Predictive Coding of Depth Images Across Multiple Views

Yannick Morvan[a], Dirk Farin[a] and Peter H. N. de With[a,b]

[a] Eindhoven University of Technology, P.O. Box 513, The Netherlands;
[b] LogicaCMG, P.O. Box 7089, 5605 JB Eindhoven, The Netherlands
y.morvan@tue.nl

ABSTRACT

A 3D video stream is typically obtained from a set of synchronized cameras, which are simultaneously capturing the same scene (multiview video). This technology enables applications such as free-viewpoint video which allows the viewer to select his preferred viewpoint, or 3D TV where the depth of the scene can be perceived using a special display. Because the user-selected view does not always correspond to a camera position, it may be necessary to synthesize a virtual camera view. To synthesize such a virtual view, we have adopted a depth-image-based rendering technique that employs one depth map for each camera. Consequently, a remote rendering of the 3D video requires a compression technique for texture and depth data. This paper presents a predictive-coding algorithm for the compression of depth images across multiple views. The presented algorithm provides (a) an improved coding efficiency for depth images over block-based motion-compensation encoders (H.264), and (b), a random access to different views for fast rendering. The proposed depth-prediction technique works by synthesizing/computing the depth of 3D points based on the reference depth image. The attractiveness of the depth-prediction algorithm is that the prediction of depth data avoids an independent transmission of depth for each view, while simplifying the view interpolation by synthesizing depth images for arbitrary view points. We present experimental results for several multiview depth sequences, that result in a quality improvement of up to 1.8 dB as compared to H.264 compression.

Keywords: Multiview video coding, 3D video coding, view synthesis, warping-based predictive coding, depth

1. INTRODUCTION

A 3D video is typically obtained from a set of synchronized cameras, which are capturing the same scene from different view points (multiview video). This technique enables applications such as free-viewpoint video or 3D-TV. First, the free-viewpoint video application provides the ability for users to interactively select a viewpoint of the scene. Second, with 3D-TV, the depth of the scene can be perceived using a multiview display. The corresponding display technology is based on showing several views of the same scene. By observing slightly different views of the scene, the human brain integrates these views into a 3D representation of the scene. To represent a 3D scene, several Image Based Rendering (IBR) approaches have been investigated and classified[1] depending on the accuracy of the geometric description of the scene.

A first class of techniques considers the use of multiple views (N-texture) of the scene but does not require any geometric description of the scene. In this case, novel views of the scene are synthesized by interpolating the image from captured images. However, to obtain high-quality 3D video rendering, it is necessary to capture the video scene from a large number of viewpoints, i.e. oversampling of the scene. Such an oversampling is inefficient because it does not exploit the redundancy across views.

A second class of techniques involves the use of a geometric description of the scene. The scene geometry is typically described by a depth map, or depth image, that specifies the distance between a point in the 3D world and the camera. In practice, a depth image can be estimated from multiple images by calculating the parallax motion of pixels between views. Using depth images, new views can be subsequently rendered using image warping algorithms. For a 3D-TV application, it is assumed that the scene is observed from a narrow field of view. In this particular case, a combination of only one texture and one depth video signal is sufficient to provide appropriate rendering quality (1-texture/1-depth). However, for free-viewpoint video applications, view rendering at an arbitrary position can be requested by the user. To address this problem, one alternative is to

combine both techniques (N-texture and 1-texture/1-depth) by using one depth image per texture image, i.e. N-depth/N-texture.[2]

A major problem when dealing with N-texture and N-depth video signals is the large amount of data to be encoded, decoded and rendered. For example, an independent transmission of 8 views of the "Breakdancers" sequence requires about 10 $Mbit/s$ and 1.7 $Mbit/s$ with a PSNR of 40 dB for the texture and depth data, respectively. Therefore, in a practical solution, a suitable compression algorithm is more efficient. Additionally, considering the free-viewpoint video application, one important feature is that the user can select his viewpoint and change it interactively. To provide this interactivity, the compression algorithm should be able to perform low-delay access to a selected view. For example, let us consider that a user selects the left-most camera to be rendered and subsequently switches to the right-most camera. In this case, all in-between views from the left-most to the right-most camera should be decoded to generate a smooth transition between views.
As a summary, the two important properties of the desired algorithm are that multiple texture and depth signals should be compressed efficiently and that random-access to encoded frames should be allowed to synthesize views in real-time.
In this paper, we propose a technique for coding multiple depth images across multiple views which

1. performs predictive coding of depth images across the views, and
2. has a prediction structure that allows fast random-access.

We propose to use a central reference depth image that enables prediction of neighboring multiple depth images, with the purpose of compressing these multiple depth images (see Figure 1 for illustration of depth images captured and predicted across multiple views). The proposed depth-prediction technique works by warping the depth of 3D points based on the reference depth image. When the distance between cameras is high, not only some part of the texture is occluded but also part of the depth. In this case, missing depth data can be filled by using a residual depth image. For compression, since the residual depth image shows similar properties to a residual motion-compensated image, we perform the encoding of such a residual depth image in a way that is similar to MPEG P-frames. To perform depth image prediction or synthesis, we propose a variant of the *Relief Texture*[3] mapping technique, adapted to the case of multiview geometry (as opposed to computer graphics geometry).
To provide random access to different views, we employ a single depth image as a reference from which neighboring depth maps are predicted. While only one reference image is used for predictive coding, we show that the depth-image prediction is sufficiently accurate to obtain an efficient compression.
To evaluate the efficiency of the depth-prediction across the views, we have integrated the relief-texture-based prediction algorithm into an H.264 encoder. Experimental results show that the proposed depth-prediction

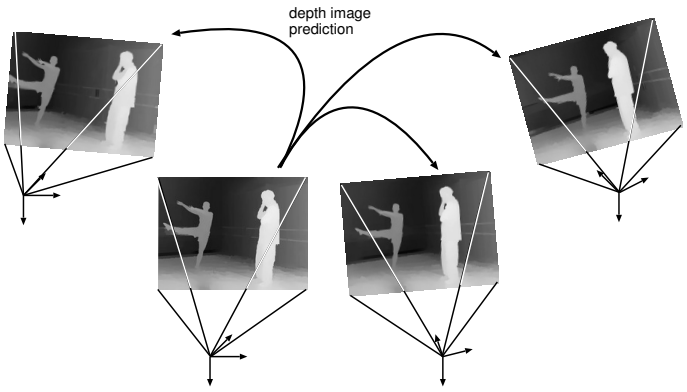


Figure 1. Multiple depth images can be obtained by calculating the parallax motion of pixels between views. For compression of depth data, depth images can be predicted using the geometry of multiple views.

algorithm yields up to 1.8 dB improvement when compared to an independent compression of depth images using H.264 coding.

It can be noted that the presented algorithm can be employed to perform the prediction and compression of *texture* data across multiple views, as well. However, because depth images are employed to perform a warping-based texture prediction, it is necessary to first address the problem of the encoding of the depth signal prior to texture data.

The remainder of this paper is organized as follows. Section 2 describes the general framework for rendering and coding multiview video. Section 3 provides details about the depth-prediction algorithm while Section 4 shows how the prediction algorithm can be integrated into an H.264 encoder. Experimental results are provided in Section 5 and the paper concludes with Section 6.

2. RENDERING AND CODING FOR FREE-VIEWPOINT VIDEO

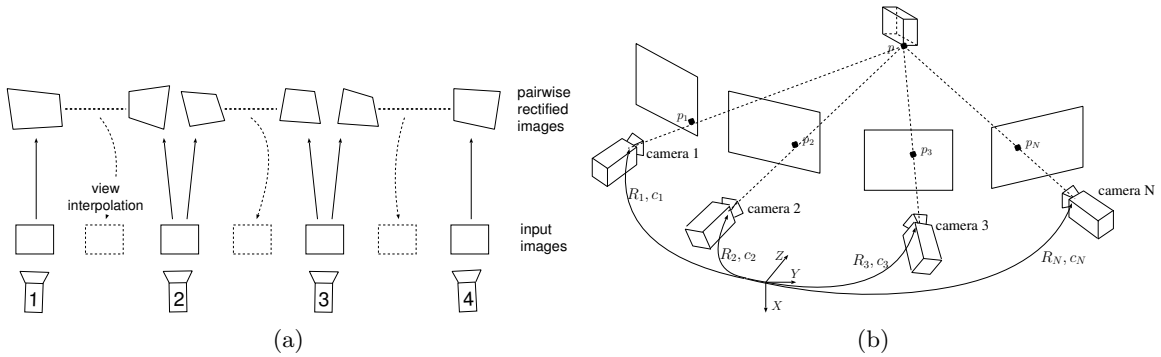
This section first aims at describing a system that enables free-viewpoint video in which the user can freely choose the camera position. Second, we discuss several prediction structures for multiview coding. More specifically, we show why an appropriate image-prediction structure is necessary to perform free-viewpoint video. Finally, we introduce a frame-prediction algorithm, different from the conventional block-based motion compensation, that employs the geometry of multiple views to predict depth images across multiple views.

2.1. Image Based Rendering for free-viewpoint video

To synthesize user-selected views of the video scene, various image-based rendering techniques have been developed.[1] The two main techniques use either a geometric model of the scene, or the user-selected views are interpolated from the neighboring cameras. Recently, it has been shown that using a mixture of both techniques enables real-time free-viewpoint video rendering.

One approach[4] allows to synthesize intermediate views along a chain of cameras. The algorithm estimates the epipolar geometry between each pair of successive cameras and rectifies the images pairwise (see Figure 2(a)). Disparity images are estimated for each pair of cameras and synthetic views are interpolated using an algorithm similar to the *View Morphing*[5] technique.

An alternative method[2] employs a similar video capturing system composed of a set of multiple cameras. As opposed to the previous approach, the cameras are fully calibrated prior to the capture session (see Figure 2(b)). Therefore, because the cameras are calibrated, the depth can be subsequently estimated for each view. To perform view synthesis, given the depth information, 3D warping techniques can be employed. Given a user-selected camera position, the algorithm warps the two nearest-neighboring views using their respective depth images. Both warped views are finally blended to generate the final rendered image.



(a) (b)

Figure 2. In a first approach (Figure 2(a)), the captured images are rectified pairwise. Disparity estimation and view interpolation between a pair of cameras is carried out on the rectified images. In an alternative approach (Figure 2(b)), the respective position and orientation of each camera is known. Depth images can therefore be estimated for each view and an image-warping algorithm can be used to synthesize virtual views.

For both approaches, it can be seen that 3D information is encoded using one depth image for each view. Therefore, to provide the ability for remote users to freely select a viewpoint in the scene, it is necessary to transmit the set of multiple texture images combined with their corresponding depth map.

2.1.1. Predictive-coding structures for random access

Several multiview coding structures for compressing multiview videos have been proposed in literature.[6]

A first straightforward solution, called *Simulcast coding*, consists of independently encoding (see Figure 3(a)) the set of multiple views using an H.264 encoder. One advantage of simulcast coding is that standardized video encoders can be used for multiview coding. However, simulcast coding neither exploits the redundancy between views, nor provides random access to different views.

As a second approach, it has been proposed[7, 8] to multiplex the views such that a single video stream is generated. The resulting video is then compressed using an H.264 encoder. The advantages of this technique are twofold. First, the use of a standardized video coding algorithm provides the backward compatibility with already existing technology. Second, the H.264 encoder can perform adaptive reference frame selection, so that the spatial and temporal correlation in the multiplexed-views sequence can be adaptively exploited (see Figure 3(b)). For this reason, H.264-based multiview video coders exploiting spatio-temporal dependencies yield a good coding performance. However, exploiting spatio-temporal correlation creates coding dependencies between past, future and neighboring views, thereby hampering an easy random-access to individual views.

To provide random access to neighboring views while still exploiting spatio-temporal redundancy, it has been proposed[2] to use predefined views as a spatial reference from which neighboring views are predicted. Observing the coding structure of Figure 3(c), it can be seen that temporal correlation is exploited *only* by the central reference view. Similarly, *only* non-central views exploit the spatial inter-view redundancy. For this reason, by exploiting an appropriate mixture of temporal and spatial prediction, views along the chain of cameras can be randomly accessed. We have therefore adopted this last coding structure to perform multiview coding. It goes without saying that the adopted prediction structure and algorithm is important for coding performance. For this reason, we discuss several prediction techniques in the next section.

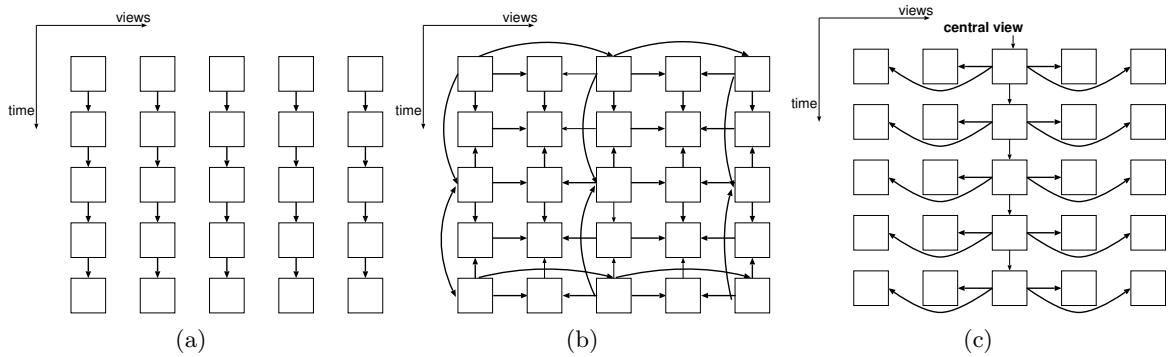


Figure 3. Figure 3(a) shows a simulcast coding structure where each view is encoded independently. Figure 3(b) shows a coding structure where both spatial and temporal redundancy are exploited. Figure 3(c) depicts a coding structure where only the central view employs temporal prediction. This central view is then used as a reference for spatial prediction.

2.1.2. View prediction algorithm

To perform frame prediction between consecutive frames, motion estimation is typically employed. For static cameras, different object movements are described using local motion vectors. Alternatively, in the case of moving cameras, large-scale motion can be described using global motion estimation. However, the parallax motion of objects between views depends on the object-depths. We therefore employ a view-prediction scheme that takes into account both the depth of objects and the motion of cameras. The prediction is based on an image-warping algorithm[9] that synthesizes an image as seen by the selected camera. The synthesis algorithm employs a reference texture and depth image as input data. Our prediction scheme originates from our earlier

work[11] and was also investigated in another publication[10] using wavelet image coding. Similar ideas were also adopted for dynamic scenes.[2]

3. WARPING-BASED VIEW PREDICTION

In this section, we first describe a 3D image-warping technique[9] proposed by McMillan *et al.* Second, a variant of the *Relief Texture*[3] mapping algorithm, which integrates the optics underlying real cameras, is proposed.

3.1. 3D image-warping

A single texture image and a corresponding depth image are sufficient to synthesize novel views from arbitrary positions. Let us consider a 3D point at *homogeneous* world coordinates $\boldsymbol{P_w} = (X_w, Y_w, Z_w, 1)^T$ captured by two cameras and projected onto the reference and predicted image planes at pixel positions $\boldsymbol{p_1} = (x_1, y_1, 1)^T$ and $\boldsymbol{p_2} = (x_2, y_2, 1)^T$, respectively (see Figure 4). We assume that the reference camera is located at the coordinate-system origin and looks along the Z-direction. The predicted camera location and orientation are described by its camera center $\boldsymbol{C_2}$ and the rotation matrix $\boldsymbol{R_2}$. This allows us to define the pixel positions $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ in both image planes by

$$\lambda_1 \boldsymbol{p_1} = [\boldsymbol{K_1}|\boldsymbol{0_3}]\, \boldsymbol{P_w} \;, \tag{1}$$

$$\lambda_2 \boldsymbol{p_2} = [\boldsymbol{K_2}|\boldsymbol{0_3}] \begin{bmatrix} \boldsymbol{R_2} & -\boldsymbol{R_2 C_2} \\ \boldsymbol{0_3^T} & 1 \end{bmatrix} \boldsymbol{P_w} = \boldsymbol{K_2 R_2} \begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} - \boldsymbol{K_2 R_2 C_2} \;, \tag{2}$$

where $\boldsymbol{K_1}$, $\boldsymbol{K_2}$ represent the 3×3 intrinsic parameter matrix of the corresponding cameras and λ_1, λ_2 some positive scaling factors.[13] Because the matrix K_1 is upper-triangular and $K_1(3,3) = 1$, the scaling factor λ_1 can be specified in this particular case by $\lambda_1 = Z_w$. From Equation (1), the 3D position of the original point $\boldsymbol{P_w}$ in the Euclidean domain can be written as

$$(X_w, Y_w, Z_w)^T = \boldsymbol{K_1^{-1}} \lambda_1 \boldsymbol{p_1} = \boldsymbol{K_1^{-1}} Z_w \boldsymbol{p_1}. \tag{3}$$

Finally, we obtain the predicted pixel position $\boldsymbol{p_2}$ by substituting Equation (3) into Equation (2) so that

$$\lambda_2 \boldsymbol{p_2} = \boldsymbol{K_2 R_2 K_1^{-1}} Z_w \boldsymbol{p_1} - \boldsymbol{K_2 R_2 C_2}. \tag{4}$$

Equation (4) constitutes the image warping[9] equation that enables the synthesis of the predicted view from the original reference view and its corresponding depth image.

One issue of the previously described method is that input pixels $\boldsymbol{p_1}$ of the reference view may not always be mapped to a pixel $\boldsymbol{p_2}$ at integer pixel position. A second difficulty is that multiple original pixels can be projected onto the same pixel position in the predicted view. For example, a foreground pixel can occlude a

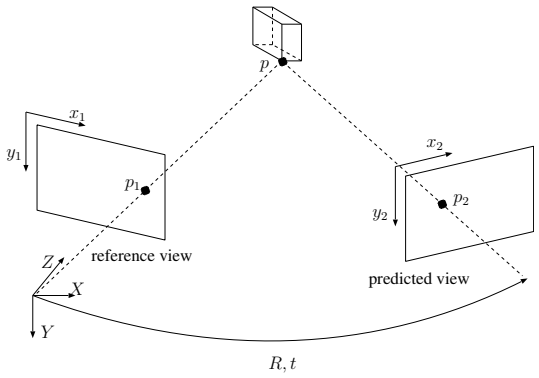


Figure 4. Two projection points $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ of a 3D point $\boldsymbol{P_w}$.

background pixel in the interpolated view, which is resulting in overlapped pixels. Additionally, some regions in the interpolated view are not visible from the original viewpoint, which results in holes in the predicted image.

To address the aforementioned issues, we propose a variant of the *relief texture* mapping technique which is adapted to the geometry of multiple views.

3.2. 3D image-warping using relief texture mapping

3.2.1. Relief texture mapping

The guiding principle of the relief texture algorithm is to factorize the 3D image-warping equation into a combination of 2D texture mapping operations. One well-know 2D texture mapping operation corresponds to a perspective projection of planar texture onto a plane defined in a 3D world. Mathematically, this projection can be defined using homogeneous coordinates by a 3×3 matrix multiplication, and corresponds to an homography transform between two images. The advantage of using such a transformation is that a hardware implementation of the function is available in most of the Graphic Processor Units (GPU). Processing time is therefore dramatically reduced.

Let us now factorize the warping function so as to obtain an homography transform in the factorization. From Equation (4), it can be written that

$$\frac{\lambda_2}{Z_w}\boldsymbol{p_2} = \boldsymbol{K_2 R_2 K_1^{-1}} \cdot (\boldsymbol{p_1} - \frac{\boldsymbol{K_1 C_2}}{Z_w}). \tag{5}$$

Analyzing the previous factorized equation, it can be observed that the first factor $\boldsymbol{K_2 R_2 K_1^{-1}}$ is equivalent to a 3×3 matrix. We thus obtain the desired homography transform between two images.

Let us now analyze the second factor of the factorized equation, i.e. $(\boldsymbol{p_1} - \frac{\boldsymbol{K_1 C_2}}{Z_w})$. This second factor projects the input pixel $\boldsymbol{p_1}$ onto an intermediate point $\boldsymbol{p_i} = (x_i, y_i, 1)^T$ that can be defined as

$$\lambda_i \boldsymbol{p_i} = \boldsymbol{p_1} - \frac{\boldsymbol{K_1 C_2}}{Z_w} \tag{6}$$

where λ_i defines an homogeneous scaling factor. It can be seen that this last operation performs the translation of the reference pixel $\boldsymbol{p_1}$ to the intermediate pixel $\boldsymbol{p_i}$. The translation vector can be expressed in homogeneous coordinates by

$$\lambda_i \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = \begin{pmatrix} x_1 - t_1 \\ y_1 - t_2 \\ 1 - t_3 \end{pmatrix} \text{ with } (t_1, t_2, t_3)^T = \frac{\boldsymbol{K_1 C_2}}{Z_w}. \tag{7}$$

Written in Euclidean coordinates, the intermediate pixel position is defined by

$$x_i = \frac{x_1 - t_1}{1 - t_3} \qquad y_i = \frac{y_1 - t_2}{1 - t_3}. \tag{8}$$

It can be seen that the previous mapping basically involves a 2D texture mapping operation, which can be further decomposed into a sequence of two 1D-transformations. In practice, these two 1D-transformations are performed first, along rows, and second, along columns. This class of warping methods is known as scanline algorithms.[14] An advantage of this supplementary decomposition is that a simpler 1D texture mapping algorithm can be employed (as opposed to 2D texture mapping algorithms). More specifically, 1D signal re-sampling can now be easily performed.

3.2.2. Signal re-sampling

We now provide details about the signal re-sampling, where the signal, in our particular case, corresponds to depth data. Let us consider two consecutive pixels in the input reference depth image (see Figure 5). Following Equation (8), both pixels are first shifted horizontally in the intermediate image. Because both pixels are mapped at sub-pixel position, it is necessary to interpolate pixel values at integer pixel positions. A similar process is subsequently repeated column-wise to obtain the intermediate image. The pseudo-code of the signal re-sampling algorithm is summarized in Algorithm 1. In this pseudo-code, the procedure "LinearInterpolation($Z_w^{prev}, Z_w, x_i^{prev}, x_i, x$)" performs a simple linear interpolation of value $Z_w^{interpolated}$ between Z_w^{prev} and Z_w at position x, where Z_w^{prev} and Z_w are defined at position x_i^{prev} and x_i.

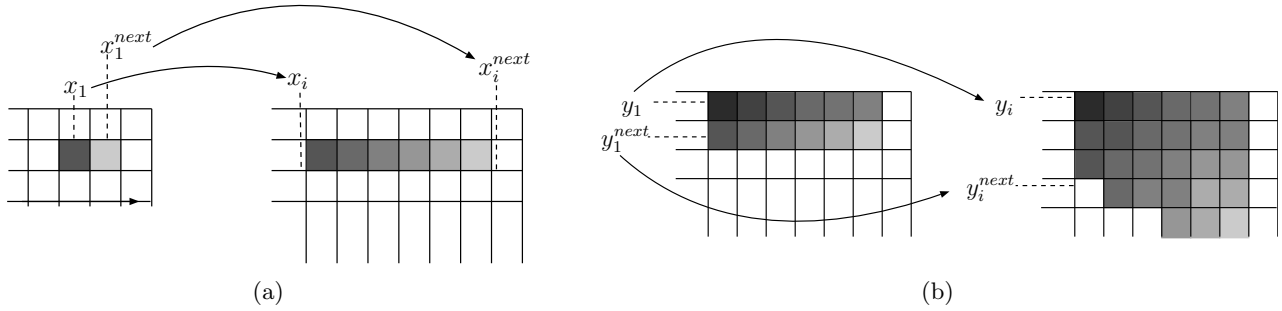


Figure 5. (a) In a first step, two consecutive pixels are horizontally shifted and resampled. (b) Next, resampled pixels are then projected into the final intermediate image by performing vertical pixel shift followed by a signal resampling procedure.

Algorithm 1 1D horizontal signal mapping and re-sampling - algorithm summary

Require: Camera parameters K_1 and C_2.
Require: Row $depth_1$ of the input depth image of width w pixels.
Require: Output buffer $depth_i$ for storing the horizontally shifted and resampled pixels.

```
procedure AskXIntermediate(x_1, Z_w)
    return x_i using Equation 8
end procedure

main
    Z_w^prev = depth_1[0]
    x_i^prev = AskXIntermediate(0, Z_w^prev)
    for (x_1 = 1; x_1 < w; x_1++) do
        Z_w = depth[x_1]
        x_i = AskXIntermediate(x_1, Z_w)
        for (x = ⌈x_i^prev⌉; x ≤ x_i; x++) do
            Z_w^interpolated = LinearInterpolation(Z_w^prev, Z_w, x_i^prev, x_i, x)
            depth_i[x] = Z_w^interpolated
        end for
        x_i^prev = x_i
        Z_w^prev = Z_w^interpolated
    end for
end main
```

3.3. Occlusion padding

For the padding of occluded depth pixels, the previously described warping algorithm ensures that occluded depth samples are interpolated between Z_W^{prev} and Z_w. However, using interpolated pixels does not always provide a sufficient prediction quality. These warping artefacts mostly occur along depth discontinuities, where depth samples are interpolated between foreground and background depth values. Therefore, we now introduce an additional simple heuristic algorithm which exploits the idea that occluded pixels belong to the background. To interpolate depth samples along discontinuities, the neighboring pixel value, Z_w^{prev} or Z_w, that has the highest depth value should be taken into account. In this case, the linear interpolation function $LinearInterpolation()$ is not executed and a simple depth-pixel copy is performed $Z_w^{interpolated} = max(Z_w^{prev}, Z_w)$. In our implementation, we assume that a depth-discontinuity is detected if $|Z_w^{prev} - Z_w| > 20$. Although this technique is based on a very simple heuristic, experiments have revealed that such a depth-occlusion handling provides important improvements for rendering quality.

Note that Algorithm 1 uniquely computes the destination position for each input pixel in the predicted view,

but that it is not possible to *invert* the projection in order to find a corresponding pixel in the input image for each pixel in the new view. This problem becomes mostly apparent in overlapping regions, where two pixels are warped onto the same position in the rendered view.

3.4. Occlusion-compatible scanning order

During the warping procedure, multiple original pixels can be projected onto the same pixel position in the interpolated view. For example, a foreground pixel can occlude a background pixel in the interpolated view, which results in overlapped pixels. This problem, known as the *visibility problem* in the computer-graphics community, can be addressed using two techniques.
First, to determine the visibility of each pixel in the rendered view, a practical method involves the usage of a *Z-buffer*. A Z-buffer is a memory that stores the depth of each pixel in the rendered view. When two pixels are warped at the same position in the rendered image, a depth comparison of the 2 pixels is performed and the foreground pixel with smallest depth value is selected for rendering. However, this technique involves the usage of a memory buffer and a depth comparison for each warped pixel.
Second, an alternative technique corresponds to scanning the original view such that eventually occluded background pixels are rendered prior to the occluding foreground pixels. This is the *occlusion-compatible scanning order*[9] algorithm for which we now derive that a proper scanning can be made.

Let us consider two true 3D scene points $\boldsymbol{X_1}$ and $\boldsymbol{X_2}$, which are projected onto a target image (predicted view) at the same pixel position x, and onto the reference view at pixel positions $\boldsymbol{x'_1}$ and $\boldsymbol{x'_2}$ (see Figure 6). The projection matrices of the reference camera is denoted P'. We additionally assume that the 3D points $\boldsymbol{X_1}$ and $\boldsymbol{X_2}$, seen from the target camera C, correspond to background and foreground objects, respectively, such that

$$|X_2 - C| < |X_1 - C|. \tag{9}$$

To perform an occlusion-compatible scanning order, it is necessary to warp the first the 3D point $\boldsymbol{X_1}$ and second $\boldsymbol{X_2}$. Projecting the 3D points $\boldsymbol{X_1}$ and $\boldsymbol{X_2}$ in the reference view, it can derived from Equation (9) that

$$|X_2P' - CP'| < |X_1P' - CP'|,$$

which is equivalent to

$$|x'_2 - e'| < |x'_1 - e'|, \tag{10}$$

where e' corresponds to the epipole in the reference view.

Therefore, in order to ensure that background pixels do not overwrite foreground pixels after the projection, we scan image lines in a direction that decreases the distance between the 3D point and the target camera. To perform such a scanning in the original image, this 3D point should be projected onto this reference image. Therefore, to perform the warping of background pixels x_1 prior to foreground pixels x_2, it is sufficient to scan the source image from from the border of the image toward the epipole e'.

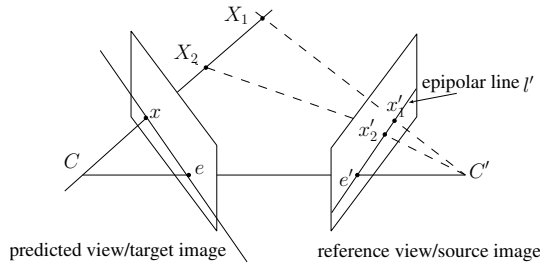


Figure 6. To perform the warping of background x'_1 prior to foreground x'_2, the reference image is scanned from the border toward the epipole e'.

3.5. Summary of the relief texture image warping algorithm

As a summary, the warping of a depth image is performed as follows.

- Step 1: perform warping of reference depth samples along horizontal scanlines as described by Algorithm 1. Input depth samples are scanned from the border of the image toward the epipole e'.
- Step 2: perform warping of the (horizontally-warped) depth image along vertical scanlines (see Algorithm 1). Similarly, depth samples are scanned from the border of the image toward the epipole e'.
- Step 3: compute the planar texture projection of the intermediate image using the homography transform defined by $\boldsymbol{K_2 R_2 K_1^{-1}}$.

4. HYBRID SPATIO-TEMPORAL H.264 ENCODER

We now propose a novel H.264 architecture dedicated to multiview depth coding that employs image-warping as a predictive-coding algorithm.

To exploit temporal redundancy between frames, H.264 encoders employ motion estimation so as to perform predictive coding of consecutive frames. Similarly, we have shown that image-warping can be employed as a spatial prediction technique. To integrate both warping-based prediction and block-based motion prediction, we propose to simply insert the warping block in the H.264 architecture and perform the prediction adaptively, depending on the frame type.

One disadvantage of such a multiview encoder is that the prediction error is not minimized, thus leading to a lower compression efficiency. An alternative technique is to employ a combination of two predictors: (a) the warping-based predictor followed by (b) the block-based motion predictor (see Figure 7).
The system concept is as follows. First, we provide an approximation of the depth image using image-warping and, second, we refine the warping-based prediction using block-based motion prediction. In the refinement stage, the search of matching blocks is performed in a region of limited size, e.g. 16×16 pixels. For comparison, a block-based motion prediction between two different views typically involves a region-search of a size as large as 64×64. Figure 7 shows an overview of the described coding architecture.

The advantages of using an H.264 encoder are two-fold. First, re-using a standardized encoder provides forms of backward compatibility with the H.264 compression functions (CABAC, etc.). Second, because the H.264 standard enables that each macroblock can be encoded using different coding modes, occluded regions

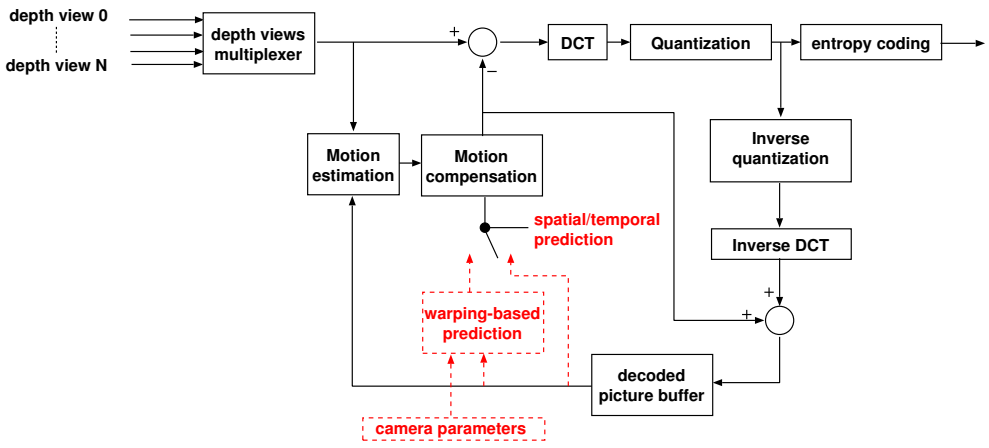


Figure 7. Architecture of an hybrid spatio-temporal H.264 encoder that employs an adaptive prediction using only a temporal prediction or a combination of temporal and spatial prediction algorithms. Novel block functions are highlighted by dashed lines.

in the depth image can be efficiently compressed. More specifically, occluded depth pixels cannot always be predicted with sufficient accuracy. In this case, the algorithm encodes an occluded macroblock in *intra-mode*. Alternatively, when the prediction accuracy of occluded pixels is sufficient, the macroblock is encoded in *inter-mode*. Thus, the H.264 standard offers sufficient flexibility in coding modes to match them with the various prediction accuracies of our algorithm. Consequently, occlusions can be coded in a rate-distortion optimal way.

5. EXPERIMENTAL RESULTS

For evaluating the performance of the coding algorithm, experiments were carried out using the "Ballet" and "Breakdancers" depth sequences using 8 views and 3 views per sequence.
The presented experiments investigate the impact of depth-prediction across multiple views. For each presented rate-distortion curve, we perform the compression of depth images under three different conditions.

1. No prediction of depth images is carried out in the compression system, for the "simulcast" case (see Figure 8(a))
2. A prediction of depth images across multiple views is carried out using the H.264 block-based motion-compensation prediction, denoted "block-based motion-compensation prediction" (see Figure 8(b)).
3. A prediction of depth images is carried out using the image warping proposed in this paper, denoted "image-warping based prediction" (see Figure 8(c)).

Since our work has focused on the spatial prediction of depth across multiple views, only the spatial redundancy was exploited for compression. Therefore, *no motion compensation* is employed in the temporal between consecutive time intervals. This ensures that the motion-compensation prediction scheme does not interfere with the evaluation of the spatial prediction algorithm. Practically, to avoid motion prediction in the temporal direction, a periodic intra-frame coded picture is inserted at specific place, within each time interval.

For simulcast coding, the rate-distortion curves were generated using a standard H.264 encoder. The multiview depth sequences were multiplexed so as to provide a single depth video sequence. To perform independent coding, each depth image is then intra-frame coded.
To measure the efficiency of the block-based motion-compensation prediction, the multiview depth sequences were similarly multiplexed and compressed by a standard H.264 encoder. In this case, an I-frame is then inserted for each time interval and neighboring views are compressed as a P-frame. Because the parallax motion of pixels is large, the motion-search range is as high as 256×256 pixels.
Finally, to measure the efficiency of our warping-based predictive-coding algorithm, we have implemented and inserted our warping-based prediction algorithm in an H.264 encoder. As described in Section 5, the warping-based prediction is followed by a prediction-error minimization. In our implementation, this refinement-minimization step is carried out by the H.264 block-based motion-compensation over a region of 16×16 pixels.
Because the prediction structures shown by Figure 8(b) and Figure 8(c)) employ a different reference frames, it can be argued that comparison of the two prediction schemes is not appropriate. More specifically, observing the prediction structure of the block-based motion-compensation prediction, it can be seen that each frame is predicted from the nearest-neighboring view, leading to recursive dependencies. At the opposite, the warping-based

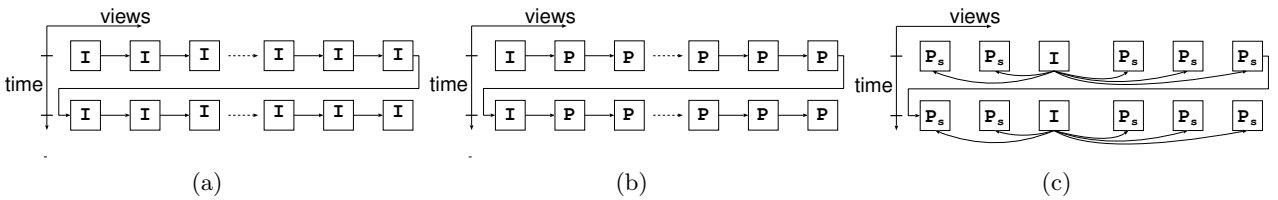


Figure 8. (a) An independent coding of each depth image is performed (simulcast coding). (b) Block-based motion-compensation, and (c) image-warping prediction is employed to predict depth images across multiple views. Frames denoted as P_s are predicted using depth-image warping prediction and coded in inter-frame mode.

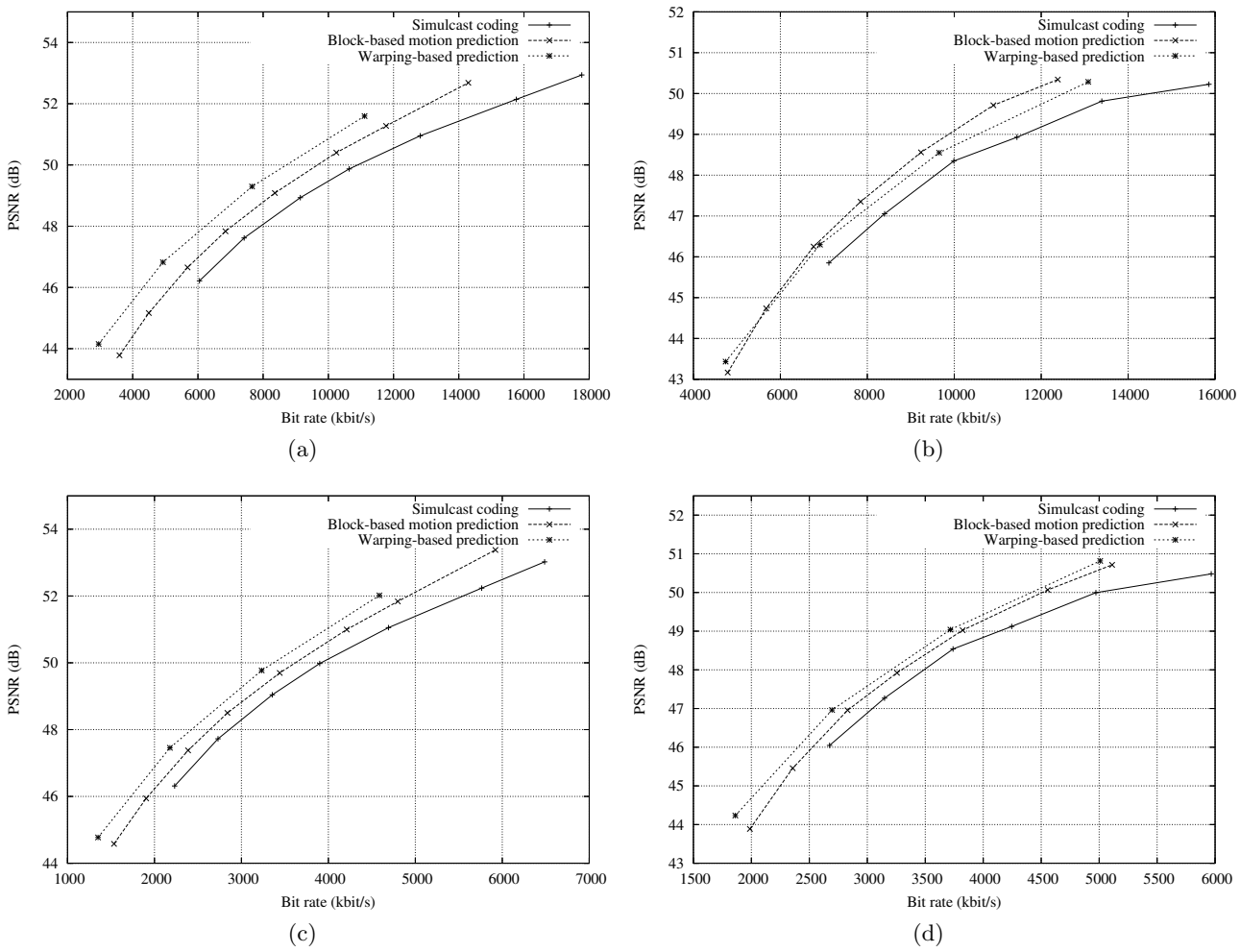


Figure 9. Rate-distortion curves for encoding 8 views of (a) the "Breakdancers" and (b) the "Ballet" depth sequences and 3 views of (c) the "Breakdancers" and (d) the "Ballet" depth sequences.

prediction algorithm employs one central reference view, providing random access. Typically, this approach is obtained at the expense of lower coding efficiency. However, although the proposed prediction algorithm employs only one reference frame, we show in the experiments that such a coding structure combined with the proposed warping-based prediction yields a high coding performance.

For coding experiments, we have employed the open-source H.264 encoder x264.[15] The arithmetic coding algorithm CABAC was enabled for all experiments. For each sequence, the frame rate is 15 frames per second. Thus, the transmission of 8 and 3 views corresponds to a frame rate of 120 and 45 frames per second, respectively. Such a high frame-rate explains the magnitude of the presented bit-rates in Figure 8 ranging from approximatively 2 $Mbit/s$ to 10 $Mbit/s$.

Let us now discuss the obtained rate-distortion curves of Figure 9(a) and Figure 9(b) obtained for 8 views and Figure 9(c) and Figure 9(d) obtained for 3 views for the "Breakdancers" and "Ballet" depth sequences. First, it can be observed that the proposed warping-based prediction algorithm consistently outperforms simulcast coding and block-based motion-prediction scheme. For example, considering Figure 9(a), it can be seen that our proposed warping-based prediction algorithm yields a quality improvement of up to 1.8 dB and 0.9 dB at 8 $Mbit/s$ over simulcast and the block-based motion prediction algorithm, respectively. However, we have noted one case (see Figure 9(b)) that shows no improved coding efficiency when using the proposed prediction algorithm.

Considering the characteristics of this depth sequence, we have noted that the motion between camera-views is high, thereby leading to large occluded regions. In this particular case, the proposed warping-based coding algorithm does not provide a sufficiently accurate prediction for obtaining an improved compression efficiency. Althought no coding improvement can be reported for this particular case, the proposed algorithm still enables random access to different views.

6. CONCLUSIONS

We have presented a new algorithm for the predictive coding of depth images that allows random access to different views. The concept is based on using a single reference depth image from which neighboring depth maps are predicted. The prediction is based on a variant of the relief texture mapping algorithm that incorporates the intrinsic and extrinsic camera parameters. Next, we have integrated the prediction scheme into an H.264 encoder, such that motion-compensation prediction is combined with the warping-based prediction. Experimental results have shown that the warping-based predictive-coding algorithm can improve the resulting depth image quality by up to 1.8 *dB* and 1 *dB* when compared to a simulcast compression and H.264 block-based motion prediction, respectively. Therefore, the presented technique demonstrates that that predictive-coding of depth images can provide a substantial compression improvement of multiple depth-images while providing random-access to individual frames for real-time rendering.

REFERENCES

1. H.-Y. Shum and S. B. Kang, "Review of image-based rendering techniques," in *Proceedings of SPIE, Visual Communications and Image Processing*, **4067**, pp. 2–13, May 2000.
2. C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-quality video view interpolation using a layered representation," *ACM Transactions on Graphics* **23**(3), pp. 600–608, 2004.
3. M. M. Oliveira, *Relief Texture Mapping*, Ph.D. Dissertation. UNC Computer Science, March 2000.
4. D. Farin, Y. Morvan, and P. H. N. de With, "View interpolation along a chain of weakly calibrated cameras," in *IEEE Workshop on Content Generation and Coding for 3D-Television*, June 2006.
5. S. M. Seitz and C. R. Dyer, "View morphing," in *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 21–30, ACM Press, (New York, NY, USA), 1996.
6. "MPEG document, Survey of algorithms used for Multi-View Video Coding (MVC)." ISO/IEC JTC1/SC29/WG11, MPEG2005/N6909, January 2005.
7. P. Merkle, K. Mueller, A. Smolic, and T. Wiegand, "Efficient compression of multi-view video exploiting inter-view dependencies based on H.264/MPEG4-AVC," in *International Conference on Multimedia and Expo, ICME 2006*, **1**, pp. 1717–1720, (Toronto, Canada), 2006.
8. U. Fecker and A. Kaup, "H.264/AVC compatible coding of dynamic light fields using transposed picture ordering," in *13th European Signal Processing Conference*, **1**, (Antalya, Turkey), September 2005.
9. L. McMillan, *An Image-Based Approach to Three-Dimensional Computer Graphics*, University of North Carolina, April 1997.
10. M. Magnor and B. Girod, "Data compression for light field rendering," *IEEE Transaction Circuits and Systems for Video Technology* **10**(3), pp. 338–343, 2000.
11. Y. Morvan, D. Farin, and P. H. N. de With, "Design considerations for view interpolation in a 3D video coding framework," in *27th Symposium on Information Theory in the Benelux*, 2006.
12. E. Martinian, A. Behrens, J. Xin, and Anthony, "View synthesis for multiview video compression," in *Picture Coding Symposium*, May 2006.
13. R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2004.
14. G. Wolberg, *Digital Image Warping*, IEEE Computer Society Press, July 1990.
15. x264 a free H264/AVC encoder, *http://developers.videolan.org/x264.html*, last visited: December 2006.