Loss-resilient Coding of Texture and Depth for Free-viewpoint Video Conferencing

Bruno Macchiavello Camilo Dorea Edson M. Hung Gene Cheung *Senior Member, IEEE* Wai-tian Tan *Senior Member, IEEE*

Abstract

Free-viewpoint video conferencing allows a participant to observe the remote 3D scene from any freely chosen viewpoint. An intermediate virtual viewpoint image is typically synthesized using two pairs of transmitted texture and depth maps from two neighboring captured viewpoints via depth-image-based rendering (DIBR). To maintain high quality of synthesized images, it is imperative to contain the adverse effects of network packet losses that may arise during texture and depth video transmission. Towards this goal, we develop an integrated approach that exploits the representation redundancy inherent in the multiple streamed videos—a voxel in the 3D scene visible to two captured views is sampled and coded twice in the two views. In particular, at the receiver we first develop an error concealment strategy that adaptively blends corresponding pixels in the two captured views during DIBR, so that pixels from the more reliable transmitted view are weighted more heavily. We then couple it with a sender-side optimization of reference picture selection (RPS) during real-time video coding, so that blocks containing pixel samples of voxels that are visible in both views are more error-resiliently coded in one view only, given adaptive blending will mitigate errors in the other view. Further, synthesized view distortion sensitivities to texture versus depth errors are analyzed, so that relative importance of texture and depth code blocks can be computed for system-wide RPS optimization. Finally, quantization parameter (QP) is adaptively selected per frame, optimally trading off source distortion due to compression with channel distortion due to potential packet losses. Experimental results show that the proposed scheme can outperform previous work by up to 2.9dB at 5% packet loss rate.

Bruno Machiavello, Camilo Dorea, Edson M. Hung are with the University of Brasilia, Asa Norte, CEP: 70910-900, Brasilia, Brazil. Email Address: {bruno, camilo, mintsu}@image.unb.br.

Gene Cheung is with the National Institute of Informatics, 2-1-2, Hitotsubashi, Chiyoda-ku, Tokyo, 101–8430, Japan. Email Address: cheung@nii.ac.jp.

Wai-tian Tan is with the Enterprise Networking Group in Cisco Systems Inc, Milpitas, CA 95035, USA. Part of this work was performed while at Hewlett-Packard Laboratories, Palo Alto, CA, USA. Email Address: dtan2@cisco.com.

This work was partly supported by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) under grants 476176/2013-1 and 310375/2011-8. This work was also partly supported by JSPS KAKENHI Grant Number 23700136.

I. Introduction

The demand for ever improving quality of video communication has already made high definition video conferencing a basic feature not only on computers, but smart-phones as well. A free-viewpoint video conferencing system [1] that continuously alters the displayed images according to a user's real-time selected viewpoints—e.g., through *motion parallax* [2], where detected head movements trigger corresponding rendering of virtual images as viewed from the observer's viewpoint—can enhance an observer's depth perception in the 3D scene, and bring video communication to a new level of immersion and realism. The techniques for capturing [3], representing [4], [5], and synthesizing [6], [7] free-viewpoint video have been well studied. Instead, in this paper we address the problem of robust low-latency streaming of free-viewpoint video over packet-loss prone networks.

We adopt the widely employed depth-image-based rendering (DIBR) approach [7] to synthesize free viewpoint images. As the name suggests, in addition to typical RGB images (texture), DIBR requires also depth images (per-pixel physical distances between captured objects in the 3D scene and capturing camera) to synthesize a freely chosen viewpoint image. Depth maps can be obtained by estimation algorithms like stereo-matching, or depth sensors like time-of-flight cameras [8]. While recent proposals such as [9], [10] suggest transmission of one pair of texture and depth maps from a single camera-captured viewpoint for synthesis of a defined neighborhood of viewpoints, this usually leads to larger disoccluded regions[1] in the synthesized view that require image inpainting [11], [12] to complete the image, resulting in high complexity. We hence assume the more customary approach of transmitting texture and depth maps of two neighboring captured views [2], [13] to ensure quality reconstruction of freely chosen intermediate virtual views.

The transport of texture and depth videos for multiple views presents both challenges and opportunities. On one hand, the complex dependency of rendered view quality on both depth and texture maps of captured views needs to be adequately modeled to realize effective error-resilience measures. On the other hand, the inherent representation redundancy in multiple texture-plus-depth videos—*a voxel*[2] *in the 3D scene visible to two captured views is sampled and coded twice in the two views*—suggests that errors in one view can be ameliorated by correct delivery of another view. Towards this end, we develop an integrated approach in this paper where receiver and sender are jointly designed to exploit and manipulate the inherent representation redundancy present in depth and texture videos from multiple views.

To exploit representation redundancy at the receiver, we first develop an error model that tracks

[1]Disoccluded region is a spatial region in the synthesized view with no corresponding pixels in the reference views during DIBR. This issue is discussed in details in Section IV.

[2]A voxel is a volume element representing a value on a regular grid in 3D space [14].

the reliability of each code block in each transmitted view given observed packet loss events. During DIBR-based rendering of a virtual view, where a synthesized pixel is typically computed as a convex combination of the two corresponding pixels in the two captured views, we perform *adaptive blending*, so that the corresponding pixel from the more reliable transmitted view is assigned a heavier weight.

To exploit representation redundancy at the sender, we optimize reference picture selection (RPS) during real-time video coding, so that blocks containing pixel samples of voxels that are visible in both views are more error-resiliently coded (e.g., through intra-coding or using a reference block in a previous frame that has been acknowledged) in one view only. This is possible because aforementioned adaptive blending is performed at decoder, suppressing errors in one view if corresponding blocks in the other view are correctly delivered. The expected reconstructed error of a block predicted from a given past frame is computed via a set of recursive equations we derive with computation efficiency in mind.

Further, because depth maps are auxiliary information that only aid in the construction of the synthesized views but are not themselves directly observed, the synthesized view distortion sensitivities to errors in depth maps are clearly different from errors in texture maps. To guide preferential protection, we analyze synthesized view distortion sensitivities to texture versus depth errors, so that relative importance of texture and depth code blocks can be computed for system-wide RPS optimization. Finally, quantization parameter (QP) is adaptively selected per frame at encoder based on observed packet loss events to-date, so that source distortion due to compression can be optimally traded off with channel distortion due to potential packet losses for best overall performance. Experimental results show that the proposed scheme can outperform previous work by up to 2.9dB at 5% packet loss rate.

The rest of the paper is organized as follows. We first discuss related work in Section II. We then overview our streaming system in Section III. We discuss our receiver error concealment strategy using adaptive blending in Section IV. We discuss our sender RPS optimization in Section V. In Section VI, an analysis of the added computational load of our proposed optimizations is presented. Finally, experimental results and conclusions are presented in Section VII and VIII, respectively.

II. Related Work

We divide the discussion of related work into three sections. We first discuss related work in multiview and free-viewpoint video coding. We then discuss related work in error-resilient video streaming and error concealment for conventional 2D video. Finally, we juxtapose the contributions of this paper to our own earlier work on the same topic.

A. Multiview and Free Viewpoint Video Coding

Multiview video coding (MVC) is concerned with the compression of texture videos captured from multiple nearby viewpoints. Early works in MVC [4], [15] focused on exploiting signal redundancy across view using *disparity compensation* for coding gain—matching of code blocks between neighboring view pictures for efficient signal prediction. However, given temporal redundancy has already been exploited via motion compensation, and neighboring temporal frames tend to be more similar than neighboring inter-view frames due to typical high-frame-rate videos captured by cameras, it was shown that additional coding gain afforded by disparity compensation is noticeable but not dramatic [15]. Given our goal is loss-resilient video streaming, to avoid potential inter-view error propagation in disparity compensated multiview video, we perform motion estimation / compensation independently in each view[3].

While MVC offers view-switches at receiver only among the discrete set of captured and coded views, transmitting both texture and depth videos of nearby views—known as "texture-plus-depth" format [5]—enables the observer to choose any intermediate virtual view for DIBR-based image rendering and display. Given compression of texture maps has been well studied in the past decades (e.g., coding standards such as H.264 [17]), recent work has focused specifically on depth map compression. These can be divided into three classes. The first class [18]–[20] designed specific coding tools (e.g., graph-based transform (GBT) [18], [20]) tailored to the unique signal characteristics of depth maps, such as sharp edges and smooth interior surfaces. In our streaming system, for simplicity we assume H.264 is employed for standard-compliant compression of texture and depth videos in each captured view. However, we note that new coding tools such as GBT can be incorporated into our streaming optimization easily.

The second class [21]–[25] of works recognized that a depth map is a source of auxiliary information that assists in the virtual view synthesis at decoder, but is itself never directly observed. Thus, one can design coding optimizations that minimize the indirect synthesized view distortion rather than the direct depth map distortion. Our analysis of synthesized view distortion sensitivities to errors in texture and depth maps is an extension of [24] from coding optimization to loss-resilient streaming optimization.

The third class [26], [27] exploited correlations between texture and depth maps of the same view (such as common edge patterns) for compression gain. In our previous study [28], we have concluded that an important depth block (e.g., one that contains an edge between a foreground object and background) can be more critical to synthesized view quality than a texture block. Predicting depth blocks from corresponding texture blocks (as done in [26] as an extension of HEVC video coding standard to texture-

[3]We note further that having independent encoders for different views fits the *multiterminal video coding* paradigm [16], which, besides benefits of lower overall encoding complexity, holds promises of better compression performance in the future via distributed source coding theory.

plus-depth format) would entail unnatural dependency of a more important depth block on a less important texture block. Given depth video typically requires roughly 10% of the total bitrate and the goal is loss resiliency of free viewpoint video streaming, we choose to forego inter-component prediction and code texture and depth videos of the same view independently using H.264.

B. Error-resilient Video Streaming

The problem of error control and concealment has been actively studied for many years, and an overview of general approaches can be found in [29]. One particular effective mechanism to control error propagation for systems with live encoder is reference picture selection (RPS). In *reactive* RPS, a live encoder reacts to receiver feedbacks by avoiding the use of notified loss-affected past frames as reference for coding of future frames. In *proactive* RPS, long prediction chains are avoided during video encoding without incorporating real-time client feedback information. Reactive RPS incurs higher bit overhead only when needed, but suffers error propagation of up to one round-trip time (RTT) at decoder. Proactive RPS incurs an overhead regardless of whether there are actually losses, and should only be applied preferentially to more important parts of the video. A study of adaptively choosing different RPS approaches in streaming of single-view H.264 video is given in [30]. In contrast, we employ at sender proactive RPS with feedbacks [31] (called *proactive-feedback* in the sequel) for selected code blocks in the depth and texture sequences during real-time video encoding, so that the likelihood of long error propagation—typical in differentially coded video—is minimized for blocks containing pixels vital to adaptive rendering at receiver. Our proactive-feedback approach integrates feedback by computing estimated distortion for all frames with and without known loss information.

There are many other techniques developed for improving resilience of single-view video, including error concealment, retransmissions and use of error correcting codes. In addition to commonly employed Reed-Solomon codes, there are also recent low-delay "streaming codes" that allows fast recovery of earlier transmitted packets without requiring all losses to be corrected [32]. Modern video compression formats also have syntax support for error resilience. For example, *flexible macroblock ordering* (FMO) [33] is an error resilient tool in H.264 that can avoid loss of large contiguous regions to make error concealment more effective. Another example is data partitioning [34], which allows decomposition of compressed single-view video into layers of different importance for preferential protection. Furthermore, it is possible to explicitly transmit additional data in various forms to aid error concealment [35]. Many of these techniques are applicable to free-viewpoint conferencing and are complementary to multi-view error concealment methods as well as to adaptive error-resilient source coding methods.

Joint optimization of compression, concealment, and channel loss has also been considered. For example, the work in [36], considers a formulation that allocates transmission power to jointly optimize the average and variance of distortion for single view video. Finally, though there have been streaming optimizations proposed in the literature for stereoscopic video [37] and multiview video [38], to the best of our knowledge we are the first to propose loss-resilient coding specifically for interactive free-viewpoint video where texture and depth maps of two neighboring camera viewpoints are transmitted [39].

C. Contribution over Our Previous Work

In [40], a similar scheme to minimize expected synthesized view distortion based on selection of reference frame at the block level was proposed for depth maps only. In another previous work [28], we extended the idea in [40] to encoding of both texture *and* depth maps.

In this work, we make significant contributions over our earlier works. First, given that an intermediate virtual view pixel is computed as a convex combination of corresponding texture pixels of its two neighboring captured views, an error concealment strategy can be performed where the source pixel with higher reliability is weighted more heavily during blending. Further, for pixels of the two views that sample the same voxels in the 3D scene, the encoder can protect pixels from only one view more heavily. Finally, we optimize the selection of QP during source coding at sender, so that the overall expected distortion due to source compression and channel losses is minimized.

III. Real-time Free Viewpoint Video Streaming System

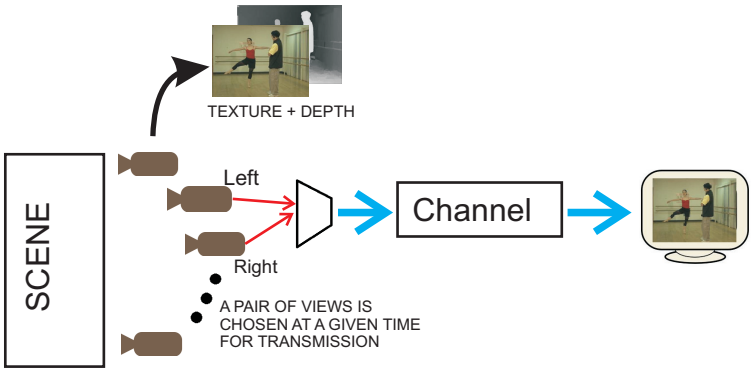


Fig. 1. A bandwidth-efficient free-viewpoint video streaming system dynamically selects two views for transmission.

We overview our proposed real-time free viewpoint video streaming system in this section. At sender, we assume a large array of closely spaced cameras are capturing videos of a 3D scene simultaneously. See Fig. 1 for an illustration. Specifically, cameras are deployed in a 1D parallel arrangement with narrow acquisition angles and a baseline between cameras of the order of 5 cm [41]. Rectification, performed

prior to encoding, is often necessary to eliminate misalignments and provide views in linear and parallel arrangement. These conditions guarantee that disparities between views are limited to 1D shifts along the x-axis. Naturally, accuracy in camera calibration parameters is also required. Furthermore, color consistency among cameras should be assured upon capturing and/or enforced through image processing methods [42] prior to encoding.

Besides texture maps, we assume depth maps of the same resolution and from the same camera viewpoints as the texture maps are available. Armed with both texture and depth maps, intermediate virtual views can be synthesized via a depth-image-based rendering (DIBR) technique like 3D warping [43], enabling user to select and render image at any desired viewpoint for observation. The ability to choose any viewpoint for image rendering is called *free viewpoint* [42].

The number of cameras capturing video at the same time can be quite large—up to 100 cameras were used in [3]. Hence real-time encoding and transmitting all of them from sender to receiver would translate to a very large network cost. Instead, we assume here that the index of the current virtual view v that the receiver is observing is constantly fed back to the sender. The sender can then estimate the range of virtual views the receiver will choose to observe during the next round-trip time (RTT), and then select only *two* neighboring camera views that can enable rendering of those virtual views for real-time coding and transmission of the corresponding texture and depth maps. This idea of selecting only a subset of camera captured views for efficient network transmission is also exploited in [13]. In this paper, we focus on the optimal transmission and error concealment of the texture and depth maps from those two selected views (called left and right views in the sequel).

IV. Error Concealment Strategy for DIBR-based Synthesis

We first review a common DIBR view synthesis procedure that assumes no information loss in either depth or texture maps. We then derive formulas to estimate texture and disparity error for every macroblock (MB) due to packet losses in motion-compensated video. Finally, we discuss how an error concealment strategy for the synthesized view—*adaptive blending*—can be performed given the estimated texture and disparity errors.

A. Standard Loss-free View Synthesis

In a multiview DIBR system, an intermediate virtual view is interpolated using texture and depth maps of two neighboring captured views via a DIBR technique like 3D warping [43]. We focus on the fundamental view blending step upon which our proposal is based. Note that prior to blending,

mechanisms aimed at robustness, such as consistency checks, may first pre-process respective texture or depth maps. We refer readers to [7] for a complete description of view synthesis procedures.

As a convention, we label left and right views as 0 and 1, respectively. Given the texture maps of the left and right views, X_t^0 and X_t^1, at time t, the pixel value $S_t^v(i,j)$ at coordinate (i,j) of a synthesized view v depends on whether we can find corresponding pixels (i,j^0) and (i,j^1) from the left and right views, respectively. Specifically, the synthesized image of view v, $0 \le v \le 1$, has pixel value $S_t^v(i,j)$ at coordinate (i,j) given by:

$$S_t^v(i,j) = \begin{cases} (1-v)\ X_t^0(i,j^0) \ + \ v\ X_t^1(i,j^1) & \text{if left \& right corresponding pixels exist} \\ X_t^0(i,j^0) & \text{if only left corresponding pixel exists} \\ X_t^1(i,j^1) & \text{if only right corresponding pixel exists} \\ \text{hole} & \text{o.w.} \end{cases} \quad (1)$$

where the weights $1-v$ and v are inversely proportional to the virtual view's distance to view 0 and 1. A chosen inpainting algorithm like [11], [12] is then applied to fill in a small number of hole pixels that have no correspondence in both views 0 and 1. In other words, weighted blending is performed if two correspondences are found, while single-pixel mapping and inpainting are invoked if one or zero correspondence is found, respectively. See [7] for details of a standard DIBR view synthesis implementation.

A corresponding pixel (i,j^0) in the left texture map X_t^0 is a pixel that, given associated disparity $Y_t^0(i,j^0)$ between left and right views subject to normalization, shifts horizontally by $Y_t^0(i,j^0) * v * \eta$ pixels to synthesized pixel of coordinate (i,j) in virtual view S_t^v, where η is a scaling factor determined by the distance between the capturing cameras. We can write j^0 and j^1 for the two corresponding pixels in left and right texture maps as:

$$\begin{aligned} j &= j^0 - Y_t^0(i,j^0) * v * \eta \\ j &= j^1 + Y_t^1(i,j^1) * (1-v) * \eta \end{aligned} \quad (2)$$

One interpretation of (1) is that the same physical point in the 3D scene (called *voxel* in computer graphics literature [44]) has observed intensities $X_t^0(i,j^0)$ and $X_t^1(i,j^1)$ from views 0 and 1, respectively. Hence, the intensity of the same voxel at an intermediate view can be modeled as a convex combination of the two corresponding values. If the surface of the observed physical object has *Lambertian reflectance* [44], then a voxel will have (roughly) the same intensity from different viewpoints; i.e., $X_t^0(i,j^0) \approx X_t^1(i,j^1)$. In such case of *redundant representation* (same intensity value is recorded in two corresponding coordinates at left and right maps), synthesized pixel $S_t^v(i,j)$ can be perfectly reconstructed even if one of the

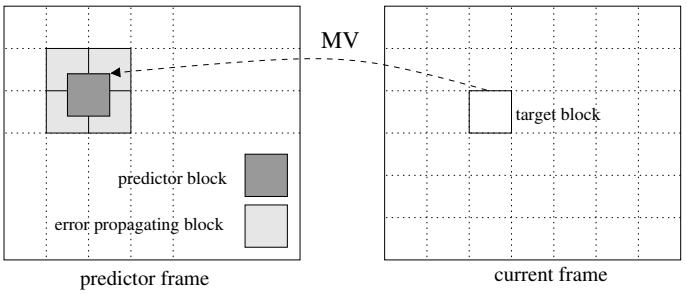


Fig. 2. Motion prediction in differentially coded video causes error propagations from predictor block to target block.

two corresponding pixels, $X_t^0(i, j^0)$ and $X_t^1(i, j^1)$, is corrupted by channel noise, assuming the decoder knows which corresponding pixel is erred. Based on this observation, and assuming that the majority of synthesized pixels are of objects with Lambertian reflectance surfaces[4], we develop the following error concealment strategy during view synthesis.

The key idea of the error concealment strategy for DIBR view synthesis is as follows. When there are two corresponding pixels $X_t^0(i, j^0)$ and $X_t^1(i, j^1)$ for a given synthesized pixel $S_t^v(i, j)$, decoder can choose to *reweigh* the combination of the pixels depending on the reliability of the two reconstructed pixels. For example, decoder can use only one corresponding pixel for synthesis if the other corresponding pixel is deemed totally unreliable. We next discuss how we estimate the reliability of the two corresponding texture pixels $X_t^0(i, j^0)$ and $X_t^1(i, j^1)$ at decoder given observed packet losses in the transmitted maps.

B. Estimating Texture Error at Decoder

We first compute the reliability of the two corresponding texture pixels $X_t^0(i, j^0)$ and $X_t^1(i, j^1)$, assuming the corresponding disparity pixels $Y_t^0(i, j^0)$ and $Y_t^1(i, j^1)$ are correct. Let $m = b(i, j^0)$ be the MB that contains texture pixel (i, j^0), and let $e_{t,m}$ be the texture error due to packet losses in MB m given differential coding of texture maps. Depending on whether MB m is received correctly or not, $e_{t,m}$ will result in error $e_{t,m}^+$ and $e_{t,m}^-$, respectively:

$$e_{t,m} = \begin{cases} e_{t,m}^+ & \text{if MB } m \text{ is correctly received} \\ e_{t,m}^- & \text{o.w.} \end{cases} \tag{3}$$

Consider first $e_{t,m}^+$. If MB m is an intra-coded block (independently coded) and is correctly received at decoder, then $e_{t,m}^+ = 0$. If MB m is an inter-coded block (differentially coded using a block of a previous frame as predictor) pointing to a previous frame $\tau_{t,m}$ with motion vector (MV) $v_{t,m}$, then $e_{t,m}^+$ depends on the errors of predictor block in frame $\tau_{t,m}$. Note that because the predictor block indicated by MV $v_{t,m}$ can be located among several coded MBs in frame $\tau_{t,m}$, there are typically multiple error propagating

[4]While objects like glass and mirrors do not have Lambertian surfaces, if the two captured viewpoints are sufficiently close, voxels of these objects can nonetheless be approximated as having the same intensity values from different but nearby viewpoints.

MBs, $k \in v_{t,m}$, that can potentially contribute to $e^+_{t,m}$. See Fig. 2 for an illustration. Thus, we write $e^+_{t,m}$ recursively as follows[5]:

$$e^+_{t,m}(\tau_{t,m}, v_{t,m}) = \begin{cases} 0 & \text{if MB } m \text{ is Intra} \\ \gamma \sum_{k \in v_{t,m}} \alpha_k \; e_{\tau_{t,m},k} & \text{o.w.} \end{cases} \quad (4)$$

where α_k's are the weights for the summation in (4) based on the amount of pixel overlaps between the designated predictor block and the error propagating MBs, and $\gamma < 1$ is the attenuation factor that reflects the dissipating effect of error in an earlier frame over a sequence of motion-compensated frames. In our simulations, γ is chosen to be 0.9.

If MB m of frame t is not correctly received, then we assume that the decoder performs a block copy from the same MB m from previous frame $t-1$. The resulting error is then error $e_{t-1,m}$ of MB m in frame $t-1$, plus δ, which is the change in intensity between MBs m in frame $t-1$ and t:

$$e^-_{t,m} = e_{t-1,m} + \delta \quad (5)$$

δ can be estimated at decoder as follows. If there is a corresponding block in texture map of the other captured view that is more reliably received, then δ is approximated as the difference between the corresponding blocks in the previous frame and the current frame of the other view. If there is no corresponding block in the other view or the corresponding block is also erred, then we estimate δ as the difference among co-located blocks of the previous two frames within the current view. If one or two previous frames are not available or not correctly received, δ is estimated from spatially neighboring blocks with similar disparity values (blocks with similar depth are likely to belong to the same physical object).

C. Estimating Disparity Error at Decoder

Similarly, we can compute the error in left disparity pixel $Y^0_t(i, j^0)$ by estimating the error in the MB $m = b(i, j^0)$ that contains it. (Right disparity pixel $Y^1_t(i, j^1)$ can be estimated in a similar manner.) We can estimate the disparity error[6] $\epsilon_{t,m}$ in MB m using recursion as done for $e_{t,m}$ in (3). In this case, derivation of disparity errors due to correctly received and lost MBs are analogous to the equations presented in (4) and (5), respectively, with the exception that δ is always estimated from co-located blocks in the previous frames.

[5]Though we have already argued that inter-view prediction is not a sensible coding option for our loss-resilient free viewpoint video streaming system, (4) can nonetheless be modified easily to account for inter-view dependencies if deployed.

[6]If a depth block is predicted from the corresponding texture block of the same view as done in [26], (4) can be easily modified accordingly to reflect the inter-component dependency.

The effect of a disparity error on the synthesized pixel (i, j), however, is more indirect; a disparity error causes the wrong corresponding pixel $X_t^0(i, j^{\#})$ to be used for synthesis of pixel (i, j) in virtual view S_t^v. Now if the pixel patch around the local neighborhood of texture pixel $X_t^0(i, j^0)$ is monotone, then using a texture pixel from a slightly erred location $(i, j^{\#})$ will result in very small increase in synthesized distortion. On the other hand, if the patch around texture pixel (i, j^o) has high texture variation, then the resulting synthesized distortion can be significant.

D. Proposed View Synthesis

Having computed the estimated texture error $e_{t,m}$ and disparity error $\epsilon_{t,m}$ for the two MBs m that contain corresponding texture pixel $X_t^0(i, j^0)$ and depth pixel $Y_t^0(i, j^0)$ of the left view respectively, as described in the past two sections, we now derive the *worst case distortion* $d^0(i, j^0)$ for left corresponding pixel $X_t^0(i, j^0)$. Then, left reliability term r^0 can be subsequently computed. Together with right reliability term r^1, they determine how the two corresponding texture pixels $X_t^0(i, j^0)$ and $X_t^1(i, j^1)$ should be reweighed for our adaptive, error-aware pixel blending.

Because disparity error leads to the selection of a wrong texture pixel for blending, given the estimated disparity error $\epsilon_{t.m}$, we will consider the possible adverse effect of using texture pixel $X_t^0(i, l)$ for synthesis instead of $X_t^0(i, j^0)$, where l in the range $[j^0 - \epsilon_{t.m}, j^0 + \epsilon_{t.m}]$. Specifically, for each texture pixel $X_t^0(i, l)$, we consider both the pixel-to-pixel texture intensity difference $|X_t^0(i, l) - X_t^0(i, j^0)|$ and the estimated texture error due to channel losses $e_{t,b(i,l)}$ for texture pixel $X_t^0(i, l)$. Mathematically, the worst-case distortion $d^0(i, j^0)$ for texture pixel $X_t^0(i, j^0)$ considering all pixels (i, l) in the range is computed as follows:

$$d^0(i, j^0) = \max_{l=j^0-\epsilon_{t,m},\ldots,j^0+\epsilon_{t,m}} \left\{ e_{t,b(i,l)} + |X_t^0(i, l) - X_t^0(i, j^0)| \right\} \tag{6}$$

As done for δ, the pixel intensity difference $|X_t^0(i, l) - X_t^0(i, j^0)|$ in (6) can be estimated at decoder using either corresponding pixels in texture map of the other captured view that is more reliably received, or MBs of previous correctly received frames of the same view with similar disparity values.

Having derived worst case distortions $d^0(i, j^0)$ and $d^1(i, j^1)$ for texture pixels, $X_t^0(i, j^0)$ and $X_t^1(i, j^1)$, we now define a reliability metric, r^0, inversely proportional to distortion:

$$r^0 = \frac{1}{d^0(i, j^0) + c} \tag{7}$$

where c is a small positive constant chosen so that r^0 is well defined even if $d^0(i, j^0) = 0$. Reliability metric r^1 is defined analogously in terms of $d^1(i, j^1)$.

Depending on reliability, corresponding pixels from left and right views are now reweighed in synthesis

of view v as

$$S_t^v(i,j) = w_t^0 \; X_t^0(i,j^0) + w_t^1 \; X_t^1(i,j^1) \tag{8}$$

where weights w_t^0 and w_t^1 are computed as

$$\begin{aligned} w_t^0 &= \frac{r^0(1-v)}{r^0(1-v)+r^1 v} \\ w_t^1 &= \frac{r^1 v}{r^0(1-v)+r^1 v}. \end{aligned} \tag{9}$$

The weights have the following properties: i) default to $(1-v)$ and v in (1) when reliability $r^0 = r^1$; ii) converge to 1 and 0 when $r^0 \gg r^1$; and iii) sum to 1.

V. Block-level Reference Picture Selection

Having described the error concealment strategy deployed during view synthesis at the receiver, we now turn our attention to optimization at the sender. At the sender side, during real-time encoding of both texture and depth maps, the encoder has the flexibility to select any MB in any past coded frame for motion compensation (MC) to encode each MB in a current frame t. Using a well matched MB from the immediate previous frame $t-1$ for MC would lead to a small encoding rate, but may result in large expected distortion at receiver due to a possibly long dependency chain of differentially coded MBs. Using a MB from a frame further into the past for MC (or an intra-coded MB), will lead to a larger encoding rate, but will also result in a smaller expected distortion. Exploiting this flexibility of *reference picture selection* (RPS), the goal is to pro-actively minimize the overall expected distortion of an intermediate view at instant t, synthesized via DIBR using texture and depth maps of two adjacent coded views at decoder as described in the previous section, and subject to a transmission rate constraint. Leveraging our derivation for estimated texture and disparity errors at the receiver in the previous section, we first discuss how synthesized distortion in an interpolated view is estimated at the sender. We then present the mathematical formulation of our proactive-feedback block-level RPS optimization.

A. Estimating Texture & Disparity Error at Sender

We first estimate the expected texture error $e_{t,m}$ of a MB m in frame t at sender in differentially coded texture video as follows. Let $e_{t,m}(\tau_{t,m}, v_{t,m})$ be the texture error given it is motion-compensated using a block identified by MV $v_{t,m}$ inside a previous transmitted frame $X_{\tau_{t,m}}$, $\tau_{t,m} < t$. Let p be the probability that MB m is correctly *received*. Similar to (3), we can write $e_{t,m}(\tau_{t,m}, v_{t,m})$ in terms of $e^+_{t,m}(\tau_{t,m}, v_{t,m})$ and $e^-_{t,m}$, the expected texture error of MB m if it is correctly received and lost, respectively. Unlike (3), there

are now two additional considerations. First, the expression is probabilistic and depends on p, because the delivery status of packet that contains MB m is not known deterministically at sender in general. A feedback channel can be used to provide packet delivery status information for transmitted packets with RTT delay. We write $e_{t,m}$ at sender as follows:

$$e_{t,m}(\tau_{t,m}, v_{t,m}) = p\ e^{+}_{t,m}(\tau_{t,m}, v_{t,m}) + (1-p)\ e^{-}_{t,m} \tag{10}$$

Second, because encoder can control the source compression ratio via a quantization parameter (QP), we consider the effects of source coding error as well. Hence unlike (4), we in addition account for source coding error $e^{s}_{t,m}$—a function of QP q_t for frame t—when writing $e^{+}_{t,m}$ mathematically as follows:

$$e^{+}_{t,m} = \begin{cases} e^{s}_{t,m}(q_t) & \text{if MB } m \text{ is Intra} \\ \gamma \sum_{k \in v_{t,m}} \alpha_k\ e^{c}_{\tau_{t,m},k} + e^{s}_{t,m}(q_t) & \text{o.w.} \end{cases} \tag{11}$$

In words, (11) states that if MB m is correctly received, then the expected error $e^{+}_{t,m}$ is: i) only the source coding error $e^{s}_{t,m}$ if it is coded as Intra, or ii) a weighted combination of channel errors $e^{c}_{\tau_{t,m},k}$'s propagated from predicted blocks *plus* $e^{s}_{t,m}$ if it is differentially coded. Note that source coding error of predicted blocks do not accumulate in descendant blocks.

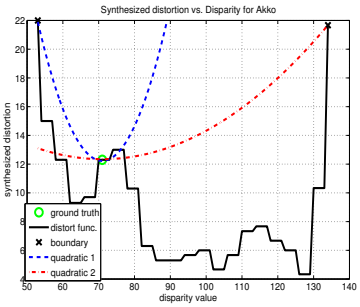
For error computation of blocks in future frames, we can compute the channel error $e^{c}_{t,m}$ for MB m as follows. First, if MB m has been ACKed (notified by receiver that it is correctly *decoded*), then obviously $e^{c}_{t,m} = 0$. Otherwise, if MB m is intra-coded, then there is a channel error only if the block is lost with probability $1-p$. In this case, the channel error is the same error as the same located block in the previous frame, $e^{c}_{t-1,m}$, plus the difference between the co-located blocks of the two frames δ. If MB m is differentially coded, then even if it is correctly received, the channel errors of the predicted blocks will nonetheless propagate. Summarizing the above, we can write $e^{c}_{t,m}$ as follows:

$$e^{c}_{t,m} = \begin{cases} 0 & \text{if MB } m \text{ is ACKed} \\ (1-p)\left(e^{c}_{t-1,m} + \delta\right) & \text{else if MB } m \text{ is Intra} \\ p\,\gamma \sum_{k \in v_{t,m}} \alpha_k\ e^{c}_{\tau_{t,m},k} + (1-p)\left(e^{c}_{t-1,m} + \delta\right) & \text{else} \end{cases} \tag{12}$$

$e^{-}_{t,m}$ can be computed recursively using previously derived (5). Expected disparity error $\epsilon_{t,m}(\rho_{t,m}, u_{t.m})$ given MV $u_{t,m}$ in previous disparity frame $Y_{\rho_{t,m}}$ can be computed in a similar fashion.

B. Computing Expected Synthesized View Distortion at Encoder

Having derived expected texture and disparity errors $e^{0}_{t,m}$ and $\epsilon^{0}_{t.m}$ for a given MB m in texture and disparity frames X^{0}_{t} and Y^{0}_{t} of view 0, we now analyze the adverse effects of these errors to expected



(a) Synthesized distortion function

(b) original frame

(c) Per-pixel curvatures

Fig. 3. Synthesized distortion and quadratic model functions for one pixel are shown in (a) for image `Akko and Kayo` [45] view 47 in (b). The resulting curvatures for for entire image are shown in (c).

distortion in the synthesized image S_t^v of virtual view v at receiver. One way to compute the expected synthesized distortion is to use a similar distortion expression as (6) for receiver that ties both error terms together in a non-trivial way. However, this introduces inter-dependency among MBs in texture and depth maps, rendering the subsequent block-level RPS optimization very difficult.

Instead, we will use the following simplification. Texture error $e_{t,m}^0$, as discussed in Section IV-B, contributes directly to the synthesized distortion. To estimate the adverse effects of disparity error $\epsilon_{t,m}^0$ to synthesized distortion, we model synthesized distortion as a quadratic function $g_{t,m}^0(\,)$ of disparity error $\epsilon_{t,m}^0$:

$$g_{t,m}^0(\epsilon_{t,m}^0) = \frac{1}{2} a_{t,m}^0 \left(\epsilon_{t,m}^0\right)^2 \tag{13}$$

where $a_{t,m}^0$ is the single parameter that describes the curvature of the quadratic function $g_{t,m}^0(\epsilon_{t,m}^0)$.

This quadratic modeling of synthesized distortion was first used in [24] for depth map coding. The key idea is to capture the synthesized view distortion sensitivity to disparity error in MB m in a single parameter $a_{t,m}^0$. In general, if the textural area corresponding to the depth MB m is smooth and inside a physical object, then synthesized distortion will not be sensitive to disparity error. The reason is that small error $\epsilon_{t,m}^0$ leading to mapping of wrong texture pixels of similar intensities in the local neighborhood will result in only small synthesized distortion. On the other hand, if the textural area corresponding to the depth MB m has large frequency contents inside a physical object or contains boundary pixels between foreground object and the background, then synthesized distortion will be sensitive to disparity error.

1) Computing block curvature $a_{t,m}^0$: Specifically, we compute parameter $a_{t,m}^0$ for each MB m as follows. First, synthesized distortion $d_{t,m}^0$ of MB m is computed as a sum of its constituent pixels (i, j)'s distortions. As discussed in Section IV-A, a texture pixel $X_t^0(i, j)$ in the left texture map can be mapped to a shifted pixel $X_t^1(i, j - Y_t^0(i, j) * \eta)$ in the right texture map. We thus express $d_{t,m}^0(\epsilon_{t,m}^0)$ as the sum of the differences

in texture pixel values between left pixels $X_t^0(i,j)$'s and mapped right pixels $X_t^1(i, j-(Y_t^0(i,j)+\epsilon_{t,m}^0)*\eta)$'s due to disparity error $\epsilon_{t,m}^0$:

$$d_{t,m}^0(\epsilon_{t,m}^0) = \sum_{(i,j)\in MB_m} |X_t^0(i,j) - X_t^1(i, j-(Y_t^0(i,j)+\epsilon_{t,m}^0)*\eta)| \tag{14}$$

As the disparity error $\epsilon_{t,m}^0$ increases, i.e., the depth value moves away from ground truth, synthesized distortion function $d_{t,m}^0(\epsilon_{t,m}^0)$ increases generally, as illustrated in Fig. 3(a). Our model $g_{t,m}^0(\epsilon_{t,m}^0)$ will simply be the sharpest of the two quadratic functions fitted to $d_{t,m}^0(\epsilon_{t,m}^0)$. For such, we identify the nearest *boundary disparity values* below and above the ground truth where $d_{t,m}^0(\epsilon_{t,m}^0)$ exceed a pre-defined threshold and construct parabolas with slope zero at ground truth, see Fig. 3(a). $a_{i,m}^0$ in (13) is the average of all chosen curvatures of pixels in m. As an example, in Fig. 3(b) and (c), the original frame from view 47 of `Akko and Kayo` [45] and the per-pixel curvatures of the quadratic model functions of the corresponding depth map are shown. We can clearly see that larger curvatures (in white) occur at object boundaries, agreeing with our intuition that a synthesized view is more sensitive to depth pixels at object boundaries.

Combining synthesized distortion due to disparity error $\epsilon_{t,m}^0$ with that of texture error $e_{t,m}^0$, we can write the combined distortion for MB m in texture and depth maps of view 0 as:

$$\bar{D}_{t,m}^0(\tau_{t,m}^0, v_{t,m}^0, \rho_{t,m}^0, u_{t,m}^0, q_t) = e_{t,m}^0(\tau_{t,m}^0, v_{t,m}^0, q_t) + g_{t,m}^0(\epsilon^0(\rho_{t,m}^0, u_{t,m}^0, q_t)) \tag{15}$$

where $\bar{D}_{t,m}^0$ denotes the expected distortion for a MB in frame t. In words, (15) says that the expected distortion for MB m is a simple sum of: i) expected texture error $e_{t,m}^0$, and ii) a quadratic term of the expected disparity error $\epsilon_{t,m}^0$. This simple model for distortion will play a significant role in simplifying the to-be-discussed RPS optimization procedure.

2) Consideration for Receiver's Error Concealment: Because of the error concealment scheme performed at decoder for MBs that are visible from both captured views, as discussed in the previous section, distortion due to errors in MB m of frames X_t^0 and Y_t^0 in view 0 will contribute to the actual synthesized view distortion *only if* errors in the corresponding pixels in frames X_t^1 and Y_t^1 of view 1 are worse. Otherwise, pixels in X_t^1 and Y_t^1 will be re-weighted more heavily in the view synthesis process. Given the above observation, we can rewrite the expected synthesized view distortion $\bar{D}_{t,m}^0$ in (15) due to texture and depth errors in MB m of texture and depth frames t of view 0, as:

$$D_{t,m}^0(\tau_{t,m}^0, v_{t,m}^0, \rho_{t,m}^0, u_{t,m}^0, q_t) = \min\left\{ e_{t,m}^0(\tau_{t,m}^0, v_{t,m}^0, q_t) + g_{t,m}^0(\epsilon_{t,m}^0(\rho_{t,m}^0, u_{t,m}^0, q_t)),\ \max_{k\in S_m^1}\left[e_{t-1,k}^1 + g_{t-1,k}^1(\epsilon_{t-1,k}^1) + \delta\right]\right\} \tag{16}$$

where S_m^1 is the set of MBs in view 1 that contains pixels corresponding to MB m in view 0. In words, (16) states that the synthesized distortion due to errors in MB m of view 0 is the smaller of distortion due to view 0 and distortion due to view 1. Note that we write distortion due to view 1 using errors in *previous frames* X_{t-1}^1 and Y_{t-1}^1 plus δ, so that $D_{t,m}^0$ will not have dependency on MVs of current frames X_t^1 and Y_t^1 of view 1. (Recall δ is the change in intensity between MBs in frame $t-1$ and t.) This avoids inter-dependency of free variables, simplifying the to-be-discussed optimization algorithm.

C. Formulation for Block-level Reference Picture Optimization

We are now ready to formally define our optimization for block-level reference picture selection. Our objective is to minimize the sum of induced synthesized distortion from MB m in both view 0 and 1:

$$\min_{q_t,\left\{\tau_{t,m}^i,v_{t,m}^i,\rho_{t,m}^i,u_{t,m}^i\right\}} \sum_{i\in\{0,1\}} \left(\sum_{m\in\bar{\mathcal{L}}_t} \bar{D}_{t,m}^i(\tau_{t,m}^i, v_{t,m}^i, \rho_{t,m}^i, u_{t,m}^i,\ q_t) + \sum_{m\in\mathcal{L}_t} D_{t,m}^i(\tau_{t,m}^i, v_{t,m}^i, \rho_{t,m}^i, u_{t,m}^i,\ q_t) \right) \tag{17}$$

where $\bar{\mathcal{L}}_t$ and $\mathcal{L}_t$ are the set of of MBs in frame t without and with corresponding MBs in the opposing captured view, respectively. Note that there are two sets of optimization variables: QP q_t for the entire frame, and MVs for individual blocks m's in the frame.

The optimization is subject to the bitrate constraint B_t at instant t:

$$\sum_{i\in\{0,1\}} \sum_{m} b_{t,m}^i(\tau_{t,m}^i, v_{t,m}^i, q_t) + \zeta_{t,m}^i(\rho_{t,m}^i, u_{t,m}^i, q_t) \le B_t \tag{18}$$

where $b_{t,m}^i$ and $\zeta_{t,m}^i$ are the resulting bit overhead required to code MB m in texture and depth frame of view i, X_t^i and Y_t^i, respectively, given selection of reference frame / MV pair, $(\tau_{t,m}^i, v_{t,m}^i)$ and $(\rho_{t,m}^i, u_{t,m}^i)$, respectively, and QP q_t.

D. Algorithm for Block-level Reference Picture Optimization

We propose an alternating two-step algorithm to solve the optimization problem. First, we fix QP q_t and solve for the reference frame / MV pairs independently as follows. Instead of solving the constrained optimization problem (17) and (18), we can solve the corresponding Lagrangian problem for given multiplier $\lambda > 0$:

$$\begin{aligned} \min_{\{\tau_{t,m}^i,v_{t,m}^i,\rho_{t,m}^i,u_{t,m}^i\}} \quad & \sum_{i\in\{0,1\}} \left(\sum_{m\in\bar{\mathcal{L}}_t} \bar{D}_{t,m}^i(\tau_{t,m}^i, v_{t,m}^i, \rho_{t,m}^i, u_{t,m}^i) + \sum_{m\in\mathcal{L}_t} D_{t,m}^i(\tau_{t,m}^i, v_{t,m}^i, \rho_{t,m}^i, u_{t,m}^i) \right) + \\ & +\lambda \sum_{i\in\{0,1\}} \sum_{m} b_{t,m}^i(\tau_{t,m}^i, v_{t,m}^i) + \zeta_{t,m}^i(\rho_{t,m}^i, u_{t,m}^i) \end{aligned} \tag{19}$$

To solve (19) optimally, it is clear that we can separately optimize each pair of MBs m in texture and

TABLE I
IMPORTANT VARIABLES IN RPS OPTIMIZATION

Symbol	Description	Symbol	Description
e	texture block error	ϵ	disparity block error
(τ, v)	texture block's RP index and MV	(ρ, u)	depth block's RP index and MV
d	worst case pixel synthesis distortion	r	pixel synthesis reliability
δ	intensity change of co-located blocks	g	model of synthesis distortion due to ϵ
$\bar{D}$	expected block distortion	D	$\bar{D}$ considering error concealment
λ	Lagrangian multiplier	R	bitrate constraint
b	texture block coding rate	ζ	depth block coding rate

disparity frame X_t^i and Y_t^i. For MB m that has no corresponding MB in opposing view, i.e., $m \in \bar{\mathcal{L}}_t$:

$$\min_{\tau_{t,m}^i, v_{t,m}^i, \rho_{t,m}^i, u_{t,m}^i} \bar{D}_{t,m}^i(\tau_{t,m}^i, v_{t,m}^i, \rho_{t,m}^i, u_{t,m}^i) + \lambda \left[b_{t,m}^i(\tau_{t,m}^i, v_{t,m}^i) + \zeta_{t,m}^i(\rho_{t,m}^i, u_{t,m}^i) \right] \qquad \forall i, m \tag{20}$$

In this case, $\bar{D}_{t,m}^i$ separates into two terms, $e_{t,m}^i$ and $g_{t,m}^i$, and the texture and depth map MB variables can be optimized separately.

For MB m that has a corresponding MB in opposing view, $\bar{D}_{t,m}^i$ is replaced by $D_{t,m}^i$ in (20). There is now an inter-dependency between texture and depth map MB variables in the distortion term. We can simplify (20) into the following two equations, where $(\tau_{t,m}^{i,*}, v_{t,m}^{i,*})$ means the best MV that minimizes texture error $e_{t,m}^i$ only, so that the searches for MV for texture and depth maps can be performed separately:

$$\begin{aligned} &\min_{\tau_{t,m}^i, v_{t,m}^i} \quad D_{t,m}^i(\tau_{t,m}^i, v_{t,m}^i, \rho_{t,m}^{i,*}, u_{t,m}^{i,*}) + \lambda\ b_{t,m}^i(\tau_{t,m}^i, v_{t,m}^i) \qquad \forall i, m \\ &\min_{\rho_{t,m}^i, u_{t,m}^i} \quad D_{t,m}^i(\tau_{t,m}^{i,*}, v_{t,m}^{i,*}, \rho_{t,m}^i, u_{t,m}^i) + \lambda\ \zeta_{t,m}^i(\rho_{t,m}^i, u_{t,m}^i) \qquad \forall i, m \end{aligned} \tag{21}$$

Equation (21) is an approximation to (20) in the following sense. If errors in the opposing views in (16) are relatively large, then errors $e_{t,m}^i$ and $\epsilon_{t,m}^i$ can indeed be separately optimally traded off with their respective rate terms $b_{t,m}^i$ and $\zeta_{t,m}^i$, as done in (21), with no loss in optimality. If errors in the opposing views are relatively small, then errors in current view can actually be ignored. Hence by setting MV of texture map to be best error-minimizing one $(\tau_{t,m}^{i,*}, v_{t,m}^{i,*})$ when optimizing MV of disparity map, we are forcing the optimization to minimize induced synthesized distortion unless errors in opposing views are very small comparatively. This conservative approach ensures the combined resulting induced synthesized distortion from texture and disparity errors will not be large. (21) is minimized by searching through all feasible MVs in all valid reference frames. This can be done efficiently, for example, in a parallel implementation. A summary of the important variables employed in our optimization and error concealment strategy is provided in Table I.

1) Optimizing QP q_t: Having discussed how (19) can be minimized assuming QP q_t is fixed, we now discuss how q_t can be optimized given the reference frame / MV pairs are fixed. We first note that

(19) needs to be solved for an appropriately chosen λ, so that the rate constraint (18) is met with little unused bandwidth B_t left over. Practically speaking, λ_t for frame t can be locally adjusted given λ_{t-1} used in previous frame $t-1$, using a search strategy such as [46]. Having solved (19) for frame t with an appropriate λ_t, we then locally search for a QP q_t so that the Lagrangian cost (19) is minimized. In particular, the Lagrangian cost is expected to have a unique local minimum at the optimal q_t. So one strategy is to first identify the change direction in q_t where the Lagrangian cost decreases, then continue the change of q_t in that direction until the cost increases. Having found the locally optimal q_t^o, we then fix QP q_t^o and search for optimal reference frame / MV pairs for blocks in the frame again. We alternate between these two steps until the variables converge.

The above strategy involves multiple encodings of frame t, which is not amenable to real-time encoding. Instead, we propose the following variation. Given QP q_t for frame t, we first search for the optimal reference frame / MV pair $(\tau_{t,m}^o, v_{t,m}^o)$ that minimizes (19) for MB m as described previously. Using $(\tau_{t,m}^o, v_{t,m}^o)$, we compute in addition the objective (19) for a local neighborhood $\mathcal{N}(q_t)$ of q_t, e.g., $\mathcal{N}(q_t) = \{q_t - 2, q_t - 1, \ldots, q_t + 2\}$, and store the results to a table. Note that QP q only affects source coding error $e_{t,m}^s(q)$ and bitrate $b_{t,m}(\tau_{t,m}, v_{t,m}, q)$ in the texture component of the objective function (similarly for depth component). If we assume error $e_{t,m}^s(q)$ and rate $b_{t,m}(\tau_{t,m}, v_{t,m}, q)$ are both exponential functions of QP q:

$$\begin{aligned} e_{t,m}^s(q) &= e^{\frac{q}{\sigma_e^2}} \\ b_{t,m}(\tau_{t,m}, v_{t,m}, q) &= e^{-\frac{q}{\sigma_b^2}}, \end{aligned} \tag{22}$$

we can estimate the model parameters σ_e and σ_b using available empirical data $e_{t,m}^s(q_t)$ and $b_{t,m}(\tau_{t,m}, v_{t,m}, q_t)$. We can thus compute objective (19) for QP neighborhood $\mathcal{N}(q_t)$ without re-encoding.

Having compute the neighborhood values for all blocks m in frame t, we check if a different QP q in neighborhood $\mathcal{N}(q_t)$ would lead to a lower Lagrangian cost than chosen q_t for frame t. If so, the optimal QP will be used for the *next* frame $t+1$. In essence, we are performing the local search for QP across temporal frames to avoid multiple encodings of the same frame.

VI. Complexity Analysis

The proposed strategies introduce additional complexity costs for both the sender and receiver of the free-viewpoint conferencing system. At the sender side, the proposal with largest computational load is the determination of the curvature parameters representing sensitivity to disparity errors employed in (13). Though in our current implementation we compute $a_{t,m}^0$ for block m in (13) by first painstakingly

computing curvature parameters for each pixel in the block as described in Section V-B, one can approximate $a^0_{t,m}$ in a computation-efficient manner by simply examining the high frequency DCT components of block m and its neighboring blocks in texture and depth maps (high frequencies can indicate depth edges and fast changing textural content.) As fast curvature parameter approximation is not the main focus of this paper, we simply assume curvature parameters can be computed efficiently and discuss complexity of our proposed optimization procedures.

Still at sender, the loss-resilient coding proposals encompass the estimation of texture/disparity errors and the optimization of RPS. Texture error estimation involves determining e^+ and e^- as combined in (10) and defined by (4) and (5), respectively. When correctly received, e^+ of a non-Intra MB of size $N \times N$ involves a weighted average over all the MB pixels and thus $N \times N$ multiplication operations. The error e^- of an incorrectly received MB is a function of δ which is taken as the difference between the current MB and the co-located MB of the previous frame. This difference contributes in $N \times N$ subtraction operations to the complexity count. Complexity analysis of disparity error estimation for encoding provides similar results. Lastly, the optimization algorithm, described in sub-section V-D, considers the tradeoff between an expected distortion measure comprised of texture and disparity errors across both views and rate terms for selecting reference pictures. It is used in substitution of the standard rate-distortion optimization at MB level employed in coding of texture and depth and thus imposes no additional computational complexity costs. The proposed QP optimization is performed at frame level, using previous estimates, as described in sub-section V-D1. The aforementioned coding proposals represent a modest complexity increment in terms of arithmetic operations. In our implementation, an average increase of only 1.9% in execution time with respect to coding with reactive feedback channel was registered. Note that for these simulations fast motion estimation was used and file I/O operations, used in our implementation, were not considered.

At the receiver, texture and disparity error estimation for decoding present a computational cost similar to the one determined for the sender. Nonetheless, when decoding, the estimation of δ in (5) is dependent on whether packets of both views are lost simultaneously or whether packets in only one view are lost. In the former case, delta is the difference among co-located MBs in the two previous frames, representing $N \times N$ subtraction operations. In the latter case, prior to estimating δ as a difference of MBs, the correctly received pixels from the adjacent view are projected onto the view with losses, representing an extra $N \times N$ addition operations. Furthermore, at the receiver, the worst case distortion defined in (6) must be calculated for the virtual view prior to adaptive blending. This calculation represents $N \times N \times (2\epsilon_{t,m} + 1)$ subtraction operations. Here, the disparity error $\epsilon_{t,m}$ is content dependent and generally small except

along boundary pixels with large intensity differences. The adaptive blending step, based on worst case distortion estimates, is equivalent in complexity to the standard blending procedure which it substitutes. In conclusion, our decoding proposals also impose a modest complexity increment in terms of additional arithmetic operations.

VII. Experimentation

We verify the merits of our proposed block-level RPS encoding strategy (proactive-feedback) at sender as well as our error-concealment strategy during view synthesis (adaptive blending) at receiver through extensive experiments in this section. We first describe our experimental setup, then present streaming results in terms of objective measures (PSNR and SSIM) and subjective comparisons.

A. Experimental Setup

Our proposed proactive-feedback RPS encoding strategy is compared to two RPS-based error resilient alternatives. In all RPS strategies we assume that a lossless feedback channel is present. This channel is used for transmission of acknowledgment packets from receiver to sender, after a round-trip-time delay (RTT), informing the status of transmitted packets (i.e., correctly received or lost). In our experiments RTT is fixed to 133 ms (4 frames of delay at 30 fps).

The first RPS alternative reacts to feedback information by avoiding the use of loss-affected regions in past frames as reference for coding of future MBs. This alternative is termed *reactive-feedback* and serves as a baseline for comparison. Note that loss-affected regions include those of confirmed loss in a frame RTT or further into the past, *and* regions of frames of more recent past whose encoding was predicted from an area of confirmed loss.

The second RPS alternative employed in our tests is based on our previous work [28]. Though it also uses receiver-sent feedbacks to define loss-affected regions, encoding is performed more proactively: it recursively computes loss probabilities and resulting errors for all reference areas in texture and depth maps. Feedback information is additionally used to reduce probabilistic dependency chains and estimate texture / disparity errors more precisely.

Our proposed proactive-feedback RPS encoding strategy differs from [28] in two important respects. First, the encoder considers inter-view redundancy when estimating distortion of the synthesized view, given the receiver performs adaptive blending. Second, source distortion is also considered during error estimation, and a newly designed QP optimization procedure is employed in addition to minimize the combined effect of source and channel errors.

Along with proactive-feedback RPS at encoder, our proposed adaptive blending view synthesis strategy at decoder is also assessed. Adaptive blending has been implemented inside the MPEG View Synthesis Reference Software (VSRS v3.5) [6]. Comparisons are drawn against alternatives employing the standard version of the software.

From the described RPS encoding and synthesis components, three setups are tested in the following sections:

1) `RFC`: encoding with reactive feedback channel (and fixed QP) plus standard blending,
2) `RPS`: proactive RPS encoding [28] with feedback (and fixed QP) plus standard blending,
3) `ARPS`: proposed proactive-feedback RPS encoding plus the proposed adaptive blending.

All results are evaluated in terms of synthesized view quality. Specifically, for each multi-view sequence, we first select three neighboring views and transmit only the left and right views, reserving the center view for synthesis at decoder and comparison to uncoded captured images. We use test sequences from the Nagoya University database [45]. The selected sequences are the first 150 frames of: `Pantomime` (1280×960 pixels), `Kendo` (1024×768 pixels) and `Akko and Kayo` (640×480 pixels), all at 30fps. For `Pantomime` view 40 was synthesized using views 39 and 41. For `Kendo` view 2 is interpolated from view 1 and 3. For `Akko and Kayo` views 47 and 49 are used to synthesize view 48. The view synthesis software [6] was set to work with integer pixel precision.

Our encoding scheme is implemented only for $P16 \times 16$ mode in H.264/AVC JM reference software v18.0 [47]. Thus, only modes available in all simulations are $P16 \times 16$, Skip or Intra blocks (i.e., no sub-block partitions are allowed). More extensive comparison using larger number of available modes is a subject of future study.

Four transport packets are used for each depth map frame, while twelve are used for each texture frame packets due to higher associated bit rates. Simulations include packet loss rate of 2%, 5% and 8% for both texture and depth maps. We assume the picture parameter set (PPS) and sequence parameter set (SPS) are reliably transmitted out-of-band. Both depth maps and texture are encoded using 128 pixel search window for motion estimation, CABAC entropy encoder and *IPPP* predictive frame structure. When a MB is lost during transmission, the co-located block from the previous frame is used as simple error concealment.

We compute permissible transmission bit budget for each sequence (texture plus depth) and each packet loss rate as follows. First, we compute the raw source compression bitrate if QP is fixed at 28: 8.4 Mbps for `Pantomime`, 5.1 Mbps for `Kendo` and 5.3 Mbps for `Akko and Kayo`. Then, for each packet loss rate, we add a sufficiently large bit budget overhead for transmission schemes to combat packet losses.

TABLE II
BITRATES USED IN ALL SETUPS (CONSIDERS BOTH VIEWS AND DEPTH MAPS)

Sequence	Loss Rate	Bitrate
Pantomime	2%	8.7 Mbps
	5%	9.0 Mbps
	8%	9.4 Mbps
Kendo	2%	6.3 Mbps
	5%	6.6 Mbps
	8%	6.9 Mbps
Akko and Kayo	2%	6.5 Mbps
	5%	6.9 Mbps
	8%	7.3 Mbps

The resulting transmission bitrates for each sequence, for each percentage of lost packets, are shown in Table II. Each transmission scheme must meet this transmission bit budget; for RFC and RPS, it would mean through the proper control of λ during optimization. For ARPS, it will have in addition the control of QP.

Operational bitrate is in general inversely proportional to λ, and sophisticated procedures to find the best λ can be devised [46]. Note that in practice, however, the choice of λ does not have to be complicated; one that is used in previous frame can be reused for current frame, with an optional local adjustment if the resulting rate is too far below or above the rate constraint.

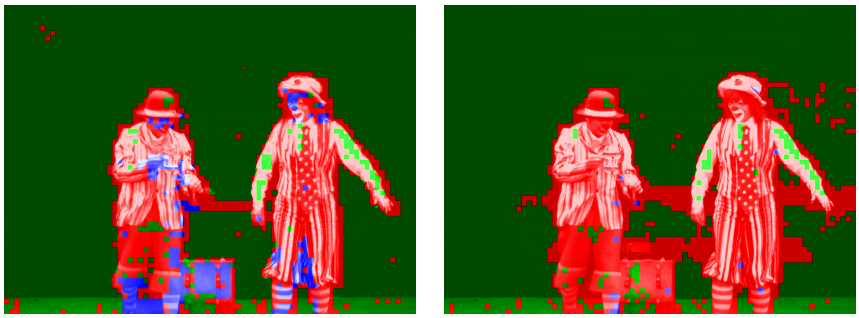
B. Comparison of Selected Coding Modes

We now illustrate how coding modes are selected by the standard (H.264 without modifications), RFC and ARPS encoding schemes. In Table III, the percentages of different MB modes chosen during encoding across all frames of views 39 and 41 of Pantomime are shown. The modes are Intra, Skip, and two sub-types of $P16\times16$ mode: P_{t-1} when the previous frame is used as reference and P_{t-n} when the chosen reference frame is further into the past. We first observe that both RFC and ARPS selected smaller percentages of P_{t-1} mode than standard encoding. This is because P_{t-1} can lead to long dependency chains due to differential coding, which can lead to long error propagation. Instead, RFC and ARPS selected larger percentage of P_{t-n} mode, which is by comparison more loss-resilient. We also observe that ARPS selected fewer Intra mode than RFC. While both Intra and P_{t-n} offer loss resiliency, P_{t-n} can do so at a smaller encoding cost, illustrating the advantage of ARPS over RFC.

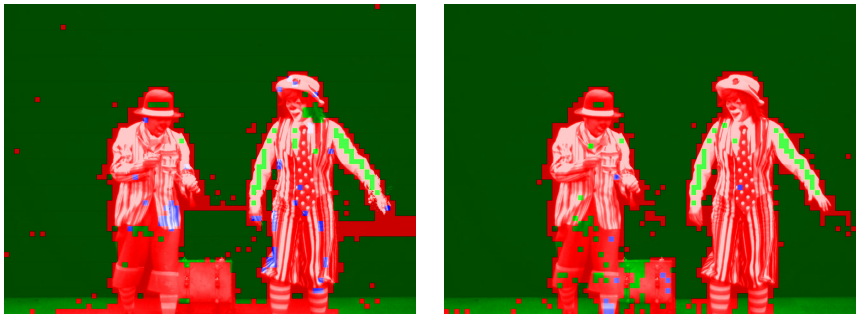
In Fig. 4, view 39 and 41 of frame 48 are shown. The chosen encoding modes are indicated by color. It can be observed in view 39 that RFC encodes several blocks as Intra to improve loss resiliency, while in ARPS most of these MBs are encoded as Inter mode which is more coding efficient. Also, in view 41 of ARPS more blocks are coded as Skip in comparison to view 39, mostly due to the fact that they have

TABLE III
PERCENTAGE OF ENCODING MODES OF PANTOMIME AT 5% ERROR RATE

View	Mode	Standard	RFC	ARPS
39	Skip	68.26%	66.46%	65.01%
	P_{t-1}	26.43%	22.84%	24.77%
	P_{t-n}	4.55%	6.12%	9.37%
	Intra	0.76%	4.58%	0.85%
41	Skip	68.34%	62.27%	66.01%
	P_{t-1}	26.30%	21.43%	22.35%
	P_{t-n}	4.34%	8.12%	9.71%
	Intra	1.02%	7.68%	1.94%



(a) RFC view 39 (b) ARPS view 39



(c) RFC view 41 (d) ARPS view 41

Fig. 4. Encoding modes used in Pantomime, frame 48. Skip, Intra and Inter ($P16\times16$) blocks are shown in green, blue and red, respectively.

already been protected in view 39 and are therefore redundant in view 41.

C. Objective Streaming Performance

We now present experimental results using PSNR and SSIM [48] as objective quality metrics. A summary of average PSNR results obtained for synthesized views (assuming original views as ground truth) can be found in Table IV. Averages are computed across all frames for each sequence for all three comparison schemes.

We first observe that RPS, based on previous work [28], already outperforms RFC. Note that these gains increase as the error rates increase for all sequences. Nevertheless, the largest gains in average PSNR are obtained for ARPS setup relative to RPS. For Pantomime at 5% error rate, gain is as high as 2.91dB.

TABLE IV
AVERAGE PSNR OVER ALL FRAMES FOR TESTED SETUPS.

Sequence	Loss Rate	Avg. PSNR		
		RFC	RPS	ARPS
Pantomime	2%	31.77 dB	32.05 dB	33.57 dB
	5%	28.77 dB	29.10 dB	32.01 dB
	8%	26.86 dB	27.44 dB	30.12 dB
Kendo	2%	32.63 dB	32.86 dB	33.45 dB
	5%	30.20 dB	30.61 dB	32.24 dB
	8%	28.63 dB	29.14 dB	30.50 dB
Akko and Kayo	2%	26.40 dB	26.45 dB	26.89 dB
	5%	25.68 dB	25.81 dB	26.71 dB
	8%	25.52 dB	25.74 dB	26.35 dB

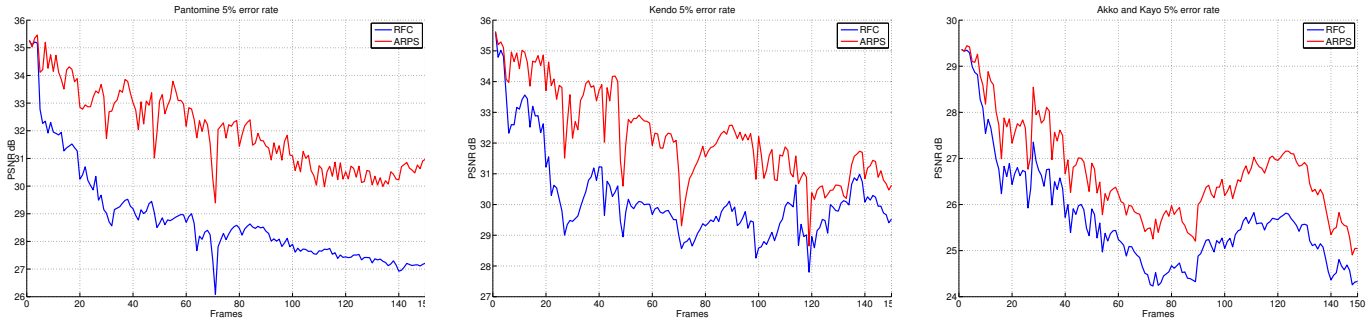
For Kendo and Akko and Kayo at the same error rates, gains are 1.63dB and 0.90dB, respectively. These gains can be explained by the two intrinsic novelties of our proposed strategy: i) exploiting inter-view dependency during encoding allows us to achieve the same level of protection for each MB with a smaller bitrate and improve synthesis at decoder through adaptive blending, and ii) considering source distortion and adjusting frame QP using our proposed joint optimization allows more MB's to benefit from RPS without significant increases in bitrate.

In Fig. 5, PSNR values for the proposed ARPS and the baseline RFC are presented for each frame. Both schemes consider a 5% packet loss rate. Due to the use of feedback, we see that both schemes are generally able to avoid continuous propagation of errors due to losses. Nevertheless, our method can better withstand the transient effect of packet losses by providing stronger protection to more important regions. For specific frames, significant gains, far above average values, can be achieved in PSNR: 4.98dB for Pantomime (frame 55), 4.09dB for Kendo (frame 29) and 1.46dB for Akko and Kayo (frame 126).

In terms of SSIM, ARPS consistently outperforms RFC for every tested frame. For Pantomime the average SSIM over all frames using RFC is 0.9314, while using ARPS is 0.9551, with a maximum difference of 0.0218. For Kendo, average values are 0.9372 and 0.9442 for RFC and ARPS, respectively, with a maximum gain of 0.0151. For Akko and Kayo, the maximum difference between both setups is 0.0241, RFC has an average value of 0.7652 and ARPS of 0.7738. Despite of the improvements in SSIM, scores for Akko and Kayo are relatively low for both schemes, which can be explained by poor view synthesis quality for this particular 3D video sequence.

D. Subjective Comparison

For subjective comparisons, Fig. 6-8 show, for each sequence, synthesized views using encoding schemes RFC and ARPS. The more significant differences among images are circled in red. Visual comparisons between the two methods are significant for all sequences, and they can be easily perceived at normal



(a) Pantomime (b) Kendo (c) Akko and Kayo

Fig. 5. PSNR per frame for ARPS and RFC at 5% packet loss rate for different sequences.

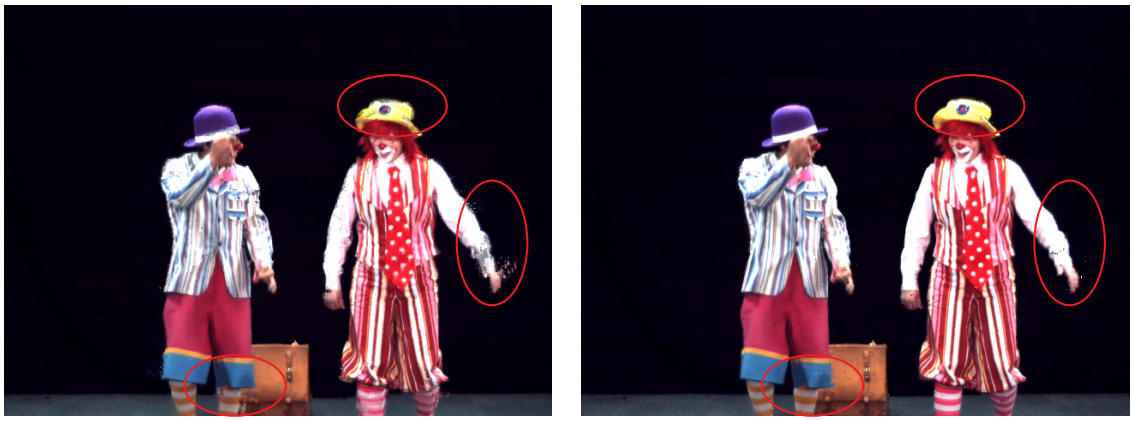
frame rate 30 fps. The PSNR and SSIM indeces of each image are also indicated.

In Kendo, for example, we can see errors in the RFC image (Fig. 7(a)) on the hands and sword of the swordsman of the right, around the audience and around the plant. The errors in the background, around audience and plant, are residue created by using co-located blocks to conceal errors in past frames which were not yet acknowledged by the feedback channel. Errors on the hands and sword are most likely caused by lost packets in depth maps, generating erroneous horizontal shifts. Note that using ARPS (Fig. 7(b)) visible errors on the hands of the swordsman are greatly reduced. The unequal protection during coding of the depth maps provided by ARPS helps conserve the sharp edges around sword and hands. The proposed adaptive blending significantly improves the synthesized view by applying weights that consider the estimated error in each pixel. The residue errors around the audience is minimal in Fig. 7(b), resulting in a more pleasing subjective viewing.

Similar results may be observed in Fig. 6 for Pantomime on the left hand and hat of the clown on the right, and on the left leg of the other clown. For Akko and Kayo (Fig 8) the subjective gains are more noticeable around the heads of both persons.

VIII. Conclusion

To enable free-viewpoint video conferencing, in this paper we study a real-time streaming system where texture and depth videos from two captured viewpoints are transmitted, so that synthesis and display of any intermediate viewpoint at receiver is enabled via depth-image-based rendering (DIBR). To provide resiliency over loss-prone transmission networks, we propose first to adaptively blend a synthesized pixel at receiver, so that the pixels from the more reliable transmitted view is weighted heavier during synthesis. We then propose a reference picture selection (RPS) scheme at sender, so that pro-actively important code blocks containing pixels vital to synthesized view quality are predicted from older past frames, lowering their expected error due to error propagation in differentially coded video. Finally, we analyze synthesized



(a) (b)

Fig. 6. Synthesized views for `Pantomime`, frame 55, at 5% packet loss with (a) RFC (PSNR: 28.81dB, SSIM: 0.9413) and (b) ARPS (PSNR: 33.78dB, SSIM: 0.9626).

(a) (b)

Fig. 7. Synthesized views for `Kendo`, frame 107, at 5% packet loss with (a) RFC (PSNR: 28.82 dB, SSIM: 0.9297) and (b) ARPS (PSNR: 30.80 dB, SSIM: 0.9383).

(a) (b)

Fig. 8. Synthesized views for `Akko and Kayo`, frame 12, at 5% packet loss with (a) RFC (PSNR: 27.84 dB, SSIM: 0.8163) and (b) ARPS (PSNR: 28.88 dB, SSIM: 0.8297).

view distortion sensitivities to texture versus depth errors, so that relative importance can be determined for texture and depth code blocks for system-wide RPS optimization. Experimental results show that our

proposed scheme, combined with feedback information, can significantly outperform a reactive feedback channel, not only by objectives metrics, but subjectively as well.

References

[1] A. Kubota, A. Smolic, M. Magnor, M. Tanimoto, T. Chen, and C. Zhang, "Multi-view imaging and 3DTV," in *IEEE Signal Processing Magazine*, vol. 24, no.6, November 2007.

[2] 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.

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

[4] M. Flierl, A. Mavlankar, and B. Girod, "Motion and disparity compensated coding for multiview video," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no.11, November 2007, pp. 1474–1484.

[5] 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.

[6] M. Tanimoto, T. Fujii, K. Suzuki, N. Fukushima, and Y. Mori, "Reference softwares for depth estimation and view synthesis," in *ISO/IEC JTC1/SC29/WG11 MPEG2008/M15377*, Archamps, April 2008.

[7] 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.

[8] 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.

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

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

[11] 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.

[12] 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.

[13] E. Kurutepe, M. R. Civanlar, and A. M. Tekalp, "Client-driven selective streaming of multiview video for interactive 3DTV," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no.11, November 2007, pp. 1558–1565.

[14] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[15] P. Merkle, A. Smolic, K. Muller, and T. Wiegand, "Efficient prediction structures for multiview video coding," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no.11, November 2007, pp. 1461–1473.

[16] Y. Zhang and Z. Xiong, "On the sum-rate loss of quadratic gaussian multiterminal source coding," in *IEEE Transactions on Information Theory*, vol. 57, no.9, September 2011, pp. 5588–5614.

[17] 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.

[18] G. Shen, W.-S. Kim, S. 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.

[19] J. Gautier, O. L. Meur, and C. Guillemot, "Depth map coding: exploiting the intrinsic properties of scenes and surface layout," in *Picture Coding Symposium 2012*, Krakow, Poland, May 2012.

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

[21] W.-S. Kim, A. Ortega, P. Lai, D. Tian, and C. Gomila, "Depth map distortion analysis for view rendering and depth coding," in *IEEE International Conference on Image Processing*, Cairo, Egypt, November 2009.

[22] ——, "Depth map coding with distortion estimation of rendered view," in *SPIE Visual Information Processing and Communication*, San Jose, CA, January 2010.

[23] 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.

[24] G. Cheung, J. Ishida, A. Kubota, and A. Ortega, "Transform domain sparsification of depth maps using iterative quadratic programming," in *IEEE International Conference on Image Processing*, Brussels, Belgium, September 2011.

[25] G. Valenzise, G. Cheung, R. Galvao, M. Cagnazzo, B. Pesquet-Popescu, and A. Ortega, "Motion prediction of depth video for depth-image-based rendering using don't care regions," in *Picture Coding Symposium 2012*, Krakow, Poland, May 2012.

[26] P. Merkle, C. Bartnik, K. Muller, D. Marpe, and T. Weigand, "3D video: Depth coding based on inter-component prediction of block partitions," in *2010 Picture Coding Symposium*, Krakow, Poland, May 2012.

[27] I. Daribo, D. Florencio, and G. Cheung, "Arbitrarily shaped sub-block motion prediction in texture map compression using depth information," in *2010 Picture Coding Symposium*, Krakow, Poland, May 2012.

[28] B. Macchiavello, C. Dorea, M. Hung, G. Cheung, and W. t. Tan, "Reference frame selection for loss-resilient texture and depth map coding in multiview video conferencing," in *Proc. of the IEEE International Conference on Image Processing*, Orlando, FL, September 2012.

[29] Y. Wang and Q.-F. Zhu, "Error control and concealment for video communication: A review," *IEEE Proceedings*, vol. 86, pp. 974–997, 1998.

[30] Y. Wang, M. Claypool, and R. Kinicki, "modeling RPS and evaluating video repair with VQM," *IEEE Trans. Multimedia*, vol. 11, no. 1, pp. 128–37, 2009.

[31] G. Cheung, W.-T. Tan, and C. Chan, "Reference frame optimization for multiple-path video streaming with complexity scaling," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no.6, June 2007, pp. 649–662.

[32] E. Martinian and M. Trott, "Delay-optimal burst erasure code construction," in *IEEE Int. Symp. on Info. Theory*, Nice, France, June 2007.

[33] P. Lambert, W. Neve, Y. Dhondt, and R. Walle, "Flexible macroblock ordering in h.264/avc," in *J. of Visual Comm. & Image Representation*, vol. 17, 2006.

[34] T. Stockhammer and M. Bystrom, "H.264/AVC data partitioning for mobile video communication," in *IEEE International Conference on Image Processing*, October 2004.

[35] T. Troger, "Inter-sequence error concealment techniques for multi-broadcast tv reception," *IEEE Trans. Broadcasting*, vol. 57, no. 4, pp. 777–793, 2011.

[36] Y. Eisenberg, F. Zhai, C. Luna, R. Berry, and A. Katsaggelos, "Variance-aware distortion estimation for wireless video communications," in *ICIP*, September 2003.

[37] A. Tan, A. Aksay, G. Akar, and E. Arikan, "Rate-distortion optimization for stereoscopic video streaming with unequal error protection," in *EURASIP Journal on Advances in Signal Processing*, 2009.

[38] C. Hou, R. Pan, Z. Yuan, and L. Yang, "Distortion analysis and error concealment for multiview video transmission," in *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*, Tianjin, China, March 2010.

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

[40] B. Macchiavello, C. Dorea, M. Hung, G. Cheung, and W. t. Tan, "Reference frame selection for loss-resilient depth map coding in multiview video conferencing," in *IS&T/SPIE Visual Information Processing and Communication Conference*, Burlingame, CA, January 2012.

[41] MPEG Video Group, "Call for contributions on 3D video test material," in *ISO/IEC JTC1/SC29/WG11 N9595*, Antalya, Turkey, January 2008.

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

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

[44] H.-Y. Shum, S.-C. Chan, and S. B. Kang, *Image-Based Rendering*. Springer, 2007.

[45] "Nagoya university ftv test sequences," in *http://www.tanimoto.nuee.nagoya-u.ac.jp/*.

[46] Y. Shoham and A. Gersho, "Efficient bit allocation for an arbitrary set of quantizers," in *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no.9, September 1988, pp. 1445–1453.

[47] "JM H.264 reference software v18.0," in *http://iphome.hhi.de/suehring/tml/*.

[48] 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.

Bruno Macchiavello received the Bachelor degree in electronic engineering from Pontical Catholic University, Lima, Peru, in 2001, and the M.Sc. and D.Sc. degrees in electrical engineering from the Universidade de Brasilia, Brasilia, Brazil, in 2004 and 2009, respectively. Currently, he is an Assistant Professor of the Computer Science Department at the University of Brasilia, Brasilia, Brazil. His main research interests include video coding, image processing, distributed source coding, mutiview and 3D processing.

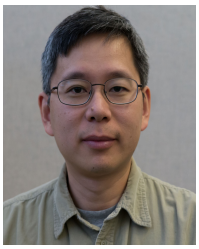
Camilo Dorea received the B.S. degree from the University of Brasilia, Brazil, in 1997, the M.S. degree from the University of Maryland at College Park, USA, in 1999, both in electrical engineering, and the Ph.D. degree in telecommunications from the Technical University of Catalonia (UPC), Barcelona, Spain, in 2007. From 2007 to 2008 he was with Thomson Corporate Research at Princeton, NJ, USA. In 2009 he joined the Department of Computer Science at the University of Brasilia where he is currently Assistant Professor. His research interests include video segmentation and analysis, video coding, and mutiview and 3D processing.

Edson Mintsu Hung received the Eng., M.Sc. and D.Sc. degrees from the Dept. of Electrical Enginnering at Universidade de Brasilia, Brazil, in 2004, 2007 and 2012, respectively. From 2004 to 2007 he was with Research and Development Center for the Security of Communication (CEPESC) of the Brazilian Intelligence Agency (ABIN), where he was a hardware and software developer. Mintsu is also CTO of Mux Tecnologia, a spin-off company that he co-founded in 2007. In 2010, he joined the Faculty of Electronic Engineering at Universidade de Brasilia in a new campus at Gama. His reseach interests include signal processing, video coding, image and video processing and multiview processing.

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.

His research interests include 3D visual representation and immersive communication. He has published over 120 international conference and journal publications. He has served as associate editor for IEEE Transactions on Multimedia from 2007 to 2011 and currently serves as associate editor for DSP Applications Column in IEEE Signal Processing Magazine and APSIPA journal on signal & information processing, and as area editor for EURASIP Signal Processing: Image Communication. He currently serves as member of the Multimedia Signal Processing Technical Committee (MMSP-TC) in IEEE Signal Processing Society (2012-2014). He has also served as area chair in IEEE International Conference on Image Processing (ICIP) 2010, 2012-2013, technical program co-chair of International Packet Video Workshop (PV) 2010, track co-chair for Multimedia Signal Processing track in IEEE International Conference on Multimedia and Expo (ICME) 2011, symposium co-chair for CSSMA Symposium in IEEE GLOBECOM 2012, and area chair for ICME 2013. He is invited as plenary speaker for IEEE International Workshop on Multimedia Signal Processing (MMSP) 2013 on the topic "3D visual communication: media representation, transport and rendering". He is a co-author of best student paper award in IEEE Workshop on Streaming and Media Communications 2011 (in conjunction with ICME 2011), best paper finalists in ICME 2011 and ICIP 2011, best paper runner-up award in ICME 2012, and best student paper award in ICIP 2013.



Wai-tian Tan received the B.S. degree in electrical engineering from Brown University, in 1992, the M.S.E.E. degree from Stanford University in 1993, and the Ph.D. degree in electrical engineering and computer science from the University of California, Berkeley, in 2000. He is a principal engineer with the Enterprise Networking Group in Cisco since 2013. Prior to that, he was a principal research scientist at Hewlett Packard Laboratories. His research interests are in developing adaptive streaming systems under different practical constraints, and more generally in coding, communications, and new applications of video.