Multimed Tools Appl (2016) 75:1899–1921
DOI 10.1007/s11042-014-2379-2

Hole filling using multiple frames and iterative texture synthesis with illumination compensation

Ting-Lan Lin · Uday Singh Thakur · Chi-Chan Chou · Shih-Lun Chen

Received: 7 May 2014 / Revised: 21 September 2014 / Accepted: 7 November 2014 /
Published online: 21 November 2014
© Springer Science+Business Media New York 2014

Abstract In this paper, we propose a novel freeview generation system. We first discuss the problem of misalignment of the edges between color frame and depth frame, and propose an optimization method to solve it. Then an original frame is separated into two parts, depending on the sizes of the resulting dis-occluded areas in the virtual view. Illumination changes between current frame and the information from previous frames are modeled and compensated. In the virtual view, the missing pixels are recovered by the proposed inpainting method. Using depth information, we compute the priority of the recovering order among the missing pixels. The recovering process accounts for pixel consistency, and the process iterates between virtual color frame and depth frame. Experimental results show that, compared with a state-of-the-art work, the proposed method has better subjective and objective performances; the PSNR gain can be up to 2.9575 dB.

Keywords Freeview generation · Depth-image-based rendering (DIBR) · Depth boundary rectification · Inpainting algorithm · Depth discontinuity · Illumination compensation

1 Introduction

Stereoscopic video industry is growing rapidly. The viewing experience in 3D is much closer to reality than traditional 2D video can provide, thanks to depth information, which creates perception of different distances for different objects in an image. 3D movie productions and

T.-L. Lin (✉) · C.-C. Chou · S.-L. Chen
Department of Electronic Engineering, Chung Yuan Christian University, 200 Zhongbei Rd., Zhongli City, Taoyuan County 320, Taiwan, Republic of China
e-mail: tinglan@cycu.edu.tw

C.-C. Chou
e-mail: g10276011@cycu.edu.tw

S.-L. Chen
e-mail: chrischen@cycu.edu.tw

U. S. Thakur
Institut für Nachrichtentechnik, RWTH Aachen University, Aachen, Germany
e-mail: Thakur@ient.rwth-aachen.de

cinemas have been growing these years, and are expected to replace 2D filming. Other than the application in single-viewpoint 3D video, depth information can also be very helpful in multiview video. Multiview videos are obtained by the same theme's being filmed with more than one camera at the same time. The relative positions among the multiple cameras are very important, and carefully designed. Multiview videos allow viewers to be able to watch from a desired viewpoint and provide a realistic perception of the theme. With this philosophy, it is nature to understand that "the more the numbers of views are, the more satisfied the viewer is." In fact, there are multiview auto stereoscopic displays supporting 9 or more different views at the same time. According to the angle at which the viewer watches the display, the viewer will see one of the view pairs and perceive the 3D effect.

However, more views mean more storage or transmission bandwidth. Especially for the video streaming application where the network bandwidth is limited, the end viewers will have difficulty watching a multiview video. The transmission costs do not justify the additional value [9]. It is inefficient to transmit a view that differs only slightly from the primary view. Multiview video plus depth (MVD) format can avoid this inefficiency. MVD format allows a technology called *free-view synthesis*, which lets viewers switch to any view they want, with only a limited number of views available. With this technology, the video transmitter can transmit the depth and color information of just, for example, views 2, 4, 6, and 8, and in the video receiver the technology can generate views 3, 5, and 7 by intermediate view interpolation [6].

The view synthesis technology discussed in this paper is based on depth-image-based rendering (DIBR). One major challenge of DIBR is how to fill the dis-occluded region (hole) in the virtual view [4]. The dis-occlusion in synthesized view is a natural consequence of DIBR. High-depth discontinuity in the depth map introduces great dis-occlusion in comparison with low-depth discontinuities, and high-depth discontinuity is more likely to produce artifacts along object boundaries [9]. Filling such dis-occlusions perfectly is difficult, because the true information may not be captured by the original cameras. Therefore, the key to the success of the multiview video transmission technology is how efficiently the dis-occlusions can be filled. *Interpolation* using two views can fill most of the generated holes by blending them together [3]. However, in special cases—for example, when generating view 1 from view 2 and similarly view 9 from view 8, where view 2 and view 8 are the corner-transmitted views—we do not have the advantage of using two views for interpolation, and so we need a proper view *extrapolation* algorithm for such type of cases in which only information available for missing pixels is rather limited. Many approaches have been proposed to fill the dis-occlusions. For example, the work in [4] interpolates the dis-occluded regions, and the work in [13] makes smooth the depth maps that produce geometric distortions. Other approaches, such as [10], perform intermediate view interpolation that separates unreliable image regions along depth discontinuities from reliable image regions. The two regions are then interpolated separately and fused. View blending has also been used for hole-filling [6]. This approach requires more than one original view for blending, and thus requires double computational and capturing resources [11]. Inpainting techniques have been proposed for filling the dis-occlusion. Image inpainting is a technique originally introduced in [1] to recover missing texture in an image. Inpainting techniques in [5, 12] use joint disparity and texture segmentation.

Most of the approaches discussed so far render the new images using only the information from frames of different views at the same time instance, without using the temporal correlation among frames. There are papers handling temporal consistency in uncovered regions. For the work in [8], each frame is divided into dynamic foreground and static background. This type of representation helps recover a virtual view without dis-occlusion. The paper does not, however, discuss how the method works when the foreground is static or where the missing information cannot be found in neighboring frames. Also, if any illumination change happens for a specific duration, the method does not provide a solution to compensate the illumination from the information in another duration of time. A recent work by Ndjiki-Nya et al. [7] uses a *background sprite* (*BG sprite*) to store the useful information from neighboring frames in order to fill the dis-occluded areas. They introduced a *covariance cloning* method to balance the illumination change, and finally used *texture synthesis* to fill the remaining dis-occlusion. The system diagram of the work in [7] is shown in Fig. 1(a).

The work in [7] has certain drawbacks:

1. The *depth boundary errors*, because of the mismatch of the edges of objects between color and the depth frames, are not taken care of; this results in wrongly rendered objects after DIBR.
2. The interframe information to build the background pixels is used *after* the DIBR, as can be seen in Fig 1(a); the pixel information after DIBR is just an estimate, and taking it into consideration to build the background leads to error accumulation. This procedure also increases complexity and memory when the end user switches views; every new virtual view demanded calls for a database storing the past information corresponding to the new virtual view to be created and updated.

In the proposed paper, we address the above problems of the state-of-the-art work by Ndjiki-Nya et al. [7]. We aim to propose a novel system to improve each of its functionalities. The system diagram of the proposed work is shown in Fig. 1(b). The novelties of the proposed paper are:

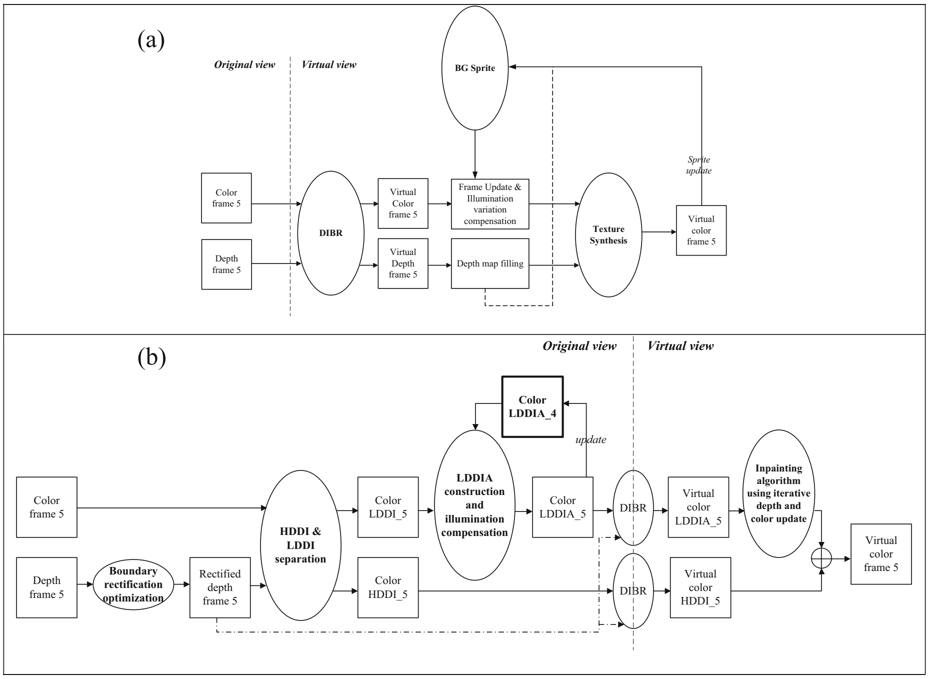


Fig. 1 The system diagram of **a** Ndjiki-Nya et al.'s work [7] and **b** the proposed work. This example is to generate the virtual view in frame 5

1. Depth boundary rectification optimization: The proposed method rectifies depth boundary mismatch with respect to its color image using our optimization algorithm. This prevents the formation of visual artifacts in the synthesized view, and enhances the visual quality of the virtual view.
2. HDDI (high-depth discontinuity information) and LDDI (low-depth discontinuity information) separation: A novel idea for separating two distinct regions in the original view based on depth discontinuity is presented. The two regions represent different difficulties in hole-filling the resulting holes in the virtual view. The two regions have the notions of foreground and background objects, but are defined differently. They are separately processed, DIBRed, and fused together in the final step.
3. LDDIA (low-depth discontinuity information accumulator) construction: This technology is used to store processed information from past frames, and to be used to perform efficient hole-filling for the current LDDI frame. Note that the proposed use of the past frame is before the DIBR, and not after the DIBR as done in [7]. The advantages of doing things this way is to build a database frame containing the past information using the true pixels, instead of rendered estimated pixels in the virtual view, to prevent error accumulation. Also, if the view-switching events occur, we need to maintain the database frame only for the current view; this saves the memory and complexity. Finally, the LDDIA update favors more the pixels in the current frame over the past frames, to make the database more current.
4. LDDIA illumination compensation: In the update process of the LDDIA, we use 2D linear regression to form an illumination difference plane using the spatially neighboring pixels of the hole in the current frame and the database frame. The found mathematical plane model can then be used to compensate the light differences in different frames at different times.
5. Inpainting algorithm using iterative depth and color update: After finding the virtual LDDIA, we need to perform texture synthesis to fill the holes as much as possible. We propose an inpainting algorithm to process the virtual color and the virtual depth frames iteratively, based on the proposed *priority computation using depth information*, *patch matching scheme*, *and efficient hole-filling for the virtual depth*.

Based on the results from the experimental section, the overall system of the proposed method is better than the work in [7] subjectively, and objectively by up to 2.9575 dB in PSNR.

The rest of the paper is organized as follows. In section 2, we discuss an optimization scheme for rectifying the depth boundary. In section 3, we introduce the ideas of HDDI, LDDI, and LDDIA, and also estimate illumination compensation. In section 4, we propose an inpainting algorithm using iterative depth and color update. In section 5, we demonstrate and discuss experimental results. Section 6 concludes the paper.

2 Depth boundary rectification optimization

Depth discontinuities give rise to dis-occlusions in the synthesized view, a natural consequence of virtual view generation. If the object boundary in the depth map and that in the color image are not aligned, the virtual view will have errors. In this section, we first explain the importance of aligning the edges of the color frame and the depth frame, and then we introduce an optimization scheme for rectifying the misaligned depth frame.

2.1 Misalignment of the color frame and the depth frame

The dis-occlusion occurs through the change of viewpoint that exposes new areas from different angles in the process of DIBR. If the object edges between the color and the depth frames are perfectly aligned, the dis-occlusion in the virtual view can separate different objects with different depths perfectly. Because of the limitation of capturing the color and depth frames, however, most of the time they are not perfectly aligned. So different objects are not separated ideally in the virtual frame, as shown the circled area in Fig. 2(a). In Fig. 2(a), the wall and the person's hairs should have different depth values, and should be separated perfectly in the virtual view. But because the edges are misaligned between the color and depth frames, the wall and the hairs are not separated perfectly; some hairs "stay" with the wall, since that portion of the hairs is mistakenly given the depth values of the adjacent object, the wall. A zoom-in version of Fig. 2(a) is shown in Fig. 2(c). This type of error visually degrades the results of the virtual view and provides an incorrect initial starting point for the later hole-filling/inpainting work; therefore it needs to be corrected.

2.2 Boundary rectification optimization algorithm

As discussed, the misalignment between the edges of the color and depth frames introduces visual artifacts in the virtual view. In this section we propose a boundary rectification optimization method to solve the problem.

The situation of the misalignment is that the edges of the color and the depth frames are not matched. Here, we put more confidence in the color frame, aiming to match the edges of the depth frame to those of the color frame. The procedures are as follows:

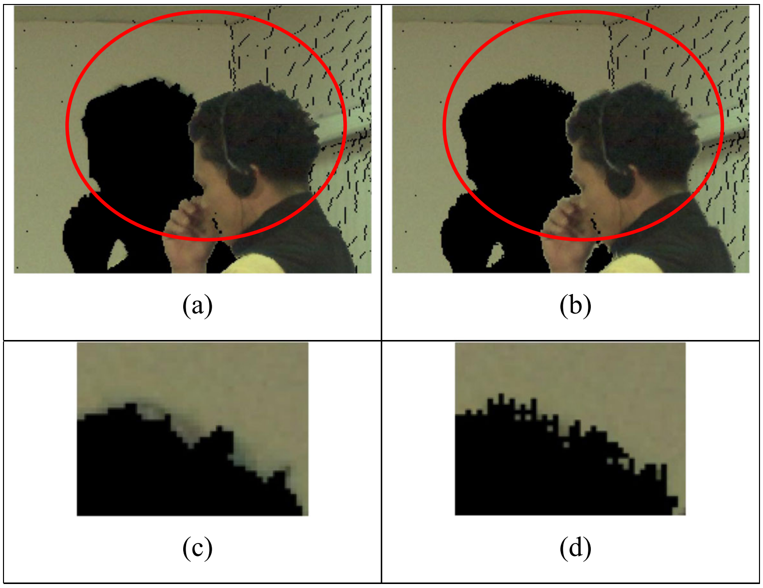


Fig. 2 The virtual view after DIBR with **a** the original depth frame (misaligned with color frame) and **b** the depth frame processed by the proposed optimization method for rectifying the depth boundary. Subfigure (**c**) is a zoom-in image of (**a**), and (**d**) is a zoom-in image of (**b**)

1. Our algorithm first uses canny edge detection on the depth frame to identify the depth edges as initial starting points.
2. The edges detected by the canny method are not always the locations we are looking for. Usually, the adjacent depth values are smooth if they are in the same object, and the adjacent depth values change drastically if they are in two objects of different depths. Therefore, to locate the object boundary more accurately, we check the depth differences in the neighborhood of the canny edges. For each canny edge, we compute the sum of the absolute differences between the canny edge and its horizontal 1-pixel neighbors. If the sum is greater than a threshold, we keep the canny edge as the edge to be rectified; otherwise, we exclude the canny edge from further processing.
3. For the depth edge to be rectified, we denote it by B_0, as shown in Fig. 3. We aim to search the color frame in the neighborhood of the B_0, as shown in Fig. 3, to see whether the B_0 is also the edge in the color frame. If it is not, we aim to find the closest color edge in the neighborhood of the B_0. We mark the closest color edge to be the modified edge B, as shown in Fig. 3.

To be specific, the neighborhood of the B_0 is defined to be a ± 5-pixel wide neighborhood region centered at $B_{0,}$ since usually the misalignment errors in the virtual view are, depending on the quality of datasets used, 2 to 5 pixels wide or more. The color edge is found by a sharp

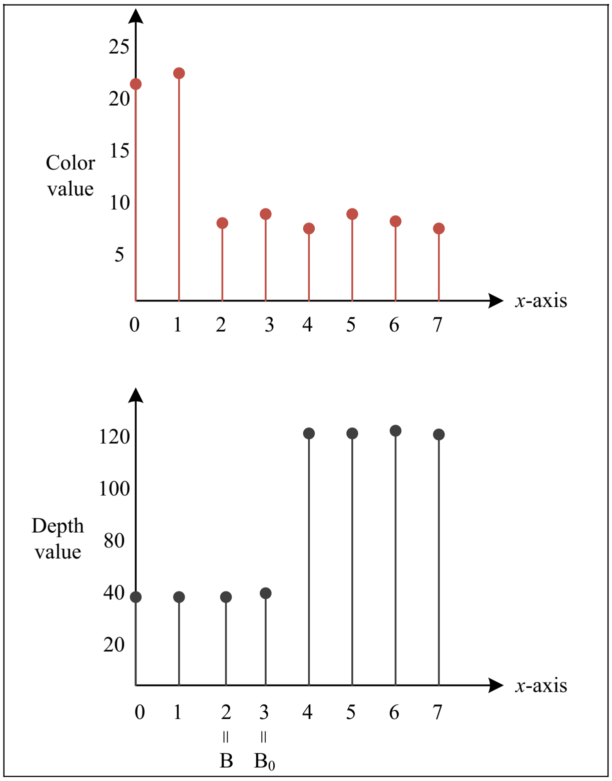


Fig. 3 Illustration of the depth boundary rectification optimization method. The color and the depth frames are positioned in the same spatial location. B_0 is the qualified initial edge detected by the canny algorithm in the depth frame, and B is the finally rectified edge point

change in adjacent color pixel values; a sharp change is defined when the absolute difference between the adjacent color values is greater than γ. *Therefore, finding the closest sharp change location B in color frame in the neighborhood of the B_0 can be formulated as the following optimization problem:*

$$B = \arg\min_{x \in \{(B_0-5), \sim, (B_0+5)\}} |x - B_0| \\ s.t. \ |color(x) - color(x-1)| > \gamma$$

After B has been found, the depth values inclusively between the locations of B_0 and B are modified: they are replaced by the depth values at location B_0+1. In the example in Fig. 3, the B_0 is 3, and the optimal edge B found by the proposed optimization problem is 2, and the depth values at x=B =2 and x=B_0=3 are modified by the depth value at x =4.

The proposed depth rectification optimization method works well. The virtual color frame in Fig. 2(b) is rendered by the rectified depth frame, and the dis-occlusion separates the wall and the hairs much better than in Fig. 2(a); we provide a much better starting point for the later hole-filling/texture synthesis process. A zoom-in version of Fig. 2(b) is shown in Fig. 2(d). This is also very important for the processing discussed in the next sections. The depth frames referred to thereafter are the rectified depth frames obtained by the method described in this section.

3 LDDIA construction & illumination compensation

In this section, we discuss a novel algorithm for collecting useful information from multiple frames. The collected information is later used to fill in the missing information. This section also discusses balancing the lighting changes over the pixel information among different frames.

3.1 Defining the HDDI and LDDI

Since a video sequence's foreground is dynamic, some regions originally occluded in a particular frame at any time instance may be exposed in the causal frames at other time instances. We use this characteristic to help fill the missing information.

The missing portions (holes) in a virtual frame are usually the "background" part, since they are dis-occluded to the viewer during the view change. Therefore, the work [7] builds a "universal background" frame to be used in the future to fill the holes in other frames. However, it is difficult to explicitly define and separate the foreground and the background. In our approach, we do not perform explicit background extraction. Instead, we classify the "background" and "foreground" in the original view by the size of "gap" (hole) between pixels in the virtual view. For example, we consider a current pixel located at (3,4) in the original view to be A, and its (previously) horizontally neighboring pixel located at (3,3) in the original view to be B. The two pixels have new locations in the virtual view after DIBR, defined as C and D, respectively. If the geometric distance (gap, hole) between C and D is small, i.e., they are close in location, it implies two meanings:

1) Pixels A and B in the original view are of similar depth range. In other word, pixels A and B are either both foreground, or both background.
2) The hole-filling work in the virtual view is relatively easy since the gap (hole) between C and D in the virtual view is small.

For example, Fig. 4(a) and (b) are used to perform DIBR to generate a virtual view, Fig. 4(c). On one hand, as can be seen, the sizes of the dis-occluded area in the virtual view are strongly related to the difference between depth pixels around the corresponding locations in the original view. As Fig. 4(c) shows, the upper part has a larger dis-occluded area, due to the larger depth difference in the corresponding area in Fig. 4(b), and the lower part has a smaller one, due to smaller depth difference in the corresponding area. Therefore, the size of the hole can help determine whether the neighboring pixels in the original view are of the same depth. On the other hand, in Fig. 4(c), the dis-occluded areas are of different sizes (in width), and it is known that a larger dis-occluded area is harder to be filled properly.

Hence, the size of the hole in the virtual view induced by the pixel in the original view is an important indication of whether it is a foreground pixel in the original view, and an important indication of how difficult the induced hole can be inpainted. The proposed algorithm therefore differentiates two distinctive regions in an original frame, using the size of the dis-occlusion that a specific pixel will introduce in the virtual frame. In this paper, we define the pixels in the original view that will produce dis-occlusion that is more than 5-pixels wide from the previous available pixel (in the left to right order) to be HDDI (high-depth discontinuity information), and the rest of the region to be LDDI (low-depth discontinuity information). An example of HDDI can be seen in Fig. 4(a). We then perform HDDI removal to obtain an LDDI image. An example can be seen in Fig. 7, in which Fig. 7(a) is the original color frame, and Fig. 7(c) its LDDI color frame.

Note that the LDDI has the concept of "background information" (both have the idea of low-depth changes), and HDDI "foreground information" (high-depth changes), but the two are not the same. The ideas of background and foreground are hard to define, especially when

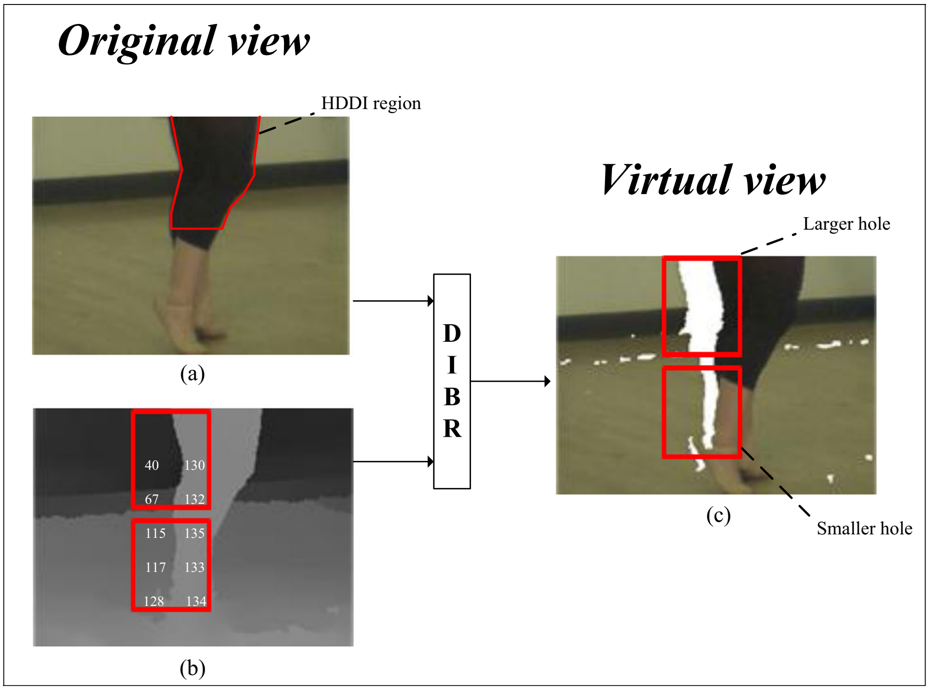


Fig. 4 Illustration of how the hole sizes in the virtual view depend on the depth differences between adjacent pixels in the original view

there a view contains objects at multiple depth levels. In the proposed work, we do not need to estimate the foreground or background, as has been done in [7]. Instead, we are concerned only with the difficulty of a particular dis-occluded area to be hole-filled, differentiated in our work by the LDDI and HDDI. The proposed method is novel and simple, whereas [7] uses complicated techniques for object removal.

3.2 LDDIA initialization

After obtaining the LDDI of a particular frame, we aim to integrate this information to update a database frame, formed by all past LDDI frames. This database frame is denoted as the Low-depth Discontinuity Information Accumulator (LDDIA). The LDDI will be further processed using the old LDDIA to become a new LDDIA. This is similar to the idea of building and updating a universal background as in [7], but the proposed method differs in concept and approach.

An example for our approach is shown in Fig. 5. The current frame that is processed is frame number 5, and its LDDI is obtained, denoted as LDDI_5. The area B in the LDDI_5 is the hole area resulted from the HDDI (HDDI_5) removal, and we want to fill into the hole by the information from a database frame, LDDIA_4 (considering the information from frame 1 to 4), by copying its collocated portion A. The resulting frame is called Ini_LDDIA_5, an initial estimate of the LDDIA_5. In next section, this Ini_LDDIA_5 will be modified to deal with the illumination differences between the current LDDI frame and the database frame LDDIA.

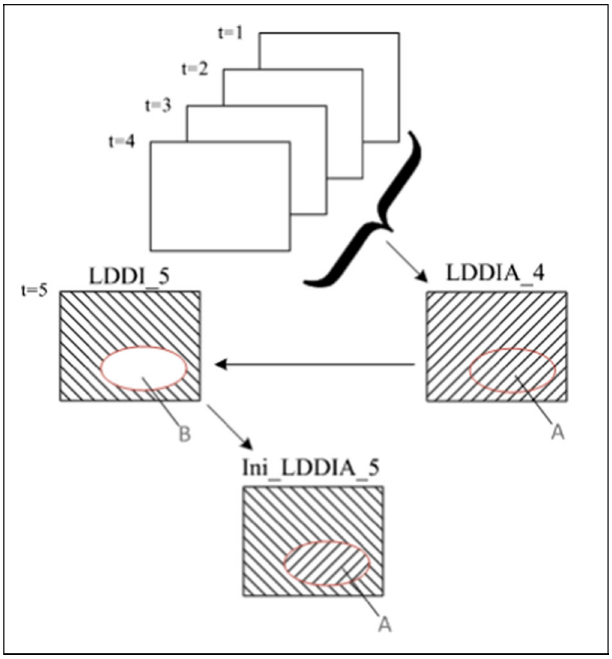


Fig. 5 The illustration of the initialization of LDDIA update. LDDIA_4 is the LDDIA considering the first 4 frames. LDDI_5 is the LDDI of frame 5. LDDI_5 and LDDIA_4 are used to obtain the initial version of LDDIA_5, denoted by Ini_LDDIA_5

As can be seen in this procedure and in Fig. 5, the major portion in the updated initial LDDI (e.g., Ini_LDDI_5) comes from the information in the current frame (LDDI_5), not from the database frame LDDIA (e.g., LDDIA_4). This is because we want to keep the latest information as much as possible to make the next LDDIA (e.g., LDDIA_5) up to date.

3.3 LDDIA illumination compensation

The LDDIA update considers the information from all previous frames, and the illumination change over that time should be accounted for. For example, in the previous subsection, in Fig. 5, portion A in LDDIA_4 can be of the information from a frame that is long time ago, and thus have a different illumination level than the current frame 5. Therefore, it is possible that in Ini_LDDIA_5 the illumination level of portion A is inconsistent with the rest of the part in Ini_LDDIA_5, introducing visual artifacts. This subsection aims to solve this problem.

The process is explained by the example in Fig. 6. We define the 4-pixel-wide neighborhood pixels of portion A in LDDIA_4 as region C, and define the 4-pixel-wide neighborhood pixels of portion B in LDDI_5 as region D. Note that the illumination condition of portion C should be very close to that of the portion A, and the illumination condition of portion D should be very close to that of portion B. Our goal is to modify the illumination levels in the portion A, to be consistent with the LDDI_5, using the information of portions C and D.

Assume that the illumination *differences* between LDDIA_4 and LDDI_5 are linearly distributed according to the spatial pixel locations (x_i,y_i) in a frame. We aim to determine this 2D illumination *difference* plane model from the information in portions C and D. The number of pixels in C and in D is the same, assumed to be N, and the corresponding pixels have the

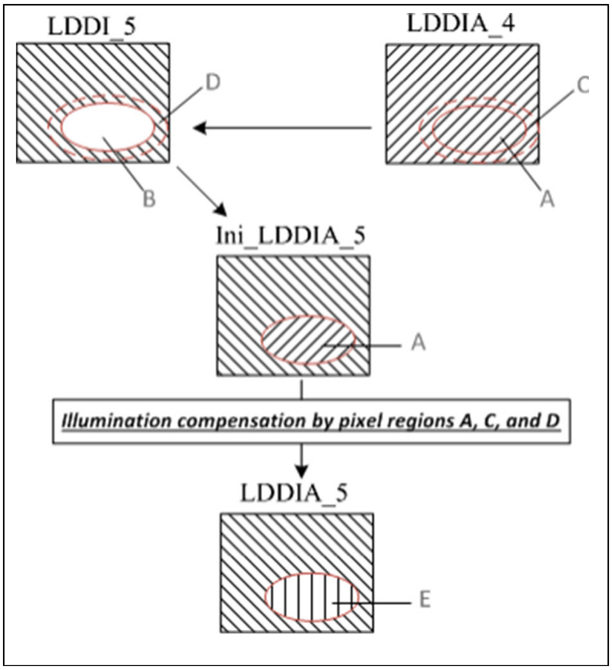


Fig. 6 The illustration of the illumination compensation process in the LDDIA update. The illumination information of the pixels in region A in Ini_LDDIA_5 will be adjusted by the proposed algorithm to find the region E pixels with proper illumination, to obtain LDDIA_5

same spatial locations, (x_i,y_i), $i=1\sim N$. The N spatial locations (x_i,y_i) are stacked into an $N\times 2$ matrix **Z**. The pixel values in portion C form an $N\times 1$ vector, defined as **c**, sorted by the order of their pixel location according to **Z**. We form an $N\times 1$ vector **d** by the same procedure from portion D. The illumination differences between C and D can be defined by $\mathbf{diff}_{dc} = \mathbf{d}-\mathbf{c}$. We also define the first column of **Z** as **x**, and the second column as **y**. Finally, we can model the illumination differences based on the spatial pixel locations as follows:

$$\mathbf{diff}_{dcN\times 1} = \alpha + \beta \mathbf{x}_{N\times 1} + \mu \mathbf{y}_{N\times 1}$$

The optimal coefficients $\{\alpha,\beta,\mu\}$ are solved to minimize the differences between $\mathbf{diff}_{dc}$ and the estimate of $\mathbf{diff}_{dc}$. After finding the optimal coefficients $\{\alpha,\beta,\mu\}$, we have the illumination difference model to estimate the illumination difference between region B in LDDI_5 and region A in LDDIA_4. We assume that M is the number of pixels in portion A, which can be formed into a $M\times 1$ vector **a**. The corresponding pixel locations (x_i,y_i), $i=1\sim M$ are stacked into another $M\times 2$ matrix **K;** and the first column is defined as **g**, and the second as **h**. We can estimate the illumination difference vector $\mathbf{diff}_{ba}=\mathbf{b}-\mathbf{a}$, where **b** is the supposed true pixel from region B, by

$$\mathbf{diff}_{baM\times 1} = \alpha + \beta \mathbf{g}_{M\times 1} + \mu \mathbf{h}_{M\times 1}.$$

These estimated illumination differences between portions A and B are then added to the pixels A in Ini_LDDIA_5 to obtain the *estimated* true pixels with correct illumination in **b**, denoted by **e**:

$$\mathbf{e} = \mathbf{a} + \mathbf{diff}_{ba}.$$

This **e** is then organized according to the spatial location to be **E. E** is to update the hole location in Ini_LDDIA_5, to obtain the final LDDIA_5 after illumination compensation, as shown in Fig. 6. This LDDIA_5 will serve as a new database frame, and the whole procedure is repeated to generate LDDIA_6; the process is repeated for all the following frames. A visual result of the LDDIA_5 can be observed in Fig. 7(e). After the LDDIA_5 has been obtained, a DIBR of it is performed to obtain the virtual LDDIA_5 in the synthesized view. The visual example of it can be seen in Fig. 7(g). This virtual LDDIA_5, after the proposed texture synthesis approach in the next section, will be combined with the virtual HDDI_5, as the final result of the virtual color view.

Compared with the work in [7], it is worth noting that the proposed use of the information from other frames is *before* the DIBR, whereas the work in [7] uses the information from other frames *after* the DIBR in the virtual view. The system comparisons are shown in Fig. 1(a) and (b). There are two advantages of doing so in the proposed method:

1. The pixels after the DIBR are estimates of the pixels in the virtual view. Using the estimated past pixels in the virtual view may provide an inaccurate starting point, and the errors may accumulate along the way of the later view generation processing. To avoid these disadvantages, the proposed method uses the information from past frames before the DIBR to yield more accurate pixel estimates.
2. Freeview generation involves the end user's ability to switch the views anytime to any virtual view. Therefore, the freeview generation algorithm should be able to deal with frequent switches of views. If the work in [7] is used, since the database frame is maintained after the DIBR, then whenever the end user wants to generate a new view, a new database frame corresponding to the new view should be created and maintained; but this is not an efficient use of the memory to store multiple database frames, nor of computation complexity to create and maintain them. On the other hand, if the database

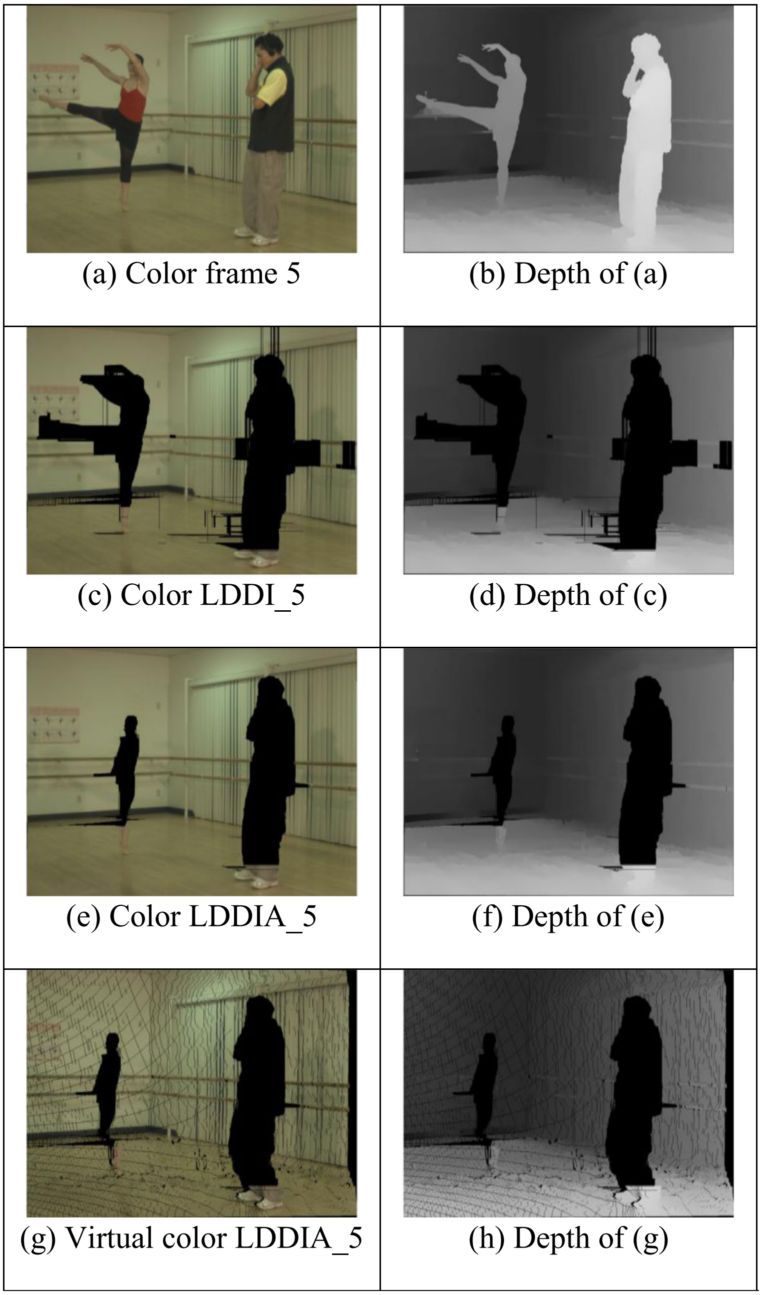


Fig. 7 Result illustration of the proposed method. The example is to generate the virtual view in frame 5 using database frame LDDIA_4

frame is used and maintained before the DIBR as in the proposed method, only the database frame of the current view is needed; this saves on memory and computational complexity.

4 Inpainting algorithm using iterative depth and color update

As can be seen in Fig. 7(e), there can be many *hole pixels* in the LDDIA, and thus in the virtual LDDIA, shown in Fig. 7(g). This is because sometimes, when HDDI is static throughout the frames in a sequence, the information is occluded throughout the video, so the LDDI has no information available to update the database frame LDDIA. In this section, we propose a texture synthesis algorithm to fill the *hole pixels*. Based on the texture synthesis method in [2], we design a novel method with a proposed processing priority and a proposed patch matching scheme. Since the above procedures require the information of the depth frame in the virtual view, we propose an efficient method to generate a complete virtual depth frame. The algorithm iterates between the generation of the virtual color and depth frames on a patch basis, which is a novel design.

4.1 Texture synthesis priority computation using depth information

Given an image with holes to be filled, we first find out the best filling order on a pixel basis. Having computed the priorities of all the pixels on the boundary of holes, we fill the hole pixels in the order from highest-priority hole pixel to lowest-priority hole pixel. This subsection discusses the proposed priority computing procedure.

Given an input image, as in Fig. 8, the target region (hole region), denoted by , is to be filled. The source region, ϕ, is defined as the entire image minus the target region. There are pixels on the boundary δ of the hole, and we want to compute the priorities for each of them. Take the pixel p among them, for example, as shown in Fig. 8. A template patch window Ψ_P centered at p is defined to help the texture synthesis process, whose size is 9×9 in our method. First, as done in [2], the confidence term $C(p)$ and the data term D (p) of p are computed:

$$C(p) = \frac{\sum_{q\in\Psi_p \bigcap (I-)} C(q)}{|\Psi_p|},\ D(p) = \frac{|\nabla I_p^{\perp}.n_p|}{\alpha},$$

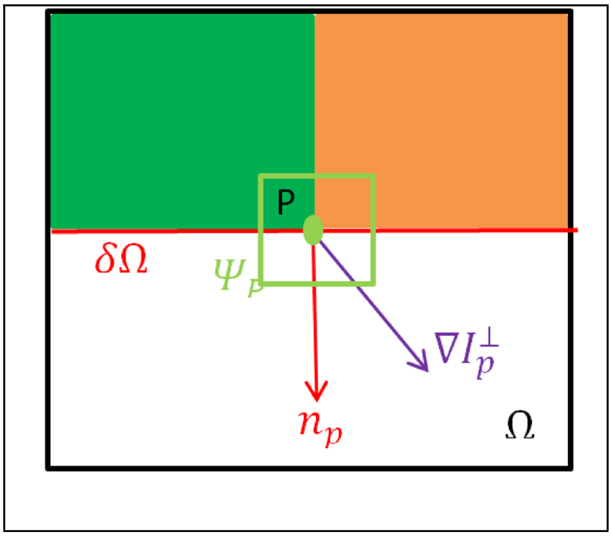


Fig. 8 Illustration of texture synthesis priority computation

where $|\Psi_p|$ is the size of the Ψ_p, α is the normalization factor, n_p is a unit vector orthogonal to the δ in the point p, and $\perp$ denotes the orthogonal operator. And the priority is computed by [2]:

$$\widetilde{P(p)} = C(p)D(p).$$

But this expression fails to consider the depth information. A hole is something that is occluded *behind* an object in an original view, and so the filled material should be something of low depth value (closer to the background). In the context of hole-filling, we should give more priority to the pixels whose depth value is low. Thus, we propose to modify the priority computation to include the depth information:

$$P(p) = \lambda(p)C(p)D(p),$$

where $\lambda(p)=1/depth(p)$ and $depth(p)$ is the depth value at p; this expression means that a pixel having low depth gives high weighting to the priority calculation.

4.2 Patch matching scheme

Once the priorities of the pixels on the hole boundary have been found in the previous subsection, the pixel with the highest priority is to be the center of the patch Ψ to fill the hole in the patch. To illustrate the proposed hole-filling method, we take Fig. 9 for example. The 9×9 source patch in Fig. 9 is the patch to be processed. In this source patch, the available pixel region is labelled A, and the hole pixel region is labelled B. The method in [2] searches every available 9×9 reference patch in a frame to find the best reference patch for its region B to fill in the region B (hole) in the source patch. The best patch for region B is found in [2] from the reference patches whose region A has the minimum average pixel difference with the source region A. *But the quality of the region B in the reference patch should be the main factor for the visual result of the patch matching process*, and it makes more sense to *consider whether the pixels in the region B of the reference patch be consistent with the source region A, in order to avoid visually bad choices.* For a visual example, in Fig. 9 the region B in reference patch *b* should be a better choice than that in reference patch *a* for filling in the region B in the source patch, since the pixel levels of region B in reference patch *b* are more consistent with those of the region A in the source patch. Therefore, we here propose the constraint to exclude reference patches whose pixels are inconsistent with the source patch from being considered in the inpainting process in [2]: we consider a reference patch in the inpainting process only when the minimum pixel value in the reference patch's region B is less than the minimum pixel value in the source patch's region A, plus a constant τ:

$$min(\text{pixels in region B of a reference patch}) \leq [(\text{pixels in region A of the source patch}) + \tau]$$

In this way, we can exclude many visually inappropriate references from the consideration of the inpainting process and so obtain the visually better reference region B.

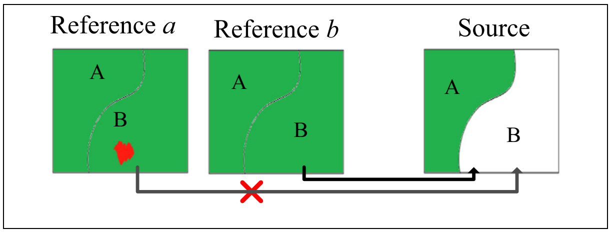


Fig. 9 Proposed patch-matching scheme

4.3 Efficient hole-filling for the virtual depth

As mentioned in subsection A, the depth values are required for the proposed priority computation method. Therefore, the depth values of the holes in the virtual view also need to be generated. Hence, we propose an efficient hole-filling method for the virtual depth. Right after the pixel with the highest priority has been patch-processed for color according to the previous section, we use that pixel location in the virtual depth map. We create a 9×9 square area centered at the location, as the source area in Fig. 9; we aim properly to fill the corresponding region B in depth. Since the texture of the depth is homogeneous, we use only a single estimated depth value to fill the whole region B. To find the estimate, we first find the minimum of the depth value in region A, denoted as Min_A, and we average all depth values that are less than $\mathrm{Min}_A+\varepsilon$ to obtain the estimated value for the hole to fill in region B in depth.

After this patch processing in the virtual "depth" view, the algorithm returns to the patch processing in the virtual "color" view (priority computing and reference patch selection) for the patch associated with the pixel of the next priority value. The algorithm iterates between virtual color and depth view on a patch basis, as shown in Fig. 10, until it has found all the missing pixels in both views.

5 Experimental results

In this section, we perform an experiment to compare the view generation performances of the proposed method against the work in [7]. The system diagrams of the two works are shown in Fig. 1(a) and (b). We first introduce the data set in the experiment. We have compared both methods objectively and subjectively.

5.1 Data set and quality measurements

In our experiment, we use multiview plus a depth sequence database including the sequences "Book arrival," "Ballet," and "Love bird." The resolution is 1024×768. We consider the view

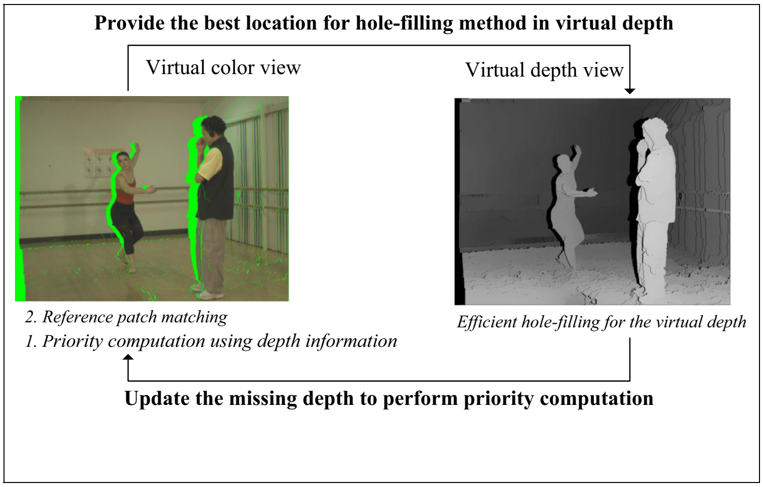

Provide the best location for hole-filling method in virtual depth
Virtual color view
Virtual depth view
2. Reference patch matching
1. Priority computation using depth information
Efficient hole-filling for the virtual depth
Update the missing depth to perform priority computation

Fig. 10 Proposed iterative approach to perform hole-filling in virtual color and depth frames. *Green areas* in the left and *black areas* in the right indicate holes

generation between the available views in the database to compute objective video quality measurements. In our experiment, the original view and the virtual view are not limited to adjacent views, but also a few available views away. Using both subjective and objective comparisons, we assess the performances of the proposed algorithm and the work in [7]. For the algorithm in [7], we implement it by ourselves following the description in the paper [7].

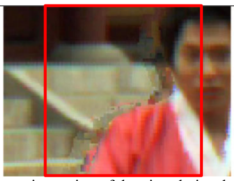
5.2 Subjective assessment

We first evaluate the performance subjectively. In Fig. 11, the sequence "Love bird" is considered. Figure 11(a) is the 20th frame in view 4, the ground truth of the frame to be synthesized. Figure 11(b) is the zoom-in version of the virtual view generated by [7], and Fig. 11(c) is the zoom-in version of the virtual view generated by the proposed method. As can be seen, the method in [7] produces visual artifacts in the hole-filling area, whereas the proposed method fills in the hole with more consistent pixels for better final quality.

For the video sequence "Ballet," the performance comparisons are in Fig. 12. As shown in Fig. 12(b), the method in [7] fills the hole area with pixels similar to those of the wall, but it does not account well for the illumination changes; the visual result is degraded. The proposed work, in contrast, can deal with the illumination changes when filling holes; the visual results are more pleasant, as shown in Fig. 12(c). Another comparison can be seen in Fig. 12(d) and (e). The work in [7] fills in the hole with incorrect pixels, whereas the proposed work can recover the missing area with better pixels, and so its image quality is much better.

Similar results can be seen in Fig. 13 for the "Book arrival" sequence. In Fig. 13(b), the hole-filling work in [7] produces visual artifacts in the hole area near the arms of the chair, whereas the proposed work can find more correct pixels to fill in, as in Fig. 13(c). Figure 13(d) and (e) also show that the proposed method can estimate the missing pixels with a texture more consistent with the surrounding available pixels.

(a) Ground truth for the virtual frame


(b) Zoom-in version of the virtual view by [12]

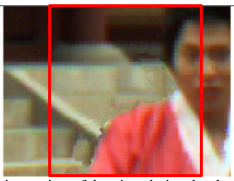
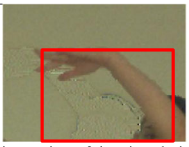
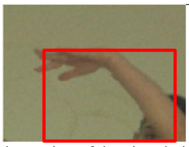
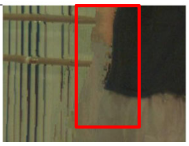

(c) Zoom-in version of the virtual view by the proposed method

Fig. 11 Virtual view results for the "Love Bird" sequence

(a) Ground truth for the virtual frame


(b) Zoom-in version of the virtual view by [12]


(c) Zoom-in version of the virtual view by the proposed method


(d) Another zoom-in version of the virtual view by [12]

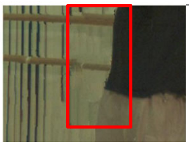
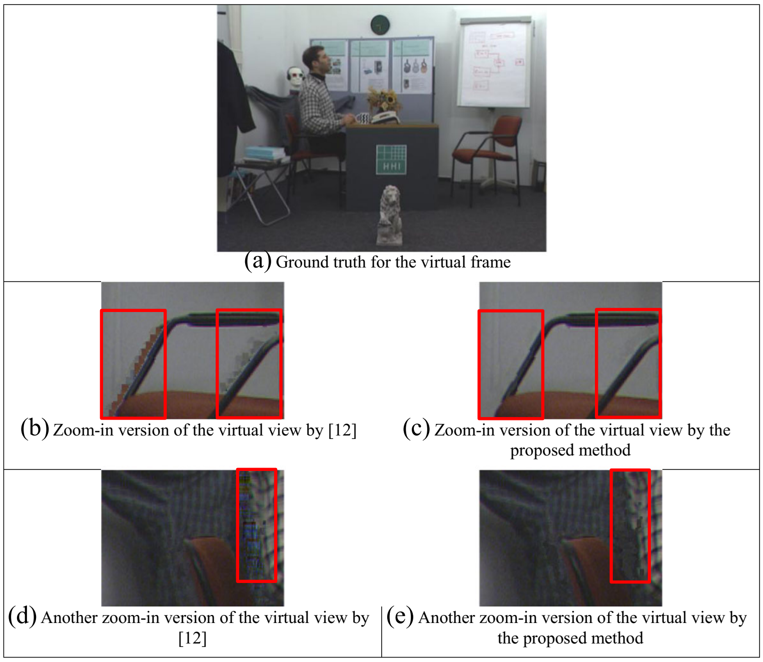

(e) Another zoom-in version of the virtual view by the proposed method

Fig. 12 Virtual view results for the "Ballet" sequence

5.3 Objective assessment

For the objective comparisons, we consider PSNR and SSIM (Structural SIMilarity). For the computation of PSNR and SSIM, we use three classes of different focusing area in a frame for the computation. For class 1, we perform the computation for the pixels in the entire frame to consider overall video quality. We define these pixels to be entire-frame pixels, and the full results are shown in Table 1. For class 2, to better account for the hole-edge artifacts, we additionally perform the computation only for the neighboring pixels around the hole edge locations. Specifically, we define a 5-pixel wide region around the hole, and we collect the pixels in the region to compute the PSNR and SSIM; this is the region where usually artifacts are observed due to depth misalignment. We define these pixels to be hole-edge pixels, and the full results are shown in Table 2. For class 3, to focus on the video quality of the hole-filling algorithm, we consider the pixels only in the hole area. We define these pixels to be hole pixels, and the full results of corresponding PSNR and SSIM are shown in Table 3.

The PSNR and SSIM for different classes are computed between the virtual views generated by the two methods against the ground truth of the virtual view, for a randomly chosen frame for each virtual view. We experiment for different video sequences, with different view generation pairs, as shown in Tables 1, 2, and 3. For example, $3 \rightarrow 4$ means the use of view 3 to generate the virtual view 4 using DIBR. As can be seen, tests are not only for the views that are adjacent to each other, as for $3 \rightarrow 4$, $2 \rightarrow 3$, and $5 \rightarrow 6$, but also for views that are two views away, such as $8 \rightarrow 10$ and $8 \rightarrow 6$. *We design the comparison in this way to discover the robustness of the two methods in generating distant virtual views.* One thing to note is that for Ballet, we have more complete videos with both color frames and depth frames for us to perform DIBR and make comparisons. However for Book arrival, Love bird and


(a) Ground truth for the virtual frame

(b) Zoom-in version of the virtual view by [12]

(c) Zoom-in version of the virtual view by the proposed method

(d) Another zoom-in version of the virtual view by [12]

(e) Another zoom-in version of the virtual view by the proposed method

Fig. 13 Virtual view results for the "Book arrival" sequence

Mobile, the number of views is limited. Also, for some particular views, it is not always that both color and depth frames are available for us to perform DIBR. Therefore, for some videos with only limited number of views that have both color and depth frames, in order to increase the number of test results, inverse DIBR is performed.

We first discuss the traditional way of measuring PSNR and SSIM by whole frame in Table 1. As can be seen, for PSNR, the proposed method is better than [7] in every case, and can outperform [7] by up to 2.9575 dB. For SSIM, even though there is one case where the proposed method is slightly worse than [7] by −0.0057, the proposed method is better in the

Table 1 PSNR & SSIM (Entire-frame pixels) comparison between the work in [7] and the proposed work

Sequence	View generation direction	PSNR (dB)			SSIM		
		[7]	Proposed	Gain	[7]	Proposed	Gain
Ballet	$3 \rightarrow 4$	29.4768	30.1829	0.7061	.8562	.8656	0.0094
Ballet	$2 \rightarrow 3$	28.8692	29.5983	0.7291	.8451	.8538	0.0087
Ballet	$5 \rightarrow 6$	28.3748	29.1855	0.8107	.8378	.8546	0.0168
Book arrival	$8 \rightarrow 10$	23.8823	24.4956	0.6133	.8795	.8738	−0.0057
Love bird	$8 \rightarrow 6$	23.5615	25.6371	2.0756	.7290	.7559	**0.0269**
Love bird	$6 \rightarrow 8$	25.1082	26.1518	1.0436	.7923	.7924	0.0004
Love bird	$4 \rightarrow 6$	25.8774	25.8862	0.0088	.8724	.8728	0.0004
Mobile	$5 \rightarrow 3$	30.5049	33.4624	**2.9575**	.9696	.9786	0.009
Mobile	$3 \rightarrow 5$	29.1126	30.0359	0.9233	.9654	.9719	0.0065

The bold number is the largest gain in the comparisons

Table 2 PSNR & SSIM (Hole-edge pixels) comparison between the work in [7] and the proposed work

Sequence	View generation direction	PSNR (dB)			SSIM		
		[7]	Proposed	Gain	[7]	Proposed	Gain
Ballet	3 → 4	26.3788	30.0509	**3.6721**	.9879	.9956	0.0077
Ballet	2 → 3	25.8654	28.5026	2.6402	.9904	.9941	0.0037
Ballet	5 → 6	24.5615	27.3239	2.7624	.9645	.9929	**0.0284**
Book arrival	8 → 10	25.1744	26.0901	0.9157	.9657	.9868	0.0211
Love bird	8 → 6	19.6281	20.330	0.992	.9678	.9718	0.004
Love bird	6 → 8	20.0250	20.2517	0.2267	.9784	.9795	0.0011
Love bird	4 → 6	18.8654	21.1935	2.3281	.9753	.9776	0.0023
Mobile	5 → 3	27.2241	30.0764	2.8523	.9957	.9973	0.0016
Mobile	3 → 5	18.7898	22.2343	3.4445	.9890	.9725	−0.0165

The bold number is the largest gain in the comparisons

rest of the cases, by up to 0.0269. This shows the outstanding overall performance of the proposed method against [7] considering the entire frame.

Next, to focus on the image quality in the hole edge portion to consider mainly the edge artifact, we discuss the results in Table 2. For the hole-edge area, the proposed method is better than the work in [7] in all cases, and the improvement is up to 3.6721 dB. For SSIM, the proposed method is better than [7] in 8 out of 9 cases by up to 0.0284 improvement. The results indicates that the proposed method outperforms [7] also in the area of hole edges, specifically due to the proposed Boundary Rectification Optimization Algorithm to fix the depth boundary error.

Finally, for the hole-filling quality, we concentrate on the recovered pixels only in the hole area, and the results are in Table 3. As can be seen, the proposed method is better than [7] in all cases both for PSNR and SSIM, and the maximum improvement is up to 7.8845 dB in PSNR and 0.0328 in SSIM, respectively. This demonstrates the outstanding performance of the proposed Inpainting Algorithm using Iterative Depth and Color update, to create more visually pleasant pixels for the holes.

In the proposed method, the great improvements over the work in [7] are mainly due to the following factors. For the background information updating process in [7], the boundary artifact

Table 3 PSNR & SSIM (Hole pixels) comparison between the work in [7] and the proposed work

Sequence	View generation direction	PSNR (dB)			SSIM		
		[7]	Proposed	Gain	[7]	Proposed	Gain
Ballet	3 → 4	24.7824	26.2257	1.4433	.9828	.9933	0.0105
Ballet	2 → 3	23.8081	25.0167	1.2086	.9807	.9911	0.0104
Ballet	5 → 6	23.8685	25.5999	1.7314	.9585	.9913	**0.0328**
Book arrival	8 → 10	12.8221	13.6996	0.8775	.8206	.8285	0.0079
Love bird	8 → 6	17.3868	19.3511	1.9643	.9545	.9717	0.0172
Love bird	6 → 8	18.1870	23.1781	4.9911	.9805	.9895	0.009
Love bird	4 → 6	17.5963	23.8353	6.2390	.9784	.9911	0.0127
Mobile	5 → 3	19.2239	27.1084	**7.8845**	.9789	.9929	0.014
Mobile	3 → 5	17.3544	19.0700	1.7156	.9521	.9723	0.0202

The bold number is the largest gain in the comparisons pixels are not prevented, and therefore are accumulated throughout the process, whereas in the proposed method, the boundary artifact pixels are first removed by the proposed depth boundary rectification optimization, and therefore are excluded in the subsequent proposed background information updating process LDDIA (low-depth discontinuity information accumulator). Also, in [7], the centroid method for depth filling is used to define what kind of information to be selected for background updating; this depends on the quality of the centroid method. In the proposed work, however, we perform LDDI separation to make sure that the background is almost static. The information starts updating the LDDIA from most recent LDDI to ensure that the LDDIA information is as up-to-date as possible. Furthermore, in the LDDIA updating process, the illumination compensation is performed to balance the lighting condition among different frames, making the lighting condition in the LDDIA more consistent. Finally, in the hole-filling stage, the proposed method iterates between the holes in color and depth frames, to improve the inpainted quality in both frames, especially with the proposed priority computation using depth information, patch matching scheme, and efficient hole-filling for the virtual depth.

6 Conclusion

We propose a freeview synthesis algorithm. Having first considered the misalignment of the edges between the color and depth frames, it rectifies the depth frame by the proposed boundary rectification optimization algorithm. We propose ideas for classifying LDDI and HDDI in a frame by the sizes of the resulting holes in the virtual view. In the algorithm, the LDDI frame is further processed with the information from the database frame, formed by the information of the past frames, to yield a more complete frame. The illumination differences between the LDDI and the database frame are computed and compensated for the LDDI frame to form the LDDIA frame. The DIBR is performed for LDDIA to obtain a virtual LDDIA, on which the proposed inpainting method performs. The processing order for the inpainting is first computed according to the depth information. The patch matching process is then performed, considering the consistency between the reference patch and the source patch. Also filled by the proposed algorithm is the corresponding hole area in the virtual depth frame. The whole inpainting process iterates between the virtual color view and the virtual depth view, to finalize the virtual LDDIA. The final result of the virtual color view is a combination of the final virtual LDDIA frame and the virtual HDDI frame. Experiment results show that the proposed system performs better than a state-of-the-art method [7] for several sequences and for several view generation pairs, subjectively and objectively. The proposed method's PSNR improvement over [7] can be up to 2.9575 dB.

Acknowledgments This research is supported by the National Science Council, Taiwan, under Grants NSC 101-2221-E-033-036, NSC 102-2221-E-033-018, NSC-102-2221-E-033-063, by the Ministry of Science and Technology, Taiwan under Grant MOST 103-2221-E-033-020, MOST-103-2221-E-033-070, MOST-103-2622-E-033-001-CC2, and by the College of Electrical Engineering and Computer Science in Chung Yuan Christian University, Taiwan under Grant CYCU-EECS-10301.

References

1. Bertalmio M, Sapiro G, Caselles V, Ballester C (2000) Image inpainting. In: SIGGRAPH '00 Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, ACM Press/Addison-Wesley Publishing Co., New York, pp 417–424

2. Criminisi A, Pérez P, Toyama K (2004) Region filling and object removal by exemplar-based image inpainting. IEEE Trans Image Process 13:1200–1212
3. Dong T, Lai PL, Patrick L, Gomila C (2009) View synthesis techniques for 3D video. In: Proc. SPIE 7443, Applications of Digital Image Processing XXXII, 74430T. doi:10.1117/12.829372
4. Fehn C (2004) Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV. In: Proc. SPIE 5291, Stereoscopic Displays and Virtual Reality Systems XI, 93. doi:10.1117/12.524762
5. Hervieu A, Papadakis N, Bugeau A, Gargallo P, Caselles V (2010) Stereoscopic image inpainting: distinct depth maps and images inpainting. Pattern Recognition (ICPR), 2010 20th International Conference on, pp 4101–4104. doi:10.1109/ICPR.2010.997
6. Kauff P, Atzpadin N, Fehn C, Müller M, Smolic A, Tanger R (2007) Depth map creation and image-based rendering for advanced 3DTV services providing interoperability and scalability. Sig Process Image Commun Spec Issue Three-Dim VIdeo Telev 22:217–234
7. Ndjiki-Nya P, Koppel M, Doshkov D, Lakshman H, Merkle P, Muller K, Wiegand T (2011) Depth image based rendering with advanced texture synthesis for 3-D video. IEEE Trans Multimed 13:453–465
8. Schmeing M, Jiang X (2010) Depth Image Based Rendering: A faithful approach for the disocclusion problem, 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), pp 1-4. doi:10.1109/3DTV.2010.5506596
9. Smolic A, Mueller K, Merkle P (2006) 3D video and free viewpoint video technologies, applications and MPEG standards. IEEE Int Conf Multimed Expo 53:2161–2164
10. Smolic A, Muller K, Dix K, Merkle P, Kauff P, Wiegand T (2008) Intermediate view interpolation based on multiview video plus depth for advanced 3D video systems. IEEE Int Conf Image Process 11:2448–2451
11. Solh M, AlRegib G (2012) Hierarchical Hole-Filling For Depth-Based View Synthesis in FTV and 3D Video. IEEE Journal of Selected Topics in Signal Processing 6(5):495–504
12. Wang L, Jin H, Yang R, Gong M (2008) Stereoscopic inpainting: joint color and depth completion from stereo images. IEEE Conf Comput Vis Pattern Recog 10:1–8
13. Zhang L, Tam W (2005) Stereoscopic image generation based on depth images for 3D TV. IEEE Trans Broadcast 51:191–199

Ting-Lan Lin received the B.S. and M.S. degrees in electronic engineering from Chung Yuan Christian University, Zhongli, Taoyuan, Taiwan, in 2001 and 2003, respectively, and the Doctorate degree in electrical and computer engineering from the University of California, San Diego, La Jolla, CA, USA, in 2010. In 2008, he interned in the Display System group at Qualcomm, San Diego, CA, USA. Since 2011, he has been an Assistant Professor with the Department of Electronic Engineering, Chung Yuan Christian University, Taiwan. His current research interests include video compression, video streaming in lossy networks, optimization of packet prioritization, and perceptual video quality.

Uday Singh Thakur received the B.S. degree in electronics & communication engineering from Amity University, Noida, India in 2011. He received the M.S. degree in electronic engineering from Chung Yuan Christian University, Zhongli, Taoyuan, Taiwan, in July 2013. His research interest focuses on freeview generation. Currently, he is a PhD scholar in Institut für Nachrichtentechnik, RWTH Aachen University, Germany. His research interests include perceptual video compression and texture synthesis.

Chi-Chan Chou received the B.S. degree in the department of electronic engineering from Chung Yuan Christian University, Zhongli, Taoyuan, Taiwan in 2013. He is now pursuing M.S. degree in the department of electronic engineering from Chung Yuan Christian University, Zhongli, Taoyuan, Taiwan. His research interests are image processing and video compression.

Shih-Lun Chen received the B.S. and M.S. degrees in Electrical Engineering from National Cheng Kung University, Tainan, Taiwan, in 2002 and 2004, respectively. He received the Ph.D. degree in the Department of Electrical Engineering, National Cheng Kung University Tainan, Taiwan in 2011. He was an Assistant Professor with the Department of Electronic Engineering, Chung Yuan Christian University in Taiwan, from 2011 to 2014. Since 2014, he has been an Associate Professor with the Department of Electronic Engineering, Chung Yuan Christian University in Taiwan. His current research interests include image processing, video, data compression, fuzzy logic control, VLSI chip design, wireless body sensor network, bio-medical signal processing, and reconfigurable architecture.