On Constructing z-dimensional DIBR-Synthesized Images

Yu Mao *Student Member, IEEE*, Gene Cheung *Senior Member, IEEE*, Yusheng Ji *Member, IEEE*

***Abstract*—The "color-plus-depth" format represents a 3D scene using multiple color and depth images captured by an array of closely spaced cameras. Using this format, a novel image as observed from a horizontally shifted virtual viewpoint can be synthesized via depth-image-based rendering (DIBR), using neighboring camera-captured viewpoint images as reference. In this paper, using the same popularized color-plus-depth representation, we propose to construct in addition novel images as observed from virtual viewpoints closer to the 3D scene, enabling a new dimension of view navigation. To construct this new image type, we first perform a new DIBR pixel-mapping for z-dimensional camera movement. We then identify expansion holes—a new kind of missing pixels unique in z-dimensional DIBR-mapped images—using a depth layering procedure. To fill expansion holes we formulate a patch-based maximum a posteriori (MAP) problem, where the patches are appropriately spaced using diamond tiling. Leveraging on recent advances in graph signal processing (GSP), we define a graph-signal smoothness prior to regularize the inverse problem. Finally, we design a fast iterative reweighted least square (IRLS) algorithm to solve the posed problem efficiently. Experimental results show that our z-dimensional synthesized images outperform images rendered by a naïve modification of VSRS 3.5 by up to 4.01dB.**

***Index Terms*—Depth-image-based rendering (DIBR), color-plus-depth representation, graph signal processing**

I. Introduction

The promise of free viewpoint video [1] is to provide users the freedom to choose any vantage point from which to construct a viewpoint image for observation of a 3D scene. To enable free viewpoint, a conventional multiview imaging system contains an array of horizontally spaced cameras to capture color maps (conventional RGB images) and depth maps (per-pixel distance between objects in the 3D scene and the capturing camera) from different viewpoints—a format called *color-plus-depth* [2]. The captured images are subsequently encoded at the sender using standardized multiview coding tools such as 3D-HEVC [3].

At the receiver, a novel image from a virtual viewpoint—a horizontally shifted camera angle from the captured views—can be synthesized using *depth-image-based rendering* (DIBR) techniques such as 3D warping [4]. In a nutshell, DIBR maps each color pixel in a reference view to a 2D grid location in the virtual view, using disparity information provided by the corresponding depth pixel. Due to *occlusion* (visible spatial areas in the virtual view that are occluded by foreground objects in the reference view), there are missing pixels in the virtual view called *disocclusion holes*. They are subsequently completed using inpainting algorithms designed specifically for disocclusion hole filling [5–7]. For small camera movement from reference to virtual view along the x-dimension (camera moving left or right), this DIBR view synthesis plus inpainting approach works reasonably well [1], and is the conventional approach in the free view synthesis literature.

In immersive applications such as teleconferencing, a sitting viewer observes rendered images on a 2D display, where the image viewpoints are interactively adjusted according to the tracked locations of the viewer's head as he shifts left or right [8]. The resulting *motion parallax* effect can greatly enhance the viewer's depth perception in the 3D scene [9]. To enable this interactive view navigation in streaming systems over networks, previous works have optimized strategies for coding of color and depth maps [10, 11], error resilience [12], packet scheduling [13], caching [14] and reference view selection [15] for visual quality and/or view interactivity.

Besides x-dimensional head movement (moving one's head left or right), z-dimensional head movement (moving one's head front or back) is also very natural for a sitting observer. However, while interactive streaming for x-dimensional view navigation has been investigated extensively [10–15], to the best of our knowledge, *the problem of synthesizing viewpoint images corresponding to large z-dimensional virtual camera movements using the color-plus-depth format has not been formally studied from a classical image interpolation perspective*. We address this problem formally in this paper, extending the capabilities of previous interactive free-viewpoint systems.

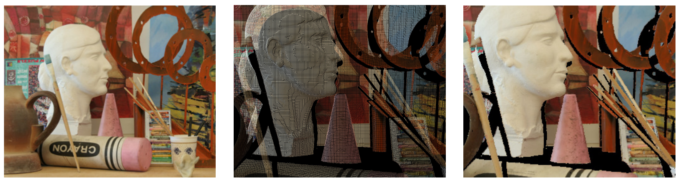

(a) captured far view (b) DIBR-synthesized (c) expansion holes filled

Fig. 1: Examples of disocclusion and expansion holes: a) camera-captured color map; b) z-dimensional DIBR-synthesized view, where *disocclusion holes* are larger contiguous empty regions next to foreground object boundaries, and *expansion holes* are smaller empty regions on the surfaces of foreground objects; c) synthesized view with expansion holes filled by our proposed scheme.

Y. Mao, G. Cheung and Y. Ji are with National Institute of Informatics (NII) and The Graduate University for Advanced Studies (SOKENDAI), 2-1-2, Hitotsubashi, Chiyoda-ku, Tokyo, Japan 101–8430 E-MAIL: {mao, cheung, kei}@nii.ac.jp, TEL:+81-03-4212-{2668, 2567, 2525}, FAX: +81-03-4212-2120.

This paper is published with the support of Sokendai's journal publishing fundings.

When the virtual camera is located closer to the 3D scene than the reference view camera, objects close to the camera will increase in size in the virtual view. This means that the aforementioned pixel-to-pixel mapping during DIBR from reference to virtual view cannot complete entire surfaces of rendered objects, resulting in *expansion holes* [16]. Note that expansion holes differ fundamentally from disocclusion holes in that the objects are visible in the reference view(s), but *the pixel sampling in the reference view is not sufficient for rendering in the synthesized view when z-directional camera movement is significant*. See Fig. 1 for an illustration of expansion and disocclusion holes.

In this paper, we propose a methodology to construct z-dimensional DIBR-synthesized images, including an efficient solution to the challenging expansion hole filling problem. We first perform a new DIBR pixel-mapping for z-dimensional camera movement. We then identify disocclusion holes using a depth layering procedure. To fill expansion holes, we formulate a patch-based maximum a posteriori (MAP) problem, where the patches are appropriately spaced using diamond tiling. Leveraging on recent advances in graph signal processing (GSP) [17], we define a graph-signal smoothness prior to regularize the inverse problem. Finally, we design a fast iterative reweighted least square (IRLS) algorithm [18] to solve the posed problem efficiently. Experimental results show that our z-dimensional synthesized images outperform images rendered by a naïve modification of VSRS 3.5 by up to 4.01dB in PSNR, and noticeably in two other quality metrics, SSIM [19] and 3DSwIM [20]. We claim that we are the first to rigorously formulate the expansion hole filling problem in DIBR images as a MAP problem, and provide a graph-based interpolation algorithm that solves it efficiently.

The outline of the paper is as follows. We first discuss related work in Section II and overview GSP theory and techniques in Section III. We describe our free view synthesis system in Section IV, including the new DIBR pixel-mapping procedure for z-dimensional camera movement. We formulate our expansion hole filling problem in Section V, and describe our joint denoising and expansion hole interpolation algorithm in Section VI. Finally, experimentation and conclusions are presented in Section VII and VIII, respectively.

II. Related Work

We divide our discussion of related work into three subsections. First, we discuss previous DIBR techniques in Section II-A. We then discuss notable works in image super-resolution in Section II-B. Finally, we discuss related works using GSP techniques for inverse imaging problems in Section II-C.

A. Depth-Image-Based Rendering

Color-plus-depth format [2], consisting of one or more color and depth image pairs from different viewpoints, is a widely used 3D scene representation. Using this format, low-complexity DIBR view synthesis procedure such as 3D warping [21] can be used to create credible virtual view images, with the aid of inpainting algorithms to complete disocclusion holes [5–7]. While the conventional approach [4] transmits two (or more) pairs of color and depth maps from neighboring viewpoints for synthesis of an intermediate virtual view, recently, the authors in [22, 23] have shown that transmission of a single color-depth map pair can be more rate-distortion (RD) optimal, *if* the resulting larger disocclusion holes can be properly filled. In this work, we assume that enough pixels from one or more reference view(s) have been transmitted to the decoder for virtual view synthesis, and we focus only on the construction of z-dimensional DIBR-synthesized images given received reference view pixels.

A few recent works on DIBR have investigated view synthesis for large z-dimensional camera movements. The EU project DIOMEDES [24] proposed to enlarge pixel size uniformly to eliminate cracks due to a large change in viewing angle. While this technique can also be used for z-dimensional camera movements, the absence of an advanced interpolation technique results in unsatisfactory synthesis quality. Further, it introduces undesirable blurring in the rendered image. Another EU project MUSCADE [25] also enlarged the pixel sizes, but avoided unnecessary blurring by adaptively adjusting individual pixel enlargement rate. However, the lack of a sophisticated interpolation method remains a problem when there is a large z-dimensional movement. A depth-layering based interpolation method is proposed in [26], where each pixel in the virtual view is back projected to the reference view and then interpolated by bicubic interpolation. However, to be shown in Section VII, bicubic interpolation in general cannot generate satisfactory images synthesis quality.

As an alternative to color-plus-depth image-based representation of the 3D scene, the captured images can also be first converted to a triangular mesh representation at the encoder [27]. At the decoder, given only color and depth values of the mesh points with no explicitly coded surface curvature information, each pixel on the 2D grid in the virtual image is then linearly interpolated using nodes that define the enclosing triangle. We will show in Section VII that our proposed expansion hole filling scheme outperforms a competing linear interpolation scheme significantly. Lumigraph-based rendering techniques [28] can synthesize a new viewpoint image from multiple closely spaced color images with camera position information. However, because the inputs are fundamentally different from the considered color-plus-depth format, the synthesis algorithm is not suitable for a typical DIBR system.

B. Image Super-Resolution

Increase in object size due to large z-dimensional virtual camera movement is analogous to increasing the resolution (*super-resolution* (SR)[1]) of the whole image. However, during z-dimensional camera motion an object closer to the camera increases in size faster than objects farther away, while in SR, resolution is increased uniformly for all spatial regions in the image. For the above reason, we cannot directly apply conventional image SR techniques [30] in rectangular pixel grid to interpolate the synthesized view. Further, recent non-local SR techniques such as [31] leveraging on *self-similarity* of natural images that require an exhaustive search of similar patches throughout an image tend to be computationally expensive. In contrast, our interpolation scheme performs only iterative local filtering, and thus is significantly more computation-efficient.

C. Graph-based Image Processing

GSP is the study of signals that live on structured data kernels described by graphs [17], leveraging on spectral graph theory [32] for frequency analysis of graph-signals. Graph-signal priors have been derived for inverse problems such as denoising [33–35], interpolation [36, 37], bit-depth enhancement [38] and de-quantization [39]. The common idea among these works is the assumption that the desired graph-signal is smooth with respect to a properly chosen graph $\mathcal{G}$ that reflects the structure of the signal. Typically one assumes that the appropriate graph $\mathcal{G}$ is known *a priori* as available side information, or can be discovered from noisy and / or partial observations of the signal. In this work, we assume the latter case and construct a suitable graph $\mathcal{G}$ from available DIBR-synthesized pixels for joint denoising / interpolation of pixels in a target patch.

III. Graph Signal Processing & Spectral Analysis

We now overview the basic concepts in graph signal processing (GSP) and graph spectral analysis [17]. GSP is the study of signals on structured data kernels described by graphs; as done in [17], we will focus on undirected graphs with non-negative edge weights. A weighted undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$ consists of a finite set of vertices $\mathcal{V}$ with cardinality $|\mathcal{V}| = N$, a set of edges $\mathcal{E}$ connecting vertices, and a weighted adjacency matrix $\mathbf{W}$. $\mathbf{W}$ is a real $N \times N$ symmetric matrix, where $W_{i,j} \geq 0$ is the weight assigned to the edge (i, j) connecting vertices i and j, $i \neq j$. $W_{i,j} = 0$ means vertices i and j are not connected, *i.e.*, $(i, j) \notin \mathcal{E}$.

Given a defined graph $\mathcal{G}$, the *degree matrix* $\mathbf{D}$ is a diagonal matrix whose i-th diagonal element is the sum of all elements in the i-th row of $\mathbf{W}$, *i.e.*, $D_{i,i} = \Sigma_{j=1}^{N} W_{i,j}$. The *combinatorial graph Laplacian* $\mathbf{L}$ (graph Laplacian for short) is then defined as:

$$\mathbf{L} = \mathbf{D} - \mathbf{W} \tag{1}$$

[1]The goal of SR "is to recover a high resolution image from one or more low resolution input images." [29]

TABLE I: Selected notations

Q	voxel in the 3D space
V_i, V_j	view points
V_0, V_1, V_2	view points
P_0, P_1, P_2	projections of Q on the image plane
μ_L, l	mean and scale parameter of Laplace distribution
p	target pixel to optimize
$\mathcal{R}$	analysis window defined around p
$S_w[p]$	structural tensor on p
$\mathbf{C}_p$	coordinates of pixel p in the image
λ_1, λ_2	eigen-values of the structural tensor
ρ	elongation factor to define the ellipse
ϕ	scaling factor to define the ellipse
$\mathbf{s}$	available pixels in the selecting ellipse
$\mathbf{s}^o$	ground truth signal in the selecting ellipse
$\hat{\mathbf{s}}$	signal reconstructed by our formulation

Because $\mathbf{L}$ is a real symmetric matrix, there exists a set of eigenvectors ϕ_i with corresponding real eigenvalues λ_i that decompose $\mathbf{L}$, *i.e.*,

$$\mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^T = \sum_i \lambda_i \phi_i \phi_i^T = \mathbf{L} \tag{2}$$

where $\mathbf{\Lambda}$ is a diagonal matrix with eigenvalues λ_i ordered from smallest to largest on its diagonal, and $\mathbf{\Phi}$ is an eigenvector matrix with corresponding eigenvectors ϕ_i as its columns. It can be shown that $\mathbf{L}$ is positive semi-definite [17], *i.e.* $\mathbf{x}^T\mathbf{L}\mathbf{x} \geq 0$, $\forall \mathbf{x} \in \mathbb{R}^N$, which implies that the eigenvalues are non-negative, *i.e.* $\lambda_i \geq 0$. The eigenvalues can be interpreted as frequencies of the graph; thus using $\mathbf{\Phi}^T$ as a transform, any input graph-signal $\mathbf{x}$ can be decomposed into its graph frequency components via $\mathbf{\Phi}^T\mathbf{x}$, where $\alpha_i = \phi_i^T\mathbf{x}$ is the i-th frequency coefficient. $\mathbf{\Phi}^T$ is called the *graph Fourier transform* (GFT) [40].

The expression $\mathbf{x}^T\mathbf{L}\mathbf{x}$—called the *graph Laplacian regularizer* [35]—captures the total variation of the signal $\mathbf{x}$ with the respect to the graph $\mathcal{G}$ [41]:

$$\mathbf{x}^T\mathbf{L}\mathbf{x} = \frac{1}{2}\sum_{(i,j)\in\mathcal{E}} W_{i,j}\,(x_i - x_j)^2. \tag{3}$$

In words, $\mathbf{x}^T\mathbf{L}\mathbf{x}$ is small if connected nodes x_i and x_j have similar values for each edge $(i, j) \in \mathcal{E}$, *or* if the weight $W_{i,j}$ is small.

$\mathbf{x}^T\mathbf{L}\mathbf{x}$ can alternatively be expressed in terms of graph frequencies λ_i:

$$\mathbf{x}^T\mathbf{L}\mathbf{x} = \sum_i \lambda_i \alpha_i^2 \tag{4}$$

Thus a small $\mathbf{x}^T\mathbf{L}\mathbf{x}$ means that the energy of signal $\mathbf{x}$ is mostly concentrated in the low graph frequencies—*smooth* with respect to the defined graph. We will employ the Laplacian regularizer $\mathbf{x}^T\mathbf{L}\mathbf{x}$—*i.e.*, the desired signal is mostly smooth with respect to a defined graph—as our image prior in the subsequent section.

IV. System Overview

A. Interactive Free Viewpoint Streaming System

We first overview our interactive free viewpoint streaming system; an illustration is shown in Fig. 2. As in conventional free viewpoint TV systems [1], we assume that a 1D array of closely spaced cameras along the

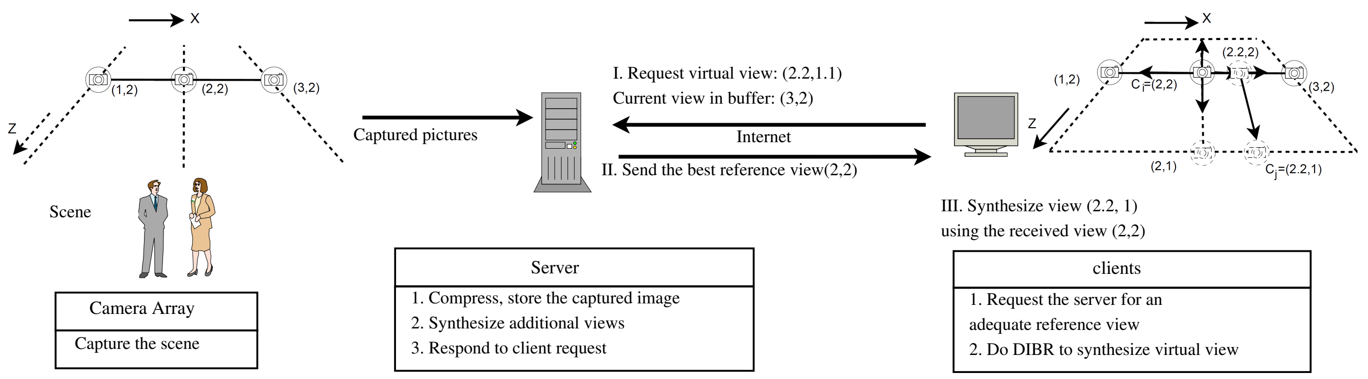

X
(1,2)
(2,2)
(3,2)
Z
Scene
Camera Array
Capture the scene
Captured pictures
I. Request virtual view: (2.2,1.1)
Current view in buffer: (3,2)
Internet
II. Send the best reference view(2,2)
Server
1. Compress, store the captured image
2. Synthesize additional views
3. Respond to client request
III. Synthesize view (2.2, 1)
using the received view (2,2)
X
(1,2)
C_i=(2,2)
(2.2,2)
(3,2)
Z
(2,1)
C_j=(2.2,1)
clients
1. Request the server for an
adequate reference view
2. Do DIBR to synthesize virtual view

Fig. 2: Interactive free view streaming system and the example of how the system present the requested virtual view, $V_j = (2.2, 1), V_i = (2, 2)$

horizontal x-axis is used to capture a 3D scene from different viewpoints. At each viewpoint, both color and depth images are captured by a camera[2]. Alternatively, depth images can be computed via a stereo-matching algorithm [42]. Both color and depth images are compressed and stored at a server using standard coding tools like MVC and 3D-HEVC [3, 43]. Representing a 3D scene using multiple viewpoints of color and depth images is called "color-plus-depth" or "video-plus-depth" in the 3D imaging literature [2, 27].

In *interactive multiview video streaming* (IMVS) [10, 11] there exists an asymmetry between data available at the server and data consumption at the client. Specifically, while the server contains coded 3D data that can synthesize a large number of views, a client can observe only one viewpoint of the 3D scene at a time rendered on a 2D display. Thus, in IMVS the client will periodically request new views for observation, and in response the server must transmit appropriate data for rendering of the requested views.

More specifically, in a conventional IMVS system [10, 11], given a set of color-plus-depth image pairs at camera-captured viewpoints $\mathcal{X} = \{1, \ldots, N\}$ at some fixed z-coordinate z_0 relative to the 3D scene, a client can select any viewpoint x between leftmost view 1 and rightmost view N to observe the scene. A virtual view x, $1 < x < N$ and $x \notin \mathbb{Z}^+$, can be synthesized using color and depth image pairs of the nearest left and right camera-captured views $\lfloor x \rfloor$ and $\lceil x \rceil$ via DIBR [4] (to be discussed in detail). Thus, when a client requests virtual view x, color and depth map pairs at one or both camera-captured viewpoints $\lfloor x \rfloor$ and $\lceil x \rceil$ (an RD decision based on bandwidth cost and synthesized view quality [22, 23]) will be transmitted to the client.

In contrast, in our enhanced IMVS system a client can in addition synthesize images at virtual camera location (x, z), where the z-coordinate can differ from the captured cameras'—*i.e.* $z \neq z_0$—as shown in Fig. 2. A virtual camera location (x, z_1) for small z_1 means that the viewpoint is closer to the 3D scene than camera location (x, z_0), $z_1 < z_0$. It means that objects close to the camera now appear bigger, and the field-of-view is narrower; see Fig. 1(c) for an example. *Note that in general one cannot practically set up a 2D camera array along both x- and z-dimension; the physical camera closer to the scene along the z-dimension will be visible and obstruct the view of a camera further from the scene.* Thus, given only a conventional 1D camera array setup, a client must synthesize a novel image at virtual viewpoint (x, z_1) using *only* color and depth map pairs at camera-captured views $(\lfloor x \rfloor, z_0)$ and / or $(\lceil x \rceil, z_0)$. As shown in Fig. 1(b), this results in expansion holes that require filling; the formulation and algorithm for expansion hole filling are discussed in Section V and VI respectively.

In the next section, we first overview the x-dimensional DIBR pixel-to-pixel mapping procedure, and then extend it to z-dimensional mapping. We then describe how we distinguish between two different kinds of missing pixels in a z-dimensional DIBR-synthesized image, each requiring a different filling method. For simplicity, we only describe the DIBR view synthesis procedure using a single reference color and depth map pair in the following discussions.

B. Depth-Image-Based Rendering

We now review the image capturing process using the popular *pinhole camera* model [44] and DIBR for x-dimensional camera movement.

1) Pinhole camera model: The optical system can be simplified as a *pinhole camera* model, shown in Fig. 3. In the figure, the camera's *aperture* is denoted by V_0 and serves as the center of the camera system. The *focal length*—the distance between the aperture and the *image plane*—is denoted by f. The image plane is orthogonal to the system's *principal axis*, which indicates the camera's viewing direction and goes through V_0.

A 3D coordinate system is established using V_0 as its origin. γ_0 is the "depth" distance between a voxel in the 3D scene and V_0—a projected component onto the principal axis. α_0 is the horizontal distance between

[2]Depth sensors like Microsoft Kinect can capture both color and depth images simultaneously.

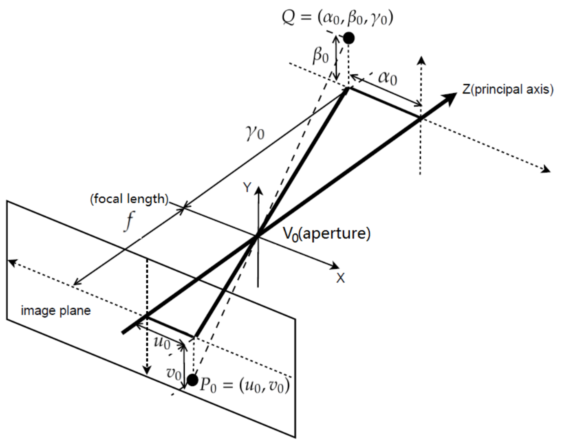


Fig. 3: Projection of a 3D voxel onto an image plane using a pinhole camera model.

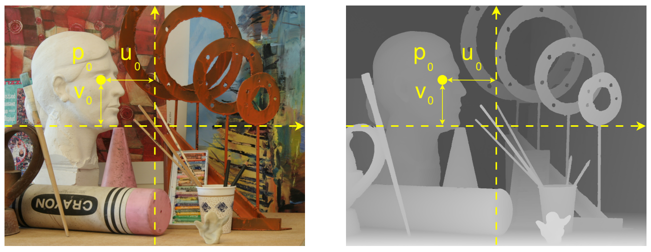
the voxel and the principal axis. Given this coordinate system, in Fig. 3 a voxel Q in the 3D space with a particular color intensity is projected through V_0 onto the image plane as pixel P_0. Note that the image projected by a pinhole camera is flipped upside down and left to right; in the sequel we revert this inversion to present the captured image more naturally. An example of the inverted captured image is shown in Fig. 4(a).

In the inverted image plane, pixel P_0's 2D coordinates in the captured image are denoted by (u_0, v_0), where u_0 can be computed using similar triangles:

$$u_0 = \alpha_0 \frac{f}{\gamma_0} \tag{5}$$

The vertical coordinate v_0 can be computed using the same procedure.

The described mapping procedure from voxels in 3D space to pixels on a 2D image plane retains color information as observed from a particular camera viewpoint. The same procedure can also be used to capture a depth image, where each 3D voxel Q now reflects its depth value γ_0 (or its reciprocal, *disparity*, $1/\gamma_0$). Fig. 4(b) shows a captured depth image from the same viewpoint.



(a) texture image (b) depth image

Fig. 4: Texture / depth image pair from the same camera viewpoint.

2) DIBR for x-dimensional camera movement: We now overview the DIBR image synthesis process when the camera movement is restricted to the x-dimension. In Fig. 5(a), we show the top view of the previously described optical system, allowing us to focus on the geometric relationships among objects on the x-z plane. A virtual camera with aperture $V_1 = (x_0 - \Delta x, z_0)$ is located at distance Δx from the reference camera with aperture V_0, resulting in a corresponding shift in the principal axis. The location of the image plane remains the same, though the center also moves from V_0 to V_1. The previously projected pixel $P_0 = (u_0, v_0)$ in the old image plane can now be translated to a location P_1 in the new image plane as follows. First, pixel P_0 is back-projected to position Q in the 3D space, and then is re-projected to location $P_1 = (u_1, v_0)$ in the new image plane. Again using similar triangles, we can calculate u_1:

$$u_1 = \alpha_1 \frac{f}{\gamma_0} = \alpha_0 \frac{f}{\gamma_0} + (\alpha_1 - \alpha_0)\frac{f}{\gamma_0} = u_0 - \Delta x \frac{f}{\gamma_0} \tag{6}$$

Thus, given the x-dimensional camera movement by Δx, the new horizontal coordinate u_1 can be computed using f and γ_0. Further, for x-dimensional camera movement there is no change in the vertical coordinate. The original and synthesized images after x-dimensional camera movement are shown in Fig. 5(b) and (c) respectively, where the pixels of the sculpture's eye are highlighted.

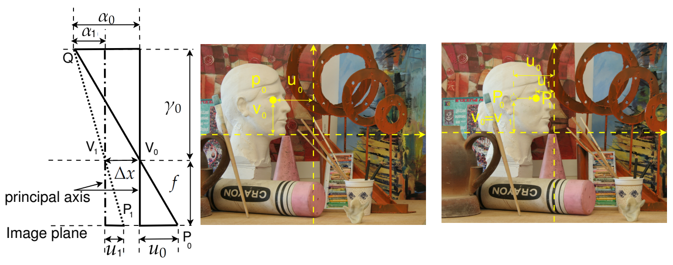
Note that multiple pixels from the old image plane with different depth values may be mapped to the same pixel location in the new image. In this case, we keep the pixel with the shallowest depth, as foreground objects occlude background objects. Note also that there are *holes* in the synthesized virtual view, *i.e.,* a pixel location in the virtual view that has no corresponding pixel in the reference view. There are three kinds of holes. The first is *disocclusion holes*, which are spatial locations that are occluded by foreground object(s) in the reference view, but become exposed in the virtual view. Disocclusion holes are large continuous areas appearing between foreground and background. There have been many depth-based image inpainting techniques proposed to complete disocclusion holes [5–7].

The second kind is *out-of-view* holes, which appear at the left or right boundaries of the image. They exist because the image's field of view has changed due to the x-dimensional camera movement. The out-of-view holes are filled using the same method as disocclusion holes.

The last kind of holes, known as *rounding holes*, appear when an object covers a slightly larger spatial area in the synthesized image due to the change of viewpoints. This change in size is typically very small during x-dimensional camera movement, and rounding holes are filled using simple local interpolation methods, such as bilinear interpolation [4].

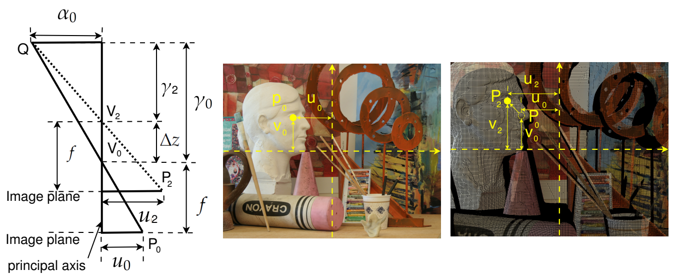
Finally, we note that in practice a pixel (u_0, v_0) in the reference view is copied to the *nearest integer position* to (u_1, v_0), *i.e.* $(u_0 + \text{round}(\Delta u), v_0)$, on the 2D image grid of the virtual view. This rounding operation introduces errors in the synthesized view. We will study the error due to this rounding operation closely later.

3) DIBR for z-dimensional camera movement: In this paper we extend DIBR to enable z-dimensional camera



(a) movement (b) reference (c) synthesized

Fig. 5: DIBR for x-dimensional camera movement.



(a) movement (b) reference (c) synthesized

Fig. 6: DIBR for z-dimensional camera movement.

movement also. The top view of the optical system is again shown in Fig. 6(a), where the virtual camera with aperture $V_2 = (x_0, z_0 - \Delta z)$ is shifted from reference camera with aperture V_0 by Δz along the principal axis. The image plane is shifted correspondingly while the principal axis remains the same. Thus, when a pixel (u_0, v_0) on the old image plane is translated to the new image plane, we can compute the new horizontal coordinate u_2 again using similar triangles:

$$u_2 = \alpha_0 \frac{f}{\gamma_2} = u_0 \frac{\gamma_0}{\gamma_2} = u_0 \frac{\gamma_0}{\gamma_0 - \Delta z} \tag{7}$$

Unlike the x-dimensional camera movement, the vertical coordinate of a pixel also changes during a z-dimensional camera movement. The new vertical coordinate can be calculated similarly as follows:

$$v_2 = v_0 \frac{\gamma_0}{\gamma_0 - \Delta z} \tag{8}$$

The original and synthesized images after the z-dimensional camera movement towards the 3D scene are shown in Fig. 6(b) and (c) respectively, with example pixels highlighted. Besides holes we encountered during a x-dimensional camera movement, we now have a scattering of missing pixels over the surfaces of objects. We call this new kind of holes—unique for z-dimensional camera movement—*expansion holes*.

Though in principle the expansion holes are similar to rounding holes, they are much larger in size when there is a large z-dimensional movement, and hence the interpolation quality strongly influences the overall synthesized image perception. We will describe a scheme specifically for expansion hole filling in a later section.

Note that though we focus on the z-directional camera movement in this paper, in practice when any mixture of x-/y-/z-directional camera movements is possible, we can take the following strategy. First, we differentiate between disocclusion holes and expansion holes. Second, we determine if the amount of z-directional movement is larger than a threshold. If so, our algorithm can be used for interpolation of expansion holes. If not, a local hole filling strategy as done in VSRS software is employed.

C. Rounding Noise in DIBR-mapped pixels

As previously discussed, when we compute new coordinates for pixels mapped from the reference to virtual view, the computed quantities are in general not integers and must be rounded to the nearest 2D grid positions for display, resulting in errors we call *rounding noise*. *To the best of our knowledge, the extent and characteristics of rounding noise in the DIBR pixel-to-pixel mapping procedure have not been studied systematically in the literature before.* We characterize such rounding noise in this section empirically.

We first seek a suitable statistical description for rounding noise. Specifically, we compute the differences between DIBR-mapped pixels from reference views and camera-captured ground truth pixels of the same views to construct an error distribution. We then describe the resulting distribution using two popular noise models—Gaussian and Laplacian distribution with probability density functions (PDF):

$$f_{Gaussian}(x|\mu_G, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{(x-\mu_G)^2}{2\sigma^2}\right\} \tag{9}$$

$$f_{Laplacian}(x|\mu_L, l) = \frac{1}{2l} \exp\left\{-\frac{|x-\mu_L|}{l}\right\} \tag{10}$$

where μ_G and μ_L are the means, and σ^2 and $2l^2$ are the variances of the distributions respectively. The reason we restrict our attention to only Gaussian and Laplacian distributions is because the subsequent MAP problem formulations stemming from these models become relatively straightforward l_2- and l_1-norm minimization, solvable via efficient iterative algorithms. Problem formulation and corresponding optimization algorithms are described in Section V and VI respectively.

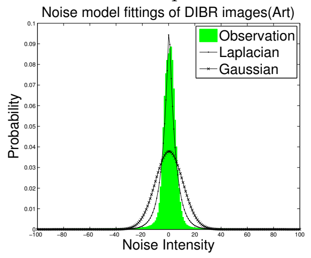
Using squared error as a criteria, we compute the error-minimizing model parameters for the Gaussian and Laplacian distributions separately. The best fitted models are shown in Fig. 7 for image sequences `Art` and `Dolls`. We observe that the root mean square errors (RMS) of the Laplacian model (0.0008 for `Art` and 0.0006 for `Dolls`) are smaller than that of Gaussian model (0.0020 for `Art` and 0.0017 for `Dolls`). Thus, we conclude that the Laplacian distribution is a better statistical description of rounding noise, and is the preferred noise model for later optimization.

D. Identification of Expansion Holes

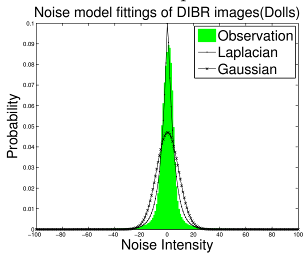
Before we can complete the expansion holes, we need to first properly identify them. For intuition, we examine

(a) `Art` sequence

(b) `Dolls` sequence



(c) DIBR noise distribution and best fitted error models for `Art`



(d) DIBR noise distribution and best fitted error models for `Dolls`

Fig. 7: Fitting noise models to DIBR noise distribution for `Art` and `Dolls`

a magnified patch of a z-dimensional DIBR-synthesized image in Fig. 8, where disocclusion holes are enclosed by yellow lines and expansion holes are enclosed by blue or green lines. Formally, 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. Since the amount of expansion is typically evenly distributed over the surface of an object, expansion holes are small "cracks" that spread all over the surface of an enlarged object. As observed, this signal characteristic is very different from the large and continuous disocclusion holes.

Further, while a disocclusion hole always appears empty, *expansion holes can often be wrongly filled with background pixels during DIBR*. The reason is because the size expansion for a foreground object surface is typically larger than a background surface during a z-dimensional camera movement, and hence in an expansion hole due to enlargement of foreground object surface, a background pixel can be mapped from reference view erroneously. Fig. 8 shows an example with a foreground detergent bottle in front of a background wooden panel. However, the pixels of the panel erroneously fill the expansion holes of the bottle. Hence it is important to correctly identify and remove these erroneously mapped background pixels *before* performing completion of expansion holes.

1) *Depth Layering to Identify Expansion Holes:* To identify pixels from the same physical object in the DIBR synthesized view, we adopt a *depth layering* approach. Specifically, for a given pixel block in the synthesized view, we first construct a histogram containing depth values of pixels in the block. Fig. 9(b) shows an example. Peaks in the histogram are labeled as layers ordered from shallow depth to deep depth. Fig. 9(a) shows the depth pixels in the block with assigned layer numbers.

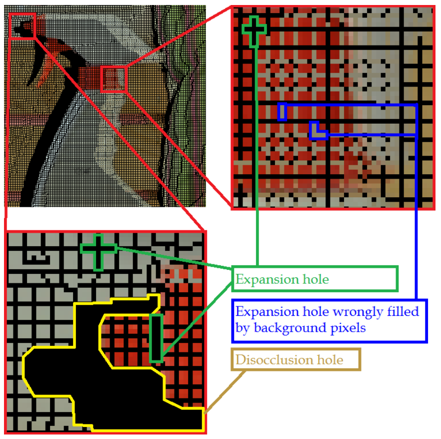
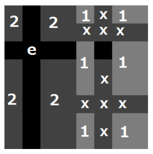
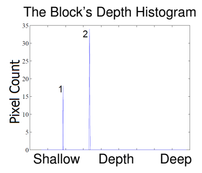


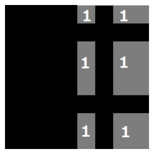
Fig. 8: Expansion holes and disocclusion holes in the output image of DIBR



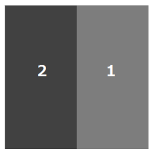
(a) block



(b) depth histogram



(c) first layer



(d) reconstructed

Fig. 9: Examples of depth layers and corresponding histogram: a) pixels in a depth block are classified into depth layers and empty pixels; b) corresponding histogram of depth values for the block; c)the first depth layer separated from the second depth layer; d) reconstructed depth block.

The identification is performed layer-by-layer, starting from the shallowest, so that when interpolating missing pixels in layer a, each pixel in layer $b > a$ that is inside a convex set spanned by pixels in layer a is treated as an empty pixel. In Fig. 9(c), we shown that layer 2 pixels are treated as empty pixels during expansion hole filling of layer 1. The reconstructed depth block is shown in Fig. 9(d). Having identified available pixels in a depth layer for interpolation of empty pixels, we next formulate our joint denoising / interpolation problem for expansion hole filling.

V. Expansion Hole Filling Formulation

Having chosen a rounding noise model and identified the expansion holes, we next denoise DIBR-mapped pixels and complete expansion holes via a unified *maximize a posteriori* (MAP) formulation for a given target patch with center at pixel p. Specifically, first we divide the image into individual patches with overlaps, where each patch contains similar pixels (Section V-A). To restore pixels in each patch, we introduce a graph-signal smoothness prior to regularize an otherwise under-determined problem (Section V-B). Finally, we formulate the MAP estimation problem in Section V-C.

A. Patch Selection via Adaptive Kernel

We first divide the pixels in the same depth layer into overlapping patches. In particular, we adaptively select patch shapes based on observed signal characteristics, because the same physical object can have distinct *textural patterns* that influence how pixels should be interpolated. For example, if the captured object is a red and blue striped shirt, then pixels inside a blue stripe should be interpolated using only neighboring blue pixels.

1) Selecting the first patch: To select the first pixel patch for joint denoising / interpolation, we first select a pixel p in the depth layer (*e.g.*, the top-left pixel), and calculate an *adaptive kernel* centered at p, similarly done in [45] (we discuss our implementation difference from [45] in details later). There are two basic steps. In the first step, the *principal gradient* in a local neighborhood is derived via computation of the *structure tensor* [46]. The structure tensor $S_w(p)$ defined on pixel p's location $\mathbf{C}_p$ is:

$$S_w(p) = \begin{bmatrix} \sum_{r\in\mathcal{R}} w(\mathbf{C}_r)(\Delta_x(\mathbf{C}_p,\mathbf{C}_r))^2 & \sum_{r\in\mathcal{R}} w(\mathbf{C}_r)\Delta_x(\mathbf{C}_p,\mathbf{C}_r)\Delta_y(\mathbf{C}_p,\mathbf{C}_r) \\ \sum_{r\in\mathcal{R}} w(\mathbf{C}_r)\Delta_x(\mathbf{C}_p,\mathbf{C}_r)\Delta_y(\mathbf{C}_p,\mathbf{C}_r) & \sum_{r\in\mathcal{R}} w(\mathbf{C}_r)(\Delta_y(\mathbf{C}_p,\mathbf{C}_r))^2 \end{bmatrix} \tag{11}$$

where $\mathcal{R}$ defines a square neighborhood around pixel p, $\Delta_x(\mathbf{C}_p,\mathbf{C}_r)$ and $\Delta_y(\mathbf{C}_p,\mathbf{C}_r)$ are the color image gradients[3] along the x- and y-axis at pixel p respectively, and $w(\mathbf{C}_r)$ is a weight assigned to neighbor r. Weights are determined by a Gaussian kernel, which is normalized so that $\sum_{r\in\mathcal{R}} w(\mathbf{C}_r) = 1$.

Having computed $S_w(p)$, we perform eigen-decomposition on the matrix, which summarizes the local gradients within $\mathcal{R}$. The ratio between the larger λ_2 and smaller eigenvalue λ_1 indicates relative strength of the principal gradient in the patch $\mathcal{R}$, and the eigenvector v_2 corresponding to the larger eigenvalue λ_2 indicates the direction of $\mathcal{R}$'s principal gradient. For example, in the case that a strong edge is present in the pixel patch, the principal gradient will be *orthogonal* to the edge. See Fig. 10 for an illustration.

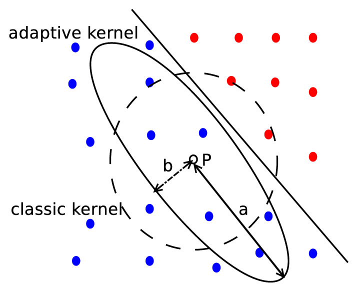


Fig. 10: Illustration of patch selection via adaptive kernel. An ellipse is elongated perpendicular to the principal gradient, so that similar pixels are selected for pixel interpolation.

[3]Gradient $\Delta(\mathbf{C}_p,\mathbf{C}_r)$ at pixel p is computed as the intensity difference from a neighbor r divided by the distance between p and r.

Finally, an ellipse centered at the target pixel p is defined to identify a subset of pixels in the same depth layer for joint denoising / interpolation. The ellipse has minor axis aligned with the tensor eigenvector v_2, and the major axis orthogonal to the minor axis. In particular, let a and b be the major and minor radius, *i.e.*, in the rotated coordinate system (x', y'),

$$\left(\frac{x'}{a}\right)^2 + \left(\frac{y'}{b}\right)^2 = 1 \tag{12}$$

We compute a and b as:

$$a = \rho\,\phi, \quad b = \rho^{-1}\,\phi \tag{13}$$

$\rho = \sqrt{\frac{\lambda_2+\omega}{\lambda_1+\omega}}$ is the *elongation factor*. ρ reflects the relative strength of the principal gradient, with $\omega > 0$ for numerical stability. Using ρ, the shape of the kernel is kept circular in flat areas, where $\lambda_1 \approx \lambda_2 \approx 0$, and elongated when near a strong edge ($\lambda_2 \gg \lambda_1$). The idea is to construct an ellipse elongated perpendicular to the principal gradient of the patch, so we can include enough similar pixels for joint denoising / interpolation.

$\phi = m\sqrt{\lambda_1\lambda_2 + \epsilon}$ is a *scaling factor*, where m is a size parameter and ϵ is used for numerical stability. Since $\sqrt{\lambda_1\lambda_2}$ is the geometric mean of the tensor's eigenvalues, ϕ induces a large kernel in a flat area to average over more pixels (better denoising), and induces a smaller kernel in a heavily textured area (avoid blurring).

In Fig. 10, an example ellipse is elongated to contain only blue neighboring pixels, resulting in a texture-adaptive pixel kernel. In contrast, a classic kernel will be a circle with a fixed radius, containing both blue and red pixels.

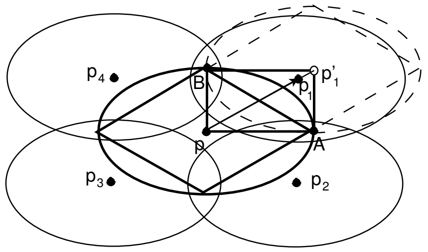


Fig. 11: Illustration of choosing next kernel center pixel

2) Selecting the next patch: We now determine the location of the next patch. The spacing of the patches should satisfy two conditions: i) cover all pixels in a depth layer for reconstruction, and ii) overlap to some controlled extent to avoid boundary artifacts. However, too large an overlap will increase the number of patches and computation complexity.

To compute an appropriate distance between two neighboring patches, we first assume that the to-be-computed kernels have the same shape and size as the just computed one. Further, we approximate a kernel's ellipse shape with a *diamond*, where the diamond's corners are the ellipse' endpoints along the major and minor axis. The appropriate spacing of the patches is then computed as the spacing of the diamonds when packed to fill a plane without overlap. It is known that a diamond—itself a concatenation of two identical triangles connected back-to-back—can be used to fill a plane without gap—a process called *tessellation*[4]. Hence the spatial areas enclosed by the tiled diamonds alone are guaranteed to cover all pixels in the 2D grid. Further, because an ellipse always properly encloses the approximating diamond, the spatial areas covered by the ellipses contain overlaps. This patch selection procedure based on diamond tiling thus satisfies both required conditions: i) covers all pixels in a depth layer for optimization, and ii) creates overlaps to eliminate boundary artifacts. See Fig. 11 for an illustration.

3) Complexity of patch-based reconstruction: We now discuss the complexity of our patch-based approach as compared to the pixel-based approach LARK [45]. First, our adaptive kernel is based on eigen-decomposition of a 2×2 structure tensor, while LARK performed singular value decomposition (SVD) of a much larger matrix. Both eigen-decomposition and SVD require $O(n^3)$ computation time, where n is the larger dimension of the target matrix. Thus for each computed kernel our implementation is significantly faster.

Second, LARK is *pixel*-based, which means that a kernel is constructed for *every* pixel for reconstruction. In contrast, our optimization is *patch*-based, and all the pixels in a patch are reconstructed simultaneously. Thus the number of calculated adaptive kernels in our method is reduced compared to LARK by a factor equals to the average non-overlapping area inside a diamond, which is approximately 6 in our experiments. We can thus estimate that our patch-based approach is no more than 1/6 the computation cost of LARK.

Finally, we note that our scheme is inherently local, which is much faster than non-local schemes like non-local means (NLM) [47] that rely on searches of similar patches in distant spatial areas and thus is computationally much more complex.

B. Graph Construction and Graph Laplacian Prior

Having identified a subset of pixels in the same depth layer suitable for interpolation, we next define a graph $\mathcal{G}$ connecting these pixels for graph-based optimization as follows. Each pixel in the kernel ellipse is represented as a node in the graph $\mathcal{G}$. We draw four edges between each pixel and its four nearest neighbors in terms of Euclidean distance, and each edge weight $e_{p,q}$ between two pixels p and q is computed as:

$$\begin{aligned} e_{p,q} &= w_{p,q} v_{p,q}, \\ w_{p,q} &= \exp\left\{ -\frac{\left\| I_p - I_q \right\|_2^2}{\sigma_I^2} \right\}, \\ v_{p,q} &= \exp\left\{ -\frac{\left\| \mathbf{C}_p - \mathbf{C}_q \right\|_2^2}{\sigma_{\mathbf{C}}^2} \right\} \end{aligned} \tag{14}$$

[4]https://en.wikipedia.org/wiki/Tessellation

where I_p and C_p are the intensity and Cartesian grid coordinate for pixel p respectively. σ_I and $\sigma_{\mathbf{C}}$ are chosen parameters. This exponential edge weight assignment is similar to one used in bilateral filter [48].

Having constructed the graph, we compute the graph Laplacian matrix $\mathbf{L}$ described in Section III to define the prior probability of the underlining signal $\mathbf{s}^o$, a vector composed of the n pixels in the graph $[s_1^o, \ldots, s_n^o]$:

$$Pr(\mathbf{s}^o) = C \exp\left\{ -d \ \mathbf{s}^{oT} \mathbf{L}^h \mathbf{s}^o \right\} \tag{15}$$

where d and h are chosen parameters for the distribution and C is chosen so that the distribution integrates to one. The graph-signal smoothness prior promotes signals that are smooth with respect to the defined graph.

C. Graph-Based Pixel Interpolation via MAP Formulation

We now derive the objective of our MAP formulation. Without loss of generality, let the n_s synthesized pixels, n_p previously interpolated pixels and $n - n_s - n_p$ empty pixels in the kernel be arranged in order in signal $\mathbf{s}$; *i.e.,* the signal has initial observation $[s_1, \ldots, s_{n_s}, p_{n_s+1}, \ldots, p_{n_s+n_p}, 0, \ldots, 0]$. Let $\mathbf{u}_i$'s be a set of $n_s + n_p$ length-n unit vectors, $[0, \ldots, 0, 1, 0, \ldots, 0]$, where the single non-zero entry is at position i. As previously discussed, the error between the synthesized pixels in $\mathbf{s}$ and the underlining signal $\mathbf{s}^o$ follows a Laplacian distribution. Thus, our *likelihood* will be the product of a series of probability density functions:

$$Pr(\mathbf{s} \,|\, \mathbf{s}^o) = \prod_{i=1}^{n_s} \frac{1}{2l} \exp\left\{ -\frac{|\mathbf{u}_i^T \mathbf{s}^o - s_i|}{l} \right\} \tag{16}$$

Given the defined prior and likelihood, we now formulate a MAP estimation problem to find the most probable $\mathbf{s}$, where the posterior probability is replaced by the product of the prior probability and the likelihood:

$$\begin{aligned} & Pr(\mathbf{s}^o|\mathbf{s}) \propto Pr(\mathbf{s}|\mathbf{s}^o) \times Pr(\mathbf{s}^o) = \\ & \prod_{i=1}^{n_s} \frac{1}{2l} \exp\left\{ -\frac{|\mathbf{u}_i^T \mathbf{s}^o - s_i|}{l} \right\} \cdot C \exp\left\{ -p \ \mathbf{s}^{oT} \mathbf{L}^h \mathbf{s}^o \right\} \end{aligned} \tag{17}$$

Then the estimation $\hat{\mathbf{s}}$ of $\mathbf{s}^o$ is computed as the argument that minimizes the negative log of our formulated objective:

$$\min_{\hat{\mathbf{s}}} \ \sum_{i=1}^{n_s} |\mathbf{u}_i^T \hat{\mathbf{s}} - s_i| \ + \ \mu \, \hat{\mathbf{s}}^T \mathbf{L}^h \hat{\mathbf{s}} \tag{18}$$

where for simplicity μ replaces various constants in previous formulation.

Further, during a particular patch-based optimization, the optimized solution $\hat{\mathbf{s}}$ should be consistent with previously optimized pixels $p_{n_s+1}, \ldots, p_{n_s+n_p}$ in the ellipse. We apply the following constraint on the difference between $\hat{\mathbf{s}}$ and previously optimized pixels:

$$\sum_{j=n_s+1}^{n_s+n_p} (\mathbf{u}_j^T \hat{\mathbf{s}} - p_j)^2 \leq \tau \tag{19}$$

We describe an efficient algorithm to solve (18) with constraint (19) next.

VI. Expansion Hole Filling Algorithm

A. Lagrangian Relaxation

To solve the formulated constrained optimization problem directly is difficult, and so we convert it into an unconstrained optimization via Lagrangian relaxation:

$$\min_{\hat{\mathbf{s}}} \sum_{i=1}^{n_s} |\mathbf{u}_i^T \hat{\mathbf{s}} - s_i| \ + \ \mu\, \hat{\mathbf{s}}^T \mathbf{L}^h \hat{\mathbf{s}} \ + \ \nu \sum_{j=n_s+1}^{n_s+n_p} (\mathbf{u}_j^T \hat{\mathbf{s}} - p_j)^2 \quad (20)$$

where weight parameter $\nu > 0$ must be chosen so that the original constraint (19) is met.

Our new objective is then to minimize a weighted sum of: i) the l_1-norm of the difference between interpolated signal $\mathbf{x}$ and n_s synthesized pixels s_i's , ii) smoothness prior $\hat{\mathbf{s}}^T \mathbf{L}^h \hat{\mathbf{s}}$, and iii) penalty term to penalize the inconsistency between the interpolated signal and n_p previously optimized pixels p_j's.

B. Iterative reweighted Least Square Algorithm

The objective (20) is a combination of one l_1-norm term and two l_2-norm terms. To solve the problem efficiently, we leverage on the idea of *iterative reweighted least square* (IRLS) [18], where the l_1-norm fidelity term of $\hat{\mathbf{s}}$ in (18) is replaced by a weighted l_2-norm:

$$\min_{\hat{\mathbf{s}}} \sum_{i=1}^{n_s} w_i (\mathbf{u}_i^T \hat{\mathbf{s}} - s_i)^2 \ + \ \mu\, \hat{\mathbf{s}}^T \mathbf{L}^h \hat{\mathbf{s}} \ + \ \nu \sum_{j=n_s+1}^{n_s+n_p} (\mathbf{u}_j^T \hat{\mathbf{s}} - p_j)^2 \quad (21)$$

where the weights w_i's are calculated as:

$$w_i = \frac{1}{|\mathbf{u}_i^T \hat{\mathbf{s}} - s_i| + \epsilon'} \quad (22)$$

where ϵ'>0 is used for numerical stability.

With weights calculated by (22), the weighted l_2-norm (21) minimizing solution $\mathbf{s}^*$ will promote a solution $\hat{\mathbf{s}}$ to the original problem with small l_1-norm. However, $\hat{\mathbf{s}}$ is unknown beforehand. To find appropriate weights w_i's for (21) to approximate (18), we design an algorithm, where (21) is solved iteratively, with weights $w_i^{(t)}$'s at iteration t computed as:

$$w_i^{(t)} = \frac{1}{|\mathbf{u}_i^T \mathbf{s}^{*(t-1)} - s_i| + \epsilon'} \quad (23)$$

where $\mathbf{s}^{*(t-1)}$ is the solution of the previous iteration $t-1$.

The idea is that in the iterative algorithm, one can assume each iteration's $\mathbf{s}^{*(t)}$ serves as a good estimate to optimal solution $\hat{\mathbf{s}}$. Hence, we can define the weights using (23), so the iteratively weighted l_2-norm can mimic the l_1-norm. Algorithm 1 shows how we adopt IRLS for optimizing (21). Each iteration of the algorithm is an unconstrained quadratic programming problem, which can be solved efficiently in closed form:

$$\mathbf{s}^{*(t)} = (\mathbf{W}^{(t)} + \mathbf{V} + \mu \mathbf{L}^h)^{-1} (\mathbf{W}^{(t)} + \mathbf{V}) \mathbf{s} \quad (24)$$

where $\mathbf{W}^{(t)}$ is a diagonal matrix with $w_i^{(t)}$'s as its first n_s diagonal elements, and $\mathbf{V}$ is another diagonal matrix whose (n_s+1)-th through (n_s+n_p)-th diagonal elements are ν. The initial weights are calculated with $\mathbf{s}_0$ where the rendered pixels are kept and the missing pixels are filled with bilinear interpolation.

Algorithm 1 Iterative algorithm to solve weighted l_2-norm minimization

```
 1: $\mathbf{s}' \leftarrow \mathbf{s}_0$;
 2: while true do
 3:     $w_i \leftarrow 1/(\|\mathbf{u}_i^T \mathbf{s}' - s_i\|_1 + \epsilon')$;
 4:     $\mathbf{s}^* = (\mathbf{W} + \mathbf{V} + \mu \mathbf{L}^h)^{-1} (\mathbf{W} + \mathbf{V}) \mathbf{s}$
 5:     if round($\mathbf{s}'$) equals round($\mathbf{s}^*$) then
 6:         return round($\mathbf{s}^*$)
 7:     else
 8:         $\mathbf{s}' \leftarrow \mathbf{s}^*$
 9:     end if
10: end while
```

1) Alternative Objective: As the IRLS can require many iterations before it converges, the l_1-norm formulation can be further reduced to an unweighted l_2-norm optimization problem shown below for computation reason:

$$\min_{\hat{\mathbf{s}}} \sum_{i=1}^{n_s} (\mathbf{u}_i^T \hat{\mathbf{s}} - s_i)^2 \ + \ \mu\, \hat{\mathbf{s}}^T \mathbf{L}^h \hat{\mathbf{s}} \ + \ \nu \sum_{j=n_s}^{n_s+n_p} (\mathbf{u}_j^T \hat{\mathbf{s}} - t_j)^2 \quad (25)$$

The intuition is that the l_2 norm can also serves as a fidelity term to penalize $\hat{\mathbf{s}}$ that deviates from $\mathbf{s}$ for $i = 1 \ldots n_s$. Further, if the reference texture and depth images are coarsely quantized during compression, the noise of the synthesized image is dominated by quantization, and modeling noise as the l_1-norm is no longer necessary.

In this alternative formulation, (25) is again an unconstrained quadratic programming problem with a closed form solution, and thus, can be solved efficiently. In the experiment, the unweighted l_2-norm formulation is referred as UL2A, and applied when large QP is used for efficient computation. We will discuss it in detail in Section VII.

C. Selection of Smoothing Parameters

The amount of smoothness applied during the optimization can be adapted locally via adjustments to parameters μ and h in (21) or (25). First, h, the power of the graph Laplacian, $\mathbf{L}$, means a signal should be smooth with respect to its h-hop neighbors. In this paper, we select h to be proportional to the major radius a of the adaptive kernel ellipse. The rationale is that an elongated ellipse means more pixels geometrically farther from the target pixel is included in the kernel, and our h selection allows the filtering to smooth over more pixels when a strong patch gradient is detected (large λ_2), resulting in a sharper textural edge. In particular, we set h to be the Manhattan distance between the target pixel and the pixel at the end of the long axis for the experiment.

Unlike our previous work [49], where parameter μ—weighting the importance of the smoothness prior relative to the fidelity term—is chosen globally and heuristically, our new proposal assigns different μ's for different patches. For each patch $\mathcal{P}_v$ around a target pixel p_v in the virtual view, we first find the corresponding center pixel p_r in the reference view by reverse DIBR, and then draw a reduced-size version of $\mathcal{P}_v$ around p_r to define $\mathcal{P}_r$.

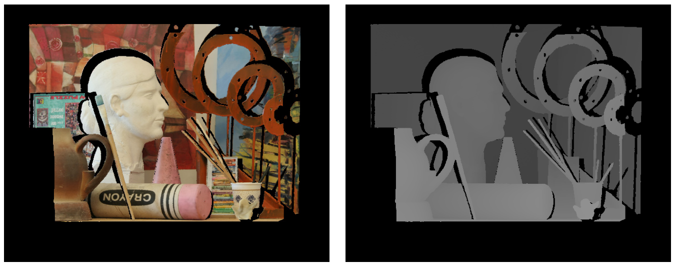
Then, we remove a selection of pixels in $\mathcal{P}_r$ and test our interpolation method using a set of parameter candidates μ's. The μ value that leads to best interpolation quality is then used to interpolate $\mathcal{P}_v$.

VII. Experimentation

A. Experimental Setup

To demonstrate the performance of our proposed z-dimensional image synthesis method, we conducted extensive experiments using the Middlebury's 2003, 2005 and 2014 datasets[5] and Nagoya datasets[6], which are multiview image sequences captured indoor using an array of cameras shifted along the x-dimension. We consider a reduction of the distance between the observer and the nearest object by half, which means the spatial resolution of the nearest object can be increased by 2x.

Without an actual array of cameras set up along both x- and z-dimension to capture different viewpoint images, we performed the following to establish ground truth. Assuming a captured image v_0 is the *near-camera* image, we first synthesized an image v_r as observed from the *far-camera*—located at roughly twice the distance from the 3D scene as the near-camera—via DIBR using v_0 as reference. This near-to-far DIBR view synthesis typically generates no expansion holes (pixel sampling in the reference view is sufficient), but has large out-of-view holes (and some disocclusion holes), since the near-camera reference views have narrower fields-of-view. To avoid the problem of filling out-of-view holes, we only considered the available field-of-view in a synthesized far-camera viewpoint image as the spatial area of interest. Examples of synthesized far-camera texture and depth images are shown in Fig. 12.



(a) `Art` color map (b) `Art` depth map

Fig. 12: Example of DIBR-synthesized far-camera color and depth images

With these synthesized narrower field-of-view far-camera images compressed using H.264 [50] with different quantization parameters (QP) as the references, we synthesized back the near-camera images $\hat{v}_0$'s via z-dimensional pixel mapping, and then identified and filled expansion holes as we propsed, and completed disocclusion holes using an existing disocclusion hole filling algorithm [7]. We compared our constructions to v_0 to compute PSNR and two other quality metrics SSIM [19] and 3DSwIM [20] to evaluate the quality of our constructed DIBR-synthesized images.

For comparison, we employed seven competing schemes. In the first scheme called `VSRS+`, we modified VSRS software version 3.5 [7] to use a single reference view for pixel mapping, and then called the default inpainting scheme in VSRS to fill in all missing pixels. The remaining three schemes first employed our proposed z-dimensional DIBR for initial pixel mapping, identified expansion holes using our proposed method, then completed expansion holes using different interpolation methods. *Note that without our proposed depth layering strategy to properly identify expansion holes, these three schemes would suffer from foreground / background confusion, as observed in the rendering results of* `VSRS+`. `Bilinear` is a conventional bilinear interpolation scheme [51], which represents the quality of mesh-based rendering techniques, as discussed in Section II-A. `Cubic+TGV` first interpolates the expansion holes via bicubic interpolation [51], then enhances the result via *Total Generalized Variation* (TGV) [52]. `LARK` is the kernel-regression based technique in [45] and can be considered the state-of-the-art among local interpolation methods. While there are recent image interpolation methods based on patch clustering, non-local methods [53, 54] or dictionary learning [55], their complexity are significantly higher than local methods. Hence we do not compare against them here for complexity reason.

Further, we compared with three methods in the literature that can be used for expansion hole filling during large z-dimensional camera movements, including two pixel-enlargement methods proposed in the EU project DIOMEDES [24] and MUSCADE [25], and the back-projection based method proposed in [26]. We denote these three techniques by `DIO`, `MUS` and `BP` respectively in the sequel.

In Section VI we described two algorithms, `IRLS` in (21) and `UL2A` in (25), where the former employs a more accurate statistical noise model and the latter is computationally faster. We tested the two algorithms using color / depth image pairs compressed with various QPs. Table II shows the difference in PSNR between the two algorithms, computed for all non-disocclusion pixels (DIBR-mapped pixels from reference view and interpolated expansion hole pixels). It shows that `IRLS` is up to 0.22dB better than `UL2A` when $QP = 4$. As QP increases, the gain of IRLS over UL2A diminishes as the quantization noise in compressed color and depth maps becomes dominant compared to rounding noise when QP becomes large. As `IRLS` requires more computation than `UL2A`, we used `IRLS` when QP is small ($QP \leq 12$) for better image quality and `UL2A` when QP is large ($QP > 12$) to reduce computation cost. Our proposed scheme is called `AGFT+` in the sequel.

[5]http://vision.middlebury.edu/stereo/data

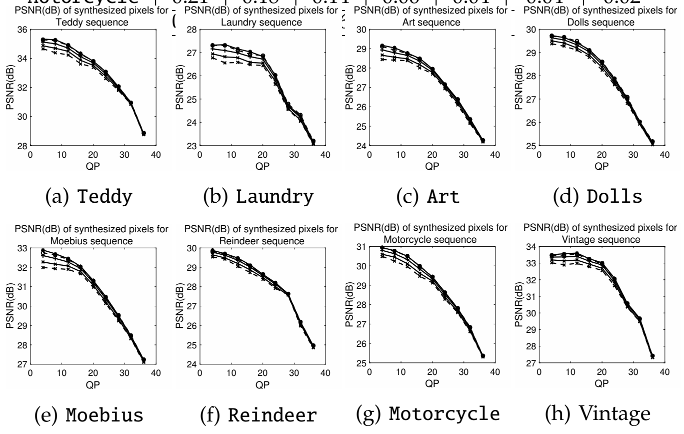
[6]http://http://www.fujii.nuee.nagoya-u.ac.jp/multiview-data

TABLE III: PSNR comparison of DIBR synthesized pixels using different methods given compressed reference views

	QP=4					QP=16				
	DIBR	AGFT+	BP	DIO	MUS	DIBR	AGFT+	BP	DIO	MUS
Teddy	35.10	**35.34**	35.27	34.86	34.66	34.10	**34.30**	34.28	33.86	33.51
Laundry	27.15	**27.30**	**27.30**	26.94	26.77	26.80	**27.01**	26.93	26.57	26.54
Art	28.85	29.08	**29.11**	28.66	28.44	28.23	28.37	**28.46**	28.17	27.77
Dolls	29.59	29.64	**29.70**	29.51	29.37	28.93	29.05	**29.08**	28.88	28.36
Moebius	32.61	**32.86**	32.75	32.27	31.99	31.93	**32.02**	32.00	31.78	31.08
Reindere	29.77	**29.83**	**29.83**	29.64	29.54	29.00	**29.12**	29.07	28.86	28.47
Motorcycle	30.80	**30.93**	30.92	30.60	30.49	29.82	**30.00**	29.98	29.62	29.21
Vintage	33.37	**33.49**	33.45	33.19	33.02	33.11	**33.25**	33.21	32.94	32.69
Akko	31.12	**31.31**	31.20	30.89	30.66	30.48	**30.58**	30.55	30.32	29.71
Balloon	29.36	**29.52**	29.28	29.18	29.02	28.68	**28.80**	28.69	28.54	28.12

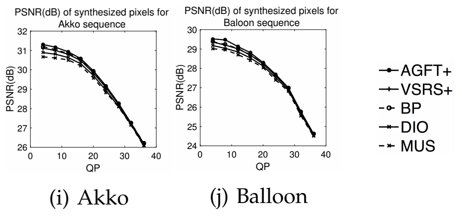
TABLE II: PSNR gain of IRLS over UL2A for different QPs

QP	4	8	12	16	20	28	36
Teddy	0.27	0.24	0.17	0.10	0.09	0.01	0.01
Laundry	0.22	0.16	0.11	0.08	0.05	0.02	0.01
Art	0.28	0.19	0.14	0.09	0.05	0.01	0.01
Dolls	0.19	0.15	0.13	0.08	0.04	0.02	0.00
Moebius	0.17	0.13	0.11	0.07	0.03	-0.01	-0.01
Reindeer	0.20	0.14	0.12	0.08	0.04	0.02	0.01
Motorcycle	0.21	0.18	0.14	0.06	0.04	0.04	0.02



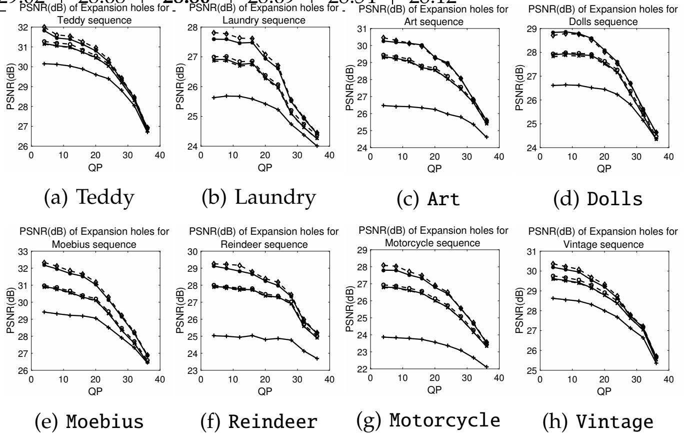
(a) Teddy (b) Laundry (c) Art (d) Dolls

(e) Moebius (f) Reindeer (g) Motorcycle (h) Vintage



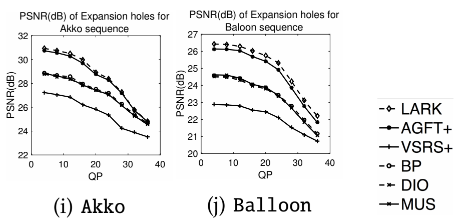
(i) Akko (j) Balloon

Fig. 13: PSNR comparison of DIBR synthesized pixels using different methods for different QPs when compressing reference views.



(a) Teddy (b) Laundry (c) Art (d) Dolls

(e) Moebius (f) Reindeer (g) Motorcycle (h) Vintage



(i) Akko (j) Balloon

Fig. 14: PSNR comparison of completed expansion holes using different methods for different QPs used when compressing reference views.

B. Experimental Comparison

1) Numerical Comparison for DIBR-synthesized Pixels: We first compare quality of DIBR-synthesized pixels with and without our proposed optimization, denoted as DIBR and AGFT+, which are shown in Fig. 13. Numerical PSNR values when $QP = 4$ and $QP = 16$ are listed in Table III. We observe that, using our proposed scheme, PSNR can be improved by up to 0.25dB when $QP = 4$, showing that the pixels synthesized by DIBR and degraded by rounding noise can by effectively restored by our algorithm. The gain diminishes as QP increases, however, as quantization becomes coarser and details become harder to recover.

We observe that the pixel-enlargement methods DIO and MUS have even lower PSNR value than DIBR as they inevitably introduces blurring in the synthesized area, degrading the image quality.

2) Numerical Comparison for Expansion Holes: We next exam reconstruction quality of expansion holes: denoised DIBR-synthesized pixels and interpolated expansion hole pixels. As shown in Fig. 14 and Table IV, for all eight sequences, Bilinear, Cubic+TGV, LARK, AGFT+ outperformed VSRS+ dramatically. This demonstrates that the correct identification of expansion holes and subsequent interpolation are crucial for z-dimensional DIBR image synthesis. Also, we see that AGFT+ outperformed Bilinear and Cubic+TGV by up to 1.84dB and 1.54dB, showing that by using our proposed interpolation method, we can achieve better image quality than the common techniques. Further, our method AGFT+ has comparable PSNR numbers as LARK while doing so at a much reduced complexity. We will show later that AGFT+ is actually better than LARK in the other two quality metrics. Comparing with BP, MUS and DIO, we observe that AGFT+ outperforms them by up to 1.89dB, 1.95dB and 2.01dB respectively in the expansion hole area due to our more advanced interpolation technique. Finally, comparing to our previous method AGFT [49] shown in gray, we see that the large reduction in computation complexity results in a very small reduction in synthesis quality.

[7]While VSRS has been updated to version 4.0, the view synthesis component remains the same as version 3.5.

TABLE IV: PSNR comparison for completed expansion holes using different methods, given compressed reference views at QP = 4 and 16

(a) QP = 4

	z-dimensional DIBR, Depth Layering					BP	MUS	DIO	VSRS+
	AGFT	AGFT+	LARK	Cubic+TGV	Bilinear				
`Teddy`	31.86	31.82	**31.99**	31.16	31.02	31.28	31.17	31.14	30.14
`Laundry`	27.62	27.58	**27.80**	26.91	26.74	27.00	26.91	26.85	25.63
`Art`	30.34	**30.44**	30.26	29.33	29.15	29.45	29.35	29.30	26.43
`Dolls`	28.80	**28.83**	28.70	27.83	27.59	27.95	27.90	27.81	26.60
`Moebius`	32.16	32.16	**32.34**	30.87	30.69	30.99	30.91	30.87	29.41
`Reindeer`	29.08	29.11	**29.25**	27.89	27.68	30.00	27.93	27.88	25.01
`Motorcycle`	27.79	27.78	**28.08**	26.82	26.71	26.93	26.81	26.78	23.86
`Vintage`	30.20	30.19	**30.36**	29.63	29.53	29.76	29.62	29.58	28.62
`Akko`	30.76	30.73	**30.92**	28.77	28.54	28.89	28.84	28.77	27.23
`Balloon`	26.13	26.14	**26.43**	23.95	23.57	24.56	24.62	24.50	22.87

(b) QP = 16

	z-dimensional DIBR, Depth Layering					BP	MUS	DIO	VSRS+
	AGFT	AGFT+	LARK	Cubic+TGV	Bilinear				
`Teddy`	31.16	31.11	**31.26**	30.76	30.68	30.86	30.71	30.73	29.87
`Laundry`	27.48	27.46	**27.60**	26.80	26.62	26.85	26.73	26.79	25.58
`Art`	29.95	29.94	**30.00**	28.93	28.78	28.80	28.66	28.70	26.32
`Dolls`	28.57	**28.57**	28.51	27.85	27.68	27.92	27.80	27.86	26.50
`Moebius`	31.49	31.50	**31.65**	30.31	30.18	30.40	30.27	30.30	29.18
`Reindeer`	28.62	28.64	**28.77**	27.74	27.59	27.81	27.69	27.74	25.04
`Motorcycle`	27.33	27.31	**27.46**	26.48	26.41	26.57	26.42	26.44	23.72
`Vintage`	29.53	29.53	**29.67**	29.21	29.15	29.29	29.13	29.16	28.30
`Akko`	29.69	29.68	**29.98**	27.88	27.74	27.96	27.82	27.86	26.20
`Balloon`	25.71	25.69	**26.02**	23.44	23.09	24.02	23.96	24.06	22.54

3) Overall Numerical and Visual Comparison: Instead of PSNR, in addition we use two other image quality metrics to evaluate the performance of our *z*-dimensional image synthesis: SSIM [19], the most commonly used image visual quality assessment metric, and 3DSwIM [20], a recently proposed metric dedicated to artifacts detection in 3D synthesized views. The numerical results are shown in Table V and VI.

Next, we examine the constructed image quality visually. For `Art` sequence, we first see in Fig. 15(a) that *z*-dimensional pixel mapping caused the erroneous mixing of foreground / background pixels during DIBR. Applying inpainting algorithm naïvely to fill in all missing pixels subsequently do not lead to acceptable quality in the expansion hole areas. Second, we see in Fig. 15(b) and Fig. 15(c) that even only using the pixels in the same depth layer for interpolation, `Bilinear` and `Cubic+TGV` will introduce significant interpolation artifacts, especially on the texture edges. Finally, we see in Fig.15(d) that by using our proposed `AGFT+`, which is comparable to result produced by `LARK` in Fig.15(e), expansion holes can be filled in a visually pleasing manner. Similar results are shown in Fig. 16, Fig. 17 and Fig. 18.

Finally, we filled in disocclusion holes using an existing scheme [7] to get the complete *z*-dimensional DIBR-synthesized images. As we can see in Fig. 19 and Fig. 20, visually pleasing images can be successfully synthesized by our proposal combined with an appropriate disocclusion hole inpainting algorithm like [7].

VIII. Conclusion

Unlike typical free viewpoint system that considers only synthesis of virtual views shifted horizontally along the *x*-dimension via depth-image-based rendering (DIBR), in this paper we consider in addition construction of *z*-dimensional DIBR-synthesized images. In such far-to-near viewpoint synthesis, there exists a new type of missing pixels called expansion holes—where objects close to the camera will increase in size and simple pixel-to-pixel mapping in DIBR from reference to virtual view will result in missing pixel areas—that demand a new interpolation scheme. We propose to first identify expansion holes via a depth layering procedure, then formulate a maximum a posteriori (MAP) problem to estimate the missing pixels using a graph-signal smoothness prior. We propose an iterative reweighted least square (IRLS) algorithm to solve the posed MAP prob-

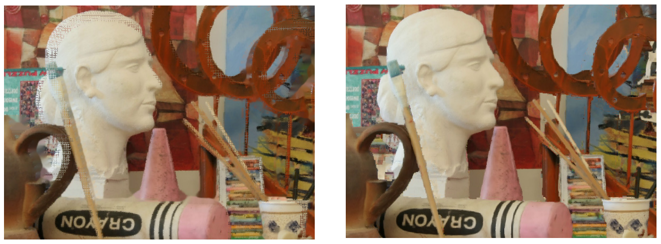

(b) `VSRS+` (a) Proposed

Fig. 19: Final output images for *z*-movement DIBR for `art`

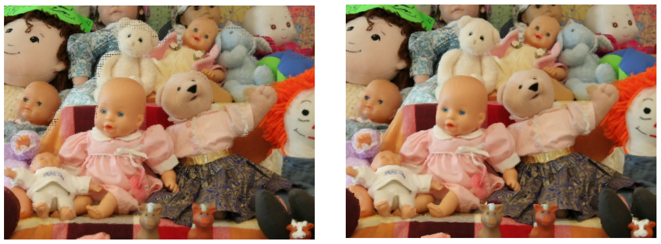

(b) `VSRS+` (a) Proposed

Fig. 20: Final output images for *z*-movement DIBR for `dolls`

TABLE V: SSIM comparison for synthesized images, reference view compression QP = 4

	z-dimensional DIBR, Depth Layering					BP	MUS	DIO	VSRS+
	AGFT	AGFT+	LARK	Cubic+TGV	Bilinear				
Teddy	0.9206	0.9202	0.9206	0.9212	0.9202	**0.9216**	0.9193	0.9186	0.9077
Laundry	0.9393	**0.9395**	0.9342	0.9346	0.9312	0.9351	0.9323	0.9314	0.8971
Art	0.9426	**0.9423**	0.9386	0.9367	0.9369	0.9374	0.9338	0.9259	0.8885
Dolls	0.9231	**0.9231**	0.9180	0.9174	0.9163	0.9192	0.9183	0.9080	0.9031
Moebius	0.9239	0.9236	**0.9242**	0.9210	0.9184	0.9167	0.9085	0.9054	0.8912
Reindeer	0.9024	**0.9026**	0.9012	0.9002	0.8999	0.9021	0.8967	0.8928	0.8775
Motorcycle	0.9343	**0.9341**	0.9279	0.9281	0.9279	0.9269	0.9257	0.9244	0.8680
Vintage	0.9490	**0.9490**	0.9472	0.9460	0.9458	0.9446	0.9430	0.9409	0.9366
Akko	0.9494	**0.9492**	0.9460	0.9453	0.9479	0.9387	0.9384	0.9367	0.8932
Balloon	0.9076	**0.9079**	0.9061	0.9048	0.9023	0.9045	0.9012	0.9006	0.8727

TABLE VI: 3DSwIM comparison for synthesized images, reference view compression QP = 4

	z-dimensional DIBR, Depth Layering					BP	MUS	DIO	VSRS+
	AGFT	AGFT+	LARK	Cubic+TGV	Bilinear				
Teddy	0.8459	**0.8462**	0.8453	0.8392	0.8384	0.8461	0.8398	0.8339	0.8425
Laundry	0.9137	**0.9132**	0.8992	0.9060	0.9062	0.9136	0.9123	0.9042	0.9101
Art	0.9741	**0.9744**	0.9621	0.9684	0.9676	0.9691	0.9672	0.9651	0.9573
Dolls	0.9500	0.9499	0.9479	0.9482	0.9499	**0.9512**	0.9476	0.9468	0.9459
Moebius	0.8938	**0.8936**	0.8925	0.8873	0.8732	0.8839	0.8724	0.8715	0.8514
Reindeer	0.9512	0.9506	0.9524	0.9545	**0.9590**	0.9524	0.9533	0.9497	0.9445
Motorcycle	0.9763	**0.9754**	0.9698	0.9679	0.9678	0.9694	0.9672	0.9613	0.9370
Vintage	0.7595	**0.7597**	0.7490	0.7496	0.7494	0.7312	0.7325	0.7322	0.7461
Akko	0.8623	**0.8619**	0.8614	0.8609	0.8602	0.8617	0.8615	0.8606	0.8329
Balloon	0.7453	**0.7453**	0.7412	0.7386	0.7367	0.7495	0.7351	0.7312	0.7376

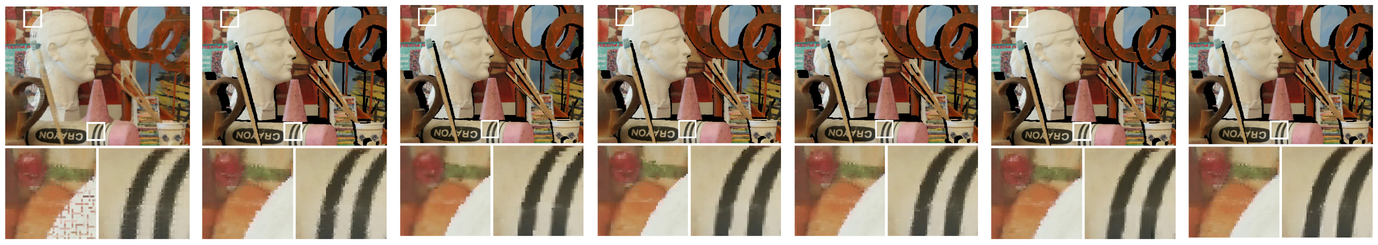

(a) VSRS+ (b) Bilinear (c) DIOMEDES (d) MUSCADE (e) BP (f) LARK (g) AGFT+

Fig. 15: Visual evaluation of synthesized images for Art with QP = 4

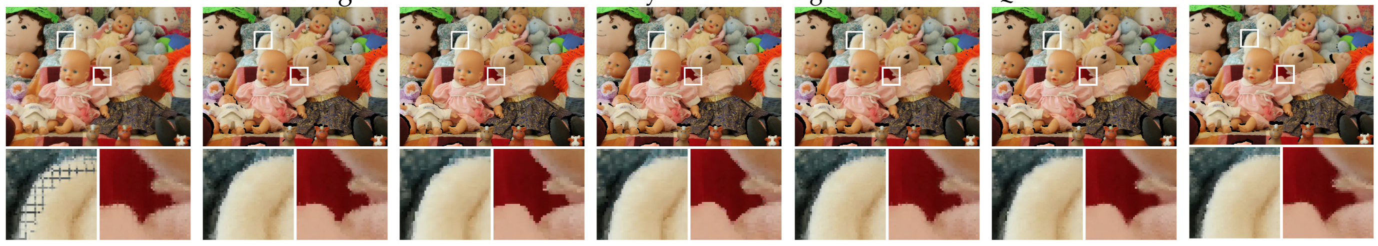

(a) VSRS+ (b) Bilinear (c) DIOMEDES (d) MUSCADE (e) BP (f) LARK (g) AGFT+

Fig. 16: Visual evaluation of synthesized images for Dolls with QP =4

lem efficiently. Experimental results show up to 4.01dB gain in PSNR over inpainting method employed in VSRS 3.5. While we focus on static multiview image rendering, our work can be extended to multiview video rendering, where additional requirements such as temporal consistency [22, 56] need to be considered also.

References

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

[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] K. Muller, H. Schwarz, D. Marpe, C. Bartnik, and et al., "3d high-efficiency video coding for multi-view video and depth data," in *IEEE Transactions on Image Processing*, vol. 22, no.9, September 2013.

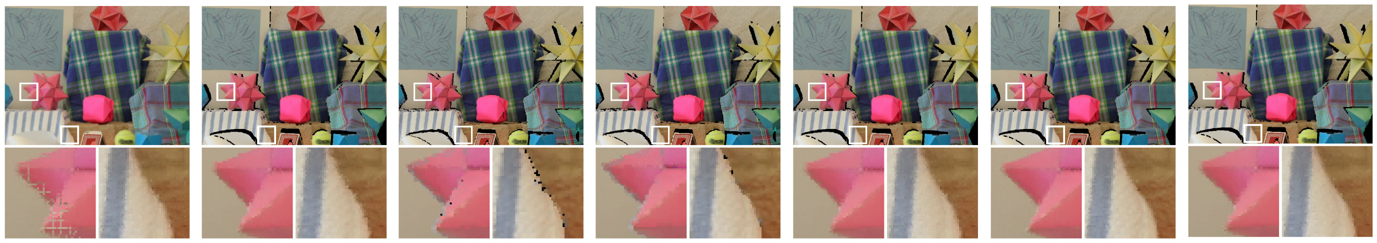
[4] D. Tian, P.-L. Lai, P. Lopez, and C. Gomila, "View synthesis techniques for 3D video," in *Applications of Digital Image Processing XXXII, Proceedings of the SPIE*, vol. 7443 (2009), 2009, pp. 74 430T–74 430T–11.

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

[6] I. Ahn and C. Kim, "Depth-based disocclusion filling for virtual view synthesis," in *IEEE International Conference on Multimedia and Expo*, Melbourne, Australia, July 2012.

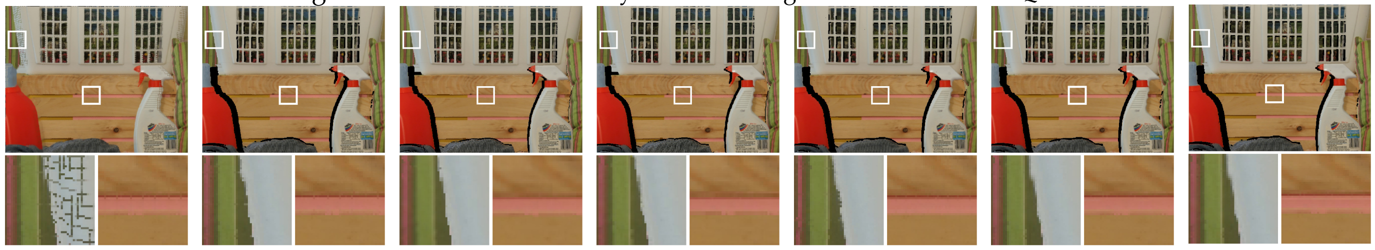
[7] S. Reel, G. Cheung, P. Wong, and L. Dooley, "Joint texture-depth pixel inpainting of disocclusion holes in virtual view synthesis," in *APSIPA ASC*, Kaohsiung, Taiwan, October 2013.

[8] C. Zhang, Z. Yin, and D. Florencio, "Improving depth perception with motion parallax and its application in teleconferencing," in



(a) `VSRS+` (b) `Bilinear` (c) `DIOMEDES` (d) `MUSCADE` (e) `BP` (f) `LARK` (g) `AGFT+`

Fig. 17: Visual evaluation of synthesized images for `Moebius` with QP = 4



(a) `VSRS+` (b) `Bilinear` (c) `DIOMEDES` (d) `MUSCADE` (e) `BP` (f) `LARK` (g) `AGFT+`

Fig. 18: Visual evaluation of synthesized images for `Laundry` with QP = 4

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] G. Cheung, A. Ortega, and N.-M. Cheung, "Interactive streaming of stored multiview video using redundant frame structures," in *IEEE Transactions on Image Processing*, vol. 20, no.3, March 2011, pp. 744–761.

[11] X. Xiu, G. Cheung, and J. Liang, "Delay-cognizant interactive multiview video with free viewpoint synthesis," in *IEEE Transactions on Multimedia*, vol. 14, no.4, August 2012, pp. 1109–1126.

[12] B. Macchiavello, C. Dorea, E. M. Hung, G. Cheung, and W. Tan, "Loss-resilient coding of texture and depth for free-viewpoint video conferencin," in *IEEE Transactions on Multimedia*, vol. 16, no.3, April 2014, pp. 711–725.

[13] L. Toni, T. Maugey, and P. Frossard, "Optimized packet scheduling in multiview video navigation systems," in *IEEE Transactions on Multimedia*, vol. 17, no.8, September 2015, pp. 1604–1616.

[14] D. Ren, G. Chan, G. Cheung, and P. Frossard, "Coding structure and replication optimization for interactive multiview video streaming," in *IEEE Transactions on Multimedia*, vol. 16, no.7, November 2014, pp. 1874–1887.

[15] D. Ren, G. Chan, G. Cheung, V. Zhao, and P. Frossard, "Anchor view allocation for collaborative free viewpoint video streaming," in *IEEE Transactions on Multimedia*, vol. 17, no.3, March 2015, pp. 307–322.

[16] Y. Mao, G. Cheung, A. Ortega, and Y. Ji, "Expansion hole filling in depth-image-based rendering using graph-based interpolation," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancouver, Canada, May 2013.

[17] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," in *IEEE Signal Processing Magazine*, vol. 30, no.3, May 2013, pp. 83–98.

[18] I. Daubechies, R. Devore, M. Fornasier, and S. Gunturk, "Iteratively re-weighted least squares minimization for sparse recovery," in *Commun. Pure Appl. Math*, 2009.

[19] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: From error visibility to structural similarity," in *IEEE Transactions on Image Processing*, vol. 13, no.4, August 2005, pp. 600–612.

[20] F. Battisti, E. Bosc, M. Carli, P. L. Callet, and S. Perugia, "Objective image quality assessment of 3D synthesized views," in *Signal Processing: Image Communication*, vol. 30, January 2015, pp. 78–88.

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

[22] 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.

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

[24] H. K. Arachchi, S. Dogan, X. Shi, E. Ekmekcioglu, S. Worrall, A. Franck, C. Hartmann, T. Korn, and P. Kovacs, "Distribution of multi-view entertainment using content aware delivery systems diomedes," European Commission, Tech. Rep. FP7-ICT-247996, Octobor 2011.

[25] P. T. Kovacs, A. Barsi, A. Ouazan, B. Gunel, D. Doyen, U. Schreiber, D. Passoni, F. Zilly, A. Laborie, and G. Berenger, "Multimedia scalable 3d for europe," Holografika, Tech. Rep. FP7-ICT-247010, June 2011.

[26] A. Chuchvara, M. Georgiev, and A. Gotchev, "Cpu-efficient free view synthesis based on depth layering," in *3DTV-Conference 2014*, Budapest, Hungary, July 2014.

[27] D. Farin, R. Peerlings, and P. H. N. de With, "Depth-image representation employing meshes for intermediate-view rendering and coding," in *3DTV-Con*, Kos Island, Greece, May 2007.

[28] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen, "Unstructured lumigraph rendering," in *SIGGRAPH*, Los Angeles, CA, August 2001.

[29] D. Glasner, S. Bagon, and M. Irani, "Super-resolution from a single image," in *IEEE International Conference on Computer Vision*, Kyoto, Japan, September 2009.

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

[31] J. Yang, Z. Wang, Z. Lin, S. Cohen, and T. Huang, "Coupled dictionary training for image super-resolution," in *IEEE Transactions on Image Processing*, vol. 21, no.8, August 2012, pp. 3467–3478.

[32] F. K. Chung, *Spectral graph theory*. American Mathematical Society, 1997.

[33] W. Hu, X. Li, G. Cheung, and O. Au, "Depth map denoising using graph-based transform and group sparsity," in *IEEE International Workshop on Multimedia Signal Processing*, Pula, Italy, October 2013.

[34] S. Chen, A. Sandryhaila, J. M. F. Moura, and J. Kovacevic, "Signal denoising on graphs via graph filtering," in *IEEE Global Conference on Signal and Information Processing*, Austin, TX, December 2014.

[35] J. Pang, G. Cheung, A. Ortega, and O. Au, "Optimal graph laplacian regularization for natural image denoising," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Brisbane, Australia, April 2015.

[36] S. K. Narang, A. Gadde, and A. Ortega, "Signal processing techniques for interpolation of graph structured data," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancouver, Canada, May 2013.

[37] X. Liu, D. Zhao, R. Xiong, S. Ma, W. Gao, and H. Sun, "Image interpolation via regularized local linear regression," in *IEEE Transactions on Image Processing*, vol. 20, no.12, December 2011.

[38] P. Wan, G. Cheung, D. Florencio, C. Zhang, and O. Au, "Image bit-depth enhancement via maximum-a-posteriori estimation of graph ac component," in *IEEE International Conference on Image Processing*, Paris, France, October 2014.

[39] X. Liu, G. Cheung, X. Wu, and D. Zhao, "Inter-block soft decoding of JPEG images with sparsity and graph-signal smoothness priors," in *IEEE International Conference on Image Processing*, Quebec City, Canada, September 2015.

[40] W. Hu, G. Cheung, A. Ortega, and O. Au, "Multi-resolution graph fourier transform for compression of piecewise smooth images," in *IEEE Transactions on Image Processing*, vol. 24, no.1, January 2015, pp. 419–433.

[41] U. Luxburg, "A tutorial on spectral clustering," in *Statistics and Computing*, vol. 17, no.4, December 2007, pp. 395–416.

[42] D. Scharstein and C. Pal, "Learning conditional random fields for stereo," in *IEEE Conference on Computer Vision and Pattern Recognition*, Minneapolis, MN, June 2007.

[43] A. Vetro and T. W. G. J. Sullivan, "Overview of the stereo and multiview video coding extensions of the H.264 / MPEG-4 AVC standard," in *Proceedings of the IEEE*, vol. 99, no.4, 2011, pp. 626–642.

[44] R. Hartley, "Theory and practice of projective rectification," in *International Journal of Computer Vision*, vol. 35, no.2, November 1999, pp. 115–127.

[45] H. Takeda, S. Farsiu, and P. Milanfar, "Kernel regression for image processing and reconstruction," in *IEEE Trans. on Image Processing*, vol. 16, no.3, Feb. 2007, pp. 349–366.

[46] J. Bigun and G. Granlund, "Optimal orientation detection of linear symmetry," in *Proceedings of the IEEE International Conference on Computer Vision*, london, UK, 1987.

[47] A. Buades, B. Coll, and J. Morel, "A non-local algorithm for image denoising," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, San Diego, CA, June 2005.

[48] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proceedings of the IEEE International Conference on Computer Vision*, Bombay, India, 1998.

[49] Y. Mao, G. Cheung, and Y. Ji, "Image interpolation during dibr view synthesis using graph fourier transform," in *3DTV-Conference 2014*, Budapest, Hungary, July 2014.

[50] 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*, vol. 13, no.7, July 2003, pp. 560–576.

[51] H. Hou and H. Andrews., "Cubic splines for image interpolation and digital filtering," in *IEEE Transactions on Acoustic, Speech, and Signal Processing*, vol. 26, no.6, January 1978, pp. 508–517.

[52] K. Bredies and M. Holler, "A TGV-based framework for variational image decompression, zooming and reconstruction. part i: Analytics," in *SIAM Journal on Imaging Sciences*, vol. 8, no.4, 2015, pp. 2814–2850.

[53] Y. Romano, M. Protter, and M. Elad, "Single image interpolation via adaptive nonlocal sparsity-based modeling," in *IEEE Transactions on Image Processing*, vol. 23, no.7, July 2014, pp. 3085–3098.

[54] X. G. K. Zhang, D. Tao, and X. Li, "Single image super-resolution with non-local means and steering kernel regression," in *IEEE Transactions on Image Processing*, vol. 21, no.11, November 2012, pp. 4544–4556.

[55] G. Yu, G. Sapiro, and S. Mallat, "Solving inverse problems with piecewise linear estimators: From gaussian mixture models to structured sparsity," in *IEEE Transactions on Image Processing*, vol. 21, no.5, May 2012, pp. 2481–2499.

[56] Z. Liu, G. Cheung, J. Chakareski, and Y. Ji, "Multiple description coding and recovery of free viewpoint video for wireless multi-path streaming," in *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no.1, 2015, pp. 151–164.

Yu Mao received the B.S. degree in Computer Science from the University of Science and Technology of China, Hefei, China, in 2011. He is currently pursuing the Ph.D. degree in Informatics from The Graduate University of Advanced Studies, Tokyo, Japan. His research interests include graph signal processing, Free-view/Multiview imaging and machine learning.

Gene Cheung (M'00—SM'07) received the B.S. degree in electrical engineering from Cornell University in 1995, and the M.S. and Ph.D. degrees in electrical engineering and computer science from the University of California, Berkeley, in 1998 and 2000, respectively.

He was a senior researcher in Hewlett-Packard Laboratories Japan, Tokyo, from 2000 till 2009. He is now an associate professor in National Institute of Informatics in Tokyo, Japan. He has been an adjunct associate professor in the Hong Kong University of Science & Technology (HKUST) since 2015.

His research interests include 3D image processing, graph signal processing, and signal processing for sleep analysis. He has served as associate editor for IEEE Transactions on Multimedia (2007–2011), DSP Applications Column in IEEE Signal Processing Magazine (2010–2014) and SPIE Journal of Electronic Imaging (2014–2016). He currently serves as associate editor for IEEE Transactions on Image Processing (2015–present), IEEE Transactions on Circuits and Systems for Video Technology (2016–present) and APSIPA Journal on Signal & Information Processing (2011–present), and as area editor for EURASIP Signal Processing: Image Communication (2011–present). He is a distinguished lecturer in APSIPA (2016–2017). He served as a member of the Multimedia Signal Processing Technical Committee (MMSP-TC) in IEEE Signal Processing Society (2012–2014), and a member of the Image, Video, and Multidimensional Signal Processing Technical Committee (IVMSP-TC) (2015–2017). He is a co-author of the best student paper award in IEEE Workshop on Streaming and Media Communications 2011 (in conjunction with ICME 2011), best paper finalists in ICME 2011, ICIP 2011 and ICME 2015, best paper runner-up award in ICME 2012 and best student paper award in ICIP 2013.

Yusheng Ji received B.E., M.E., and D.E. degrees in electrical engineering from the University of Tokyo. She joined the National Center for Science Information Systems, Japan (NACSIS) in 1990. Currently, she is a Professor at the National Institute of Informatics, Japan (NII), and the Graduate University for Advanced Studies (SOKENDAI). Her research interests include network architecture, resource management, and quality of service provisioning in wired and wireless communication networks. She has served as a Board Member of Trustees of IEICE, Steering Committee Member of Quality Aware Internet (QAI) SIG and Internet and Operation Technologies (IOT) SIG of IPSJ, Associate Editor of IEICE Transactions and IPSJ Journal, Guest Editor-in-Chief, Guest Editor, and Guest Associate Editor of Special Sections of IEICE Transactions, Guest Associate Editor of Special Issues of IPSJ Journal, etc. She has also served as a TPC member of many conferences, including IEEE INFOCOM, ICC, GLOBECOM, VTC etc., and Symposium Co-chair of IEEE GLOBECOM 2012, 2014. She is an Editor of IEEE Transactions of Vehicular Technology, Track Co-chair of IEEE VTC 2016 Fall, and an Expert Member of IEICE Technical Committees on Internet Architecture, and Communication Quality.