications such as teleconferencing [8], a viewer in a sitting position observes a real-time synthesized image on a 2D display, whose rendering perspective is adaptively changed in response to the up-to-date tracked head position of the viewer. The resulting *motion parallax* effect can enhance the viewer's depth perception in the 3D scene [9]. Besides x-dimensional head movement (moving one's head left-right), z-dimensional head movement (moving one's head front-back) is also natural for a sitting observer to make, inducing perspective change. Yet, to the best of the authors' knowledge, no DIBR-based view synthesis work in the literature has formally addressed the problem of synthesizing virtual view corresponding to large camera motion in the z-dimension. We address this problem in our paper.

When the virtual viewpoint is located much closer to the 3D scene than the reference view, objects closer to the camera will increase in size in the virtual view faster than objects further away. A large increase in object size means that a patch of pixels sampled from an object surface in the reference view will be scattered to a larger spatial area, resulting in *expansion holes*. To further complicate matters, in-between these dispersed pixels there can be scattering of synthesized pixels from a further-away object (larger z distance), that should have been occluded by the closer object if a sufficient number of closer object pixels were rendered.

In this paper, we investigate the problem of identification and filling of expansion holes in the virtual view. We first propose a method based on depth histograms to identify missing or erroneously synthesized pixels as expansion holes. We then propose two techniques to fill in expansion holes with different computation complexity: i) linear interpolation, and ii) graph-based interpolation with a sparsity prior. Experimental results show that proper identification and filling of expansion holes can dramatically outperform inpainting procedure employed in VSRS 3.5 (up to 4.25dB).

The structure of the paper is as follows. We first discuss related work in Section 2. We then overview our interactive DIBR view synthesis system in Section 3. We present our methodology to identify and to fill in expansion holes in the virtual view in Section 4 and 5, respectively. Finally, experimental results and conclusions are presented in Section 6 and 7, respectively.

2. RELATED WORK

It is known that texture-plus-depth format [2]—representation of the 3D scene in one or more texture and depth map pairs from different viewpoints—can enable low-complexity rendering of freely chosen viewpoint images at decoder via DIBR [4]. While the traditional approach [3] advocates transmission of two (or more) pairs of texture and depth maps from neighboring viewpoints for synthesis of an intermediate virtual view, recent investigations [10, 11] show that transmission of a single texture-depth map pair can be more rate-distortion (RD) optimal, *if* the resulting larger disocclusion holes can be smartly handled. We will also assume availability of a single texture-depth map pair from the same reference viewpoint for

DIBR-based view synthesis in our work. Nonetheless, we note that synthesizing an intermediate virtual view using two texture-depth map pairs will not eliminate the expansion hole problem if camera movement in the z-dimension is significant, though the number and sizes of expansion holes will in general be smaller.

Increase in size of a closer object in the virtual view due to significant z-dimensional camera motion can be solved using conventional image super-resolution (SR) [12] in rectangular pixel grid. For example, texture and depth maps in the reference view can be super-resolved into a finer rectangular grid of sufficiently high resolution (one where all possible expansion holes in the virtual view will be covered), then performing DIBR to see which of the super-resolved pixels actually land on the virtual view pixel grid. Unlike this SR approach which requires computation of a possibly very large number of super-resolved pixels in the reference view (and only a smaller subset get mapped to the grid points in the virtual view), our approach is a *parsimonious* one: only grid samples identified as expansion hole pixels in the virtual view—empty pixels that require filling—are interpolated, leading to a lower complexity relative to the aforementioned SR approach.

Besides linear interpolation (with low complexity), we also advocate an interpolation method based on *graph-based transform* (GBT), which uses the eigenvectors of a defined graph Laplacian matrix to provide a Fourier-like frequency interpretation [13], for expansion hole filling. Unlike previous fixed transform based interpolation like DCT [14] defined on rectangular pixel grid, GBT is adaptive to a more general setting where any n unknown pixels can be interpolated using any m known pixels, all connected via a weighted graph. Compared to non-local image interpolation methods [15], the complexity of our GBT interpolation is bounded by the few number of pixels within the neighborhood of an expansion hole used to construct the graph. While GBT has been used for compression of depth maps [16, 17], this is the first work in the literature of using GBT for image patch interpolation.

3. INTERACTIVE FREE-VIEWPOINT SYSTEM

We first describe the system model for our interactive free viewpoint streaming system. The goal is for the server to transmit a minimum number of bits to the client, so that the client can interactively select a viewpoint of his/her choosing for DIBR-based image rendering of the 3D scene. Like [10, 11], we assume the server transmits texture and depth maps of only one camera-captured viewpoint (called *reference view* in the sequel) to the client to cover a defined neighborhood of virtual views. If the client moves outside the current neighborhood to a new one, a new pair of texture and depth maps (differentially coded from the previously transmitted pair) will be transmitted to cover the new neighborhood of virtual views. In this paper, we focus only on synthesis of virtual view images in the neighborhood with significant z-dimensional camera movements.

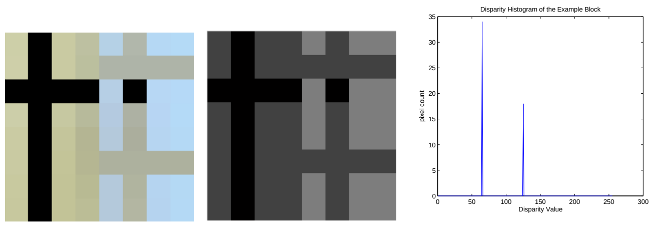
3.1. Hole Filling in DIBR Synthesized Image

In 3D warping, we often observe *holes* in the virtual view; i.e., a pixel in the virtual view that has no corresponding pixel in the reference view. There are two main kinds of holes. The first kind is *disocclusion holes*: the corresponding pixel in the reference view is occluded by a pixel of another object closer to the camera. Disocclusion holes can be filled using depth-based image inpainting techniques [5, 6], and are outside the scope of this paper. The second kind is *expansion holes*. We define an expansion hole as follows: a spatial area of an object's surface in the virtual view, whose corresponding area in the reference view is visible but smaller in size. Unlike disocclusion holes, expansion holes can leverage on information of neighboring pixels with similar depth (indicating they are of the same object) for interpolation.

We first identify pixels in the virtual view as expansion holes using a method based on depth histogram. We then propose two methods for pixel interpolation: i) linear interpolation, and ii) graph-based interpolation with a sparsity prior. We discuss these next.

4. EXPANSION HOLE IDENTIFICATION

We perform the following procedure to identify expansion holes in the DIBR-synthesized virtual view image. Denote (x, y) the coordinate of a pixel in the reference view and $t(x, y)$ and $d(x, y)$ the texture and depth values at that coordinate, respectively. When rendering from reference view to virtual view, a rendering function $\mathcal{F}(x, y) = (x', y')$ maps a pixel (x, y) in reference view to (x', y') in virtual view. The inverse mapping function $\mathcal{F}'(x', y') = (x, y)$ maps from (x', y') in virtual view to (x, y) in reference view. Both $\mathcal{F}$ and $\mathcal{F}'$ can be easily derived from standard 3D warping equations [3]. We denote the distance between two pixels i and j: $H((x_i, y_i), (x_j, y_j)) = |x_i - x_j| + |y_i - y_j|$.



(a) texture block (b) disparity block (c) depth histogram

Fig. 1. Example of texture and disparity block, and constructed depth histogram.

We first divide the virtual view into non-overlapping $b \times b$ blocks. For a given block, we next decompose it into depth layers as follows: i) construct a histogram of depth values of the synthesized pixels in the block, ii) separate depth pixels into layers by identifying local minima in the histogram and using them as layer-dividing boundaries. Fig. 1 shows an example texture and disparity[1] block, and corresponding depth histogram. We next process each layer in order of increasing depth values (closest layer to the camera first).

When processing a layer l, all synthesized pixels of high layers $l + 1, \ldots$ are treated as empty pixels; this affords us an opportunity to erase a synthesized background pixel from an expansion hole of the foreground object. We examine each empty pixel in the block as follows. As shown in Fig. 2, we divide the neighborhood of an empty pixel (marked X in Fig. 2) into four quadrants. In each quadrant, we find the synthesized pixel i at (x'_i, y'_i) that is closest to the empty pixel in distance $H()$. It is possible that there are no synthesized pixels in a particular quadrant. After we acquired a maximum set of four nearest pixels in the four quadrants, we check if each pair of nearest pixels in neighboring quadrants, i and j, are nearby pixels in the reference view. Specifically, using inverse mapping function

[1]There is a one-to-one correspondence between depth and disparity, where disparity is inversely proportional to depth. Thus, disparity map can be equivalently processed instead of depth map.

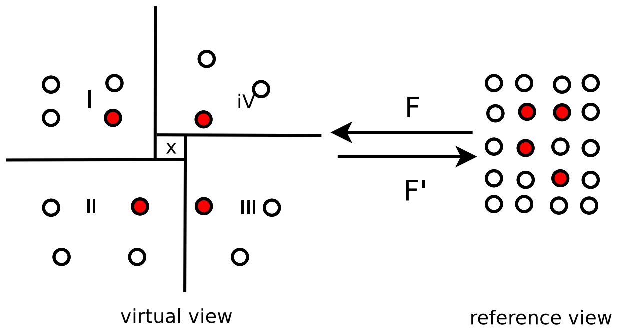


Fig. 2. Expansion Holes on a Depth Layer

$\mathcal{F}'$, we check if $H(\mathcal{F}'(x_i', y_i'), \mathcal{F}'(x_j', y_j')) < n$, where n is a preset distance threshold. If we find two or more neighboring pairs among an empty pixel's nearest pixel set, we declare that this empty pixel belongs to an expansion hole. The parameter n determines the sensitivity of this method. We set $n = 2$ in our experiments to balance the false positives and false negatives.

The intuition behind this method is that if the set of closest pixels in the virtual view are nearby pixels in the reference view, then the empty pixel in the virtual view is very likely to be inside the convex set spanned by the closest pixels in the reference view. After identifying expansion hole empty pixels, we interpolate them using one of two methods described next.

5. EXPANSION HOLE PIXEL INTERPOLATION

Having identified expansion hole empty pixels, we now discuss how we interpolate them using two methods: linear interpolation and graph-based interpolation with sparsity prior.

5.1. Linear Interpolation

For linear interpolation, for each identified empty pixel in an expansion hole, we search for the three nearest synthesized pixels i, j, k and construct a linear plane that connects their texture values, $t(x_i, y_i)$, $t(x_j, y_j)$, $t(x_k, y_k)$ given their coordinates (x_i, y_i), (x_j, y_j), (x_k, y_k). The empty pixel is interpolated using the constructed plane and its own pixel coordinate. The advantage of this method is that it is simple and has low computation complexity.

5.2. Graph-based Interpolation with Sparsity Constraint

We next describe a more complex method using GBT for interpolation. We first discuss how GBT basis functions are derived. We then discuss how the optimization is formulated and performed given the derived GBT basis functions.

5.2.1. Constructing a Graph-based Transform

We first overview the procedure to construct a GBT [16], which is a signal-adaptive block transform. We begin by defining a graph $\mathcal{G}$ connecting pixels in the block (nodes in the graph), where an edge connecting pixel i and j has edge weight $e_{i,j}$. Next, we define the degree matrix $\mathbf{D}$ and adjacency matrix $\mathbf{A}$ from the constructed graph $\mathcal{G}$. Adjacency matrix $\mathbf{A}$ has entry $A_{i,j}$ containing edge weight $e_{i,j}$ if edge connecting i and j exists, and 0 otherwise. Degree matrix $\mathbf{D}$ is a diagonal matrix with non-zero entries $D_{i,i} = \sum_j e_{i,j}$. A *graph Laplacian* $\mathbf{L} = \mathbf{D} - \mathbf{A}$ can then be defined. Finally, we perform eigen-decomposition on $\mathbf{L}$, i.e., find eigen-vectors ϕ_i's such that $\mathbf{L}\phi_i = \rho_i \phi_i$, where ρ_i is the i-th graph frequency. The basis vectors of the GBT are ϕ_i's. If we now project a signal $\mathbf{s}$ in the graph $\mathcal{G}$ onto the eigen-vectors ϕ_i's of the Laplacian $\mathbf{L}$, it becomes the spectral decomposition of the signal; i.e., it provides a "frequency domain" interpretation of signal $\mathbf{s}$ given graph support $\mathcal{G}$.

The performance of GBT-based interpolation depends to a large extent on how the graph $\mathcal{G}$ is constructed. We propose to construct $\mathcal{G}$ for empty pixels and synthesized pixels in a depth layer l as follows. First, we draw an edge from each pixel to its k nearest pixels in terms of coordinate distance $H()$. Next, we assign an edge weight $e_{i,j}$ between pixels i and j, if i and j are connected, as follows:

1. If i and j are both synthesized pixels, then $e_{i,j}$ is assigned a weight inverse proportional to their texture value difference.
2. If one of i and j is an empty pixel, then $e_{i,j}$ is assigned a weight inverse proportional to their coordinate distance $H()$.

5.2.2. Linear Program Formulation

Without loss of generality, let the N synthesized pixels in a patch be $s_1, \ldots, s_N$, and the interpolated length-M signal, $M > N$, be $\hat{\mathbf{s}} = [\hat{s}_1, \ldots, \hat{s}_M] = \Phi\mathbf{w}$, where columns of $M \times M$ matrix Φ, ϕ_j's, are the M GBT basic vectors as derived earlier, and $\mathbf{w}$ is the coefficient vector for signal interpolation. Let $\mathbf{u}_i$'s be a set of N length-M unit vectors, $[0, \ldots, 0, 1, 0, \ldots, 0]$, where the single non-zero entry is at position i. Our objective is to minimize a weighted sum of: i) the l_1-norm of the difference between interpolated signal $\hat{\mathbf{s}}$ and original signal $\mathbf{s}$ at the N synthesized pixel locations, and ii) weighted l_0-norm of the coefficient vector $\mathbf{w}$ (low frequencies $\mathbf{w}_l$ has weight λ_l and high frequencies $\mathbf{w}_h$ has weight λ_h):

$$\min_{\mathbf{w}} \|\sum_{i=1}^{N} \mathbf{u}_i^T \Phi \mathbf{w} - s_i\|_1 + \lambda_l \|\mathbf{w}_l\|_0 + \lambda_h \|\mathbf{w}_h\|_0 \tag{1}$$

As typically done in the literature for sparse signal recovery, we can swap the l_0-norm above with a l_1-norm:

$$\min_{\mathbf{w}} \|\sum_{i=1}^{N} \mathbf{u}_i^T \Phi \mathbf{w} - s_i\|_1 + \lambda_l \|\mathbf{w}_l\|_1 + \lambda_h \|\mathbf{w}_h\|_1 \tag{2}$$

(2) can now be easily rewritten as a linear programming (LP) formulation [18], and can thus be solved using any one of a set of known LP algorithms [19].

6. EXPERIMENTATION

6.1. Experimental Setup

We used `art` and `laundry` in Middlebury's 2005 datasets[2] as our multiview image test sequences. For `Art`, we used the fifth view of the sequence as the ground truth for virtual view v_0, which was resized to 1104×1384 so that the pixel rows and columns were multiples of 8. Then, we used DIBR to generate the reference view v_r for our experiment, where v_r is further away from the camera than v_0. Since pixels are moving towards each other during this view-switch, there would be no expansion holes. We used a standard inpainting algorithm to fill the disocclusion holes to complete v_r. Similar procedure was performed for `laundry`.

Using texture and depth maps of v_r, we used DIBR again to generate virtual view v_0. We used one of the following three methods

[2]http://vision.middlebury.edu/stereo/data/scenes2006/

Table 1. PSNR Comparison of VSRS Inpainting, Linear Interpolation and GBT Interpolation for the identified expansion hole areas.

method	VSRS+	Linear	GBT
art PSNR(dB)	19.11	22.87	23.36
laundry PSNR(dB)	19.17	21.94	22.53

for filling of expansion holes. In the first method we call VSRS+, we modified VSRS software version 3.5 as follows. Because we are generating the virtual view from a single pair of texture and depth maps from the reference view, we skipped the step of blending two virtual view images synthesized from two different reference views after DIBR, and proceeded directly to the inpainting part of VSRS, which called an OPENCV inpainting algorithm. We note that the VSRS software is not designed for synthesis of virtual view images with significant z-dimensional camera movements. Nonetheless, we used VSRS as one benchmark comparison because: i) it is a well accepted and commonly used view synthesis software, and ii) to the best of the authors' knowledge, there are no other well accepted DIBR-based view synthesis strategies for virtual view with significant z-dimensional camera movement.

linear and GBT are the linear and graph-based interpolation methods as described in Section 5.

6.2. Experimental Results

After generating the virtual view images using the three methods, we calculated the PSNR of the virtual view images interpolated using the aforementioned three interpolation methods against the ground truth v_0. Since we proposed identification and interpolation of the expansion hole area, we will only calculated the PSNR of the identified expansion hole areas. The PSNR comparison is shown in Table 1. For the art sequence, we see that both Linear and GBT outperformed VSRS+ significantly: by 3.76dB and 4.25dB, respectively. This demonstrates that the correct identification of expansion holes and subsequent interpolation are important for DIBR image synthesis of virtual view with significant z-dimensional camera movement. Further, we see that GBT outperformed linear by 0.49dB, showing that using graph-based interpolation, we can achieve better image quality than simple linear interpolation.

For the laundry sequence, we observe similar trend. In this case, we see that Linear and GBT outperformed VSRS+ by 2.77dB and 3.36dB, respectively.

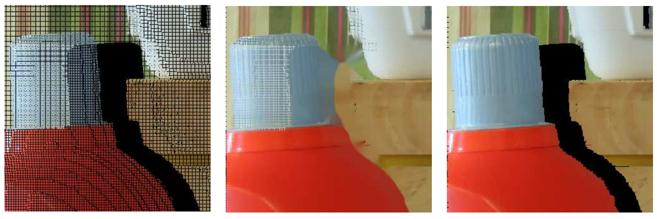

(a) expansion holes (b) VSRS+ (c) GBT

Fig. 3. Expansion holes and visual comparison between VSRS+ and GBT for sequence laundry.

Next, we examine the generated image quality visually. In Fig. 3, we show an example patch of the DIBR-synthesized image before filling of expansion holes, and after filling of expansion holes using VSRS+ and GBT, respectively, for the laundry sequence. The resolution of the patch is 320×320. First, we see visually in Fig. 3(a) that the presence of expansion holes is everywhere and is a significant problem. Note also that the nature of expansion holes is very different from disocclusion holes (e.g., right of the detergent bottle), which are larger contiguous regions. Second, we see in Fig. 3(b) that applying inpainting algorithm naively to fill in all missing pixels indiscriminately do not lead to acceptable quality for expansion hole areas. Finally, we see in Fig. 3(b) that using GBT, expansion holes can be filled in a visually pleasing manner.

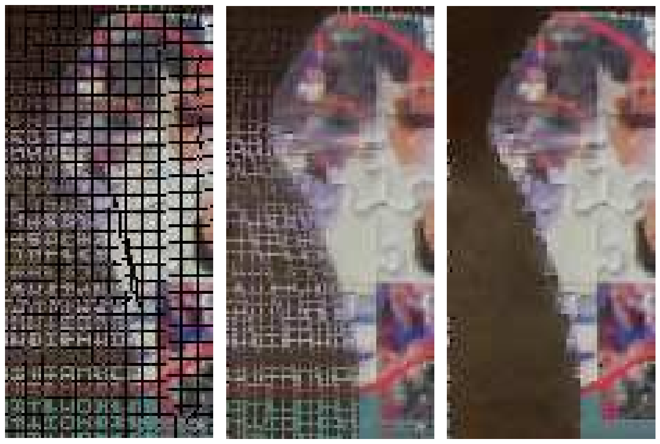

(a) expansion holes (b) VSRS+ (c) GBT

Fig. 4. Expansion holes and visual comparison between VSRS+ and GBT for sequence art.

Similarly, we show example image patch with expansion holes before and after filling in Fig. 4 for the art sequence. The resolution of this patch is 75×150. We again see that GBT can significantly improve visual quality of the DIBR-synthesized image. What is interesting in this image set is that using GBT, we were able to erase erroneously synthesized background pixels in the foreground expansion hole areas before performing interpolation, leading to better performance.

7. CONCLUSION

When the viewer's chosen virtual viewpoint for image rendering via DIBR involves significant camera motion in the z-dimension relative to the reference viewpoint (camera moving closer to the 3D scene), objects closer to the camera will increase in size faster than objects further away. Because insufficient number of pixel samples are available in the reference image, expansion holes will appear in the DIBR-rendered image in the virtual view. Unlike disocclusion holes that are filled using inpainting methods (extrapolation), expansion holes can be filled by interpolating from neighboring synthesized pixels of the same object surface. In this paper, we propose a procedure to correctly identify them in the virtual viewpoint image, then fill them using one of two methods: i) linear interpolation, and ii) graph-based interpolation with a sparsity prior. Experimental results show that up to 4.25dB gain can observed over inpainting method employed in VSRS 3.5.

8. REFERENCES

[1] 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.

[2] 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.

[3] D. Tian, P.-L. Lai, P. Lopez, and C. Gomila, "View synthesis techniques for 3D video," in *Applications of Digital Image Processing XXXII, Proceedings of the SPIE*, 2009, vol. 7443 (2009), pp. 74430T–74430T–11.

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

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

[6] I. Daribo and B. Pesquet-Popescu, "Depth-aided image in-painting for novel view synthesis," in *IEEE International Workshop on Multimedia and Signal Processing*, Saint-Malo, France, October 2010.

[7] M. Tanimoto, M. P. Tehrani, T. Fujii, and T. Yendo, "Free-viewpoint TV," in *IEEE Signal Processing Magazine*, January 2011, vol. 28, no.1.

[8] 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.

[9] S. Reichelt, R. Hausselr, 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.

[10] T. Maugey, P. Frossard, and G. Cheung, "Temporal and view constancy in an interactive multiview streaming system," in *IEEE International Conference on Image Processing*, Orlando, FL, September 2012.

[11] I. Daribo, G. Cheung, T. Maugey, and P. Frossard, "RD optimized auxiliary information for inpainting-based view synthesis," in *3DTV-Conference 2012*, Zurich, Switzerland, October 2012.

[12] X. Li and M. Ochard, "New edge-directed image interpolation," in *IEEE Transactions Image Processing*, October 2001, vol. 10, no.10, pp. 1521–1527.

[13] F. K. Chung, *Spectral Graph Theory (CBMS Regional Conference Series in Mathematics, No. 92)*, American Mathematical Society, 1996.

[14] Z. Alkachouh and M. Bellanger, "Fast DCT-based spatial domain interpolation of blocks in images," in *IEEE Transactions on Image Processing*, April 2000, vol. 9, no.4.

[15] G. Guy and S. Osher, "Nonlocal operators with applications to image processing," *Multiscale Modeling & Simulation*, pp. 1005–1028, 2008.

[16] G. Shen, W.-S. Kim, S.K. Narang, A. Ortega, J. Lee, and H. Wey, "Edge-adaptive transforms for efficient depth map coding," in *IEEE Picture Coding Symposium*, Nagoya, Japan, December 2010.

[17] W. Hu, G. Cheung, X. Li, and O. Au, "Depth map compression using multi-resolution graph-based transform for depth-image-based rendering," in *IEEE International Conference on Image Processing*, Orlando, FL, September 2012.

[18] G. Cheung, A. Kubota, and A. Ortega, "Sparse representation of depth maps for efficient transform coding," in *IEEE Picture Coding Symposium*, Nagoya, Japan, December 2010.

[19] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge, 2004.