Muddala *et al. EURASIP Journal on Image and Video Processing* (2016) 2016:9
DOI 10.1186/s13640-016-0109-6

EURASIP Journal on Image and Video Processing
a SpringerOpen Journal

RESEARCH Open Access

Spatio-temporal consistent depth-image-based rendering using layered depth image and inpainting

Suryanarayana M. Muddala, Roger Olsson and Mårten Sjöström*

Abstract

Depth-image-based rendering (DIBR) is a commonly used method for synthesizing additional views using video-plus-depth (V+D) format. A critical issue with DIBR-based view synthesis is the lack of information behind foreground objects. This lack is manifested as disocclusions, holes, next to the foreground objects in rendered virtual views as a consequence of the virtual camera "seeing" behind the foreground object. The disocclusions are larger in the extrapolation case, i.e. the single camera case. Texture synthesis methods (inpainting methods) aim to fill these disocclusions by producing plausible texture content. However, virtual views inevitably exhibit both spatial and temporal inconsistencies at the filled disocclusion areas, depending on the scene content. In this paper, we propose a layered depth image (LDI) approach that improves the spatio-temporal consistency. In the process of LDI generation, depth information is used to classify the foreground and background in order to form a static scene sprite from a set of neighboring frames. Occlusions in the LDI are then identified and filled using inpainting, such that no disocclusions appear when the LDI data is rendered to a virtual view. In addition to the depth information, optical flow is computed to extract the stationary parts of the scene and to classify the occlusions in the inpainting process. Experimental results demonstrate that spatio-temporal inconsistencies are significantly reduced using the proposed method. Furthermore, subjective and objective qualities are improved compared to state-of-the-art reference methods.

Keywords: View synthesis, Depth-image-based rendering, Image inpainting, Texture synthesis, Hole filling, Disocclusions, Layered depth image, Temporal consistency

1 Introduction

Three-dimensional television (3DTV) and free viewpoint television (FTV) technologies are actively pursued in both research and compression standardization, which is supported by the development of enabling display technologies [1]. The 3DTV application aims to provide realistic depth impressions by presenting several views of a scene simultaneously from slightly different viewpoints. In comparison, the FTV application offers users the ability to look around the scene by generating new views (in 2D or 3D) from any arbitrary position [2]. Hence, both applications require the display of a large number of views, all from different viewpoints, in order to provide a smooth transition between the views and thereby an immersive visual experience.

However, capturing and transmission of the required number of views is not a feasible solution from a bandwidth point-of-view. Therefore, view synthesis is employed as an efficient solution to produce the necessary number of views at the receiver side using a smaller set of captured and transmitted views. Depth-image-based rendering (DIBR) methods generate these views from formats such as video-plus-depth (V+D) and multiview video-plus-depth (MVD) [3]. In this work, we define the input views that a DIBR method operates on as original views and output views it produces as the virtual (i.e. rendered) views. A virtual view is produced using DIBR by utilizing texture (the 2D color information) and a corresponding depth image. Using geometrical 3D warping, which relies on knowledge about scene depth and camera parameters,

*Correspondence: marten.sjostrom@miun.se
Department of Information and Communication Systems, Mid Sweden University, Holmgatan 10, 85170 Sundsvall, Sweden

pixels from the original view are projected into the virtual view's image plane. An illustration of the view rendering is shown in Fig. 1, where the V+D format is adopted.

Warped images, which are rendered using DIBR, exhibit several artifacts due to the inherent properties of warping as shown in Fig. 2a. DIBR artifacts are ghosting and uncovered areas (i.e. holes). Ghosting artifacts are blended texture values caused by objects that are neighboring in the original view but becomes overlapped in the virtual view due to the depth and texture misalignments. A typical cause of this misalignment is the noise introduced in the depth map around object borders, frequently occurring in depth from stereo matching. Ghosting artifacts also cause inconsistent hole filling. Holes are defined as pixels in the warped image that receives no texture values from the forward warping process (see Fig. 2a). Holes can be classified into three groups: *cracks*, *disocclusions* and *out-of-field areas*. Cracks are holes of size one to two pixel wide caused by rounding the floating point position of a warped pixel to make it fit the pixel grid of the virtual view image plane. Disocclusions and out-of-field areas are holes of larger size caused by the depth discontinuities between objects within the scene and the limited field of view of the camera. The size of the holes increases as the distance between the original and warped camera view position increases. Dealing with disocclusions and out-of-field areas in a visually plausible manner is a much greater challenge. In the next section, we will elaborate on how we address these problems.

In order to solve the problems caused by the holes, we use the following terminology. We define foreground as the part of the scene closer to the camera that occludes other objects and background to be the part of the scene farther from the camera partially occluded by the foreground. Disocclusions are occluded regions in the original view that are revealed in the virtual view as a result of warping. Disocclusions occur at the borders of the object depending on the relative position translation between the original and virtual view camera. Out-of-field areas are caused by the limited field of view in the original view and they occur at the image boundaries of the virtual view. Moreover, out-of-field areas usually occur in the linear camera set-ups, whereas in converging camera set-up may not. The size of the holes (disocclusions and out-of-field areas) increases when the size of the of the inter-camera distance (i.e. the baseline) increases.

1.1 Related work

The literature proposes several pre-processing and post-processing methods to overcome DIBR artifacts. Pre-processing methods smooth the depth map prior to warping using, e.g. symmetric or asymmetric Gaussian filter [4, 5]. However, pre-processing of the depth map may imply geometrical distortions in the warped

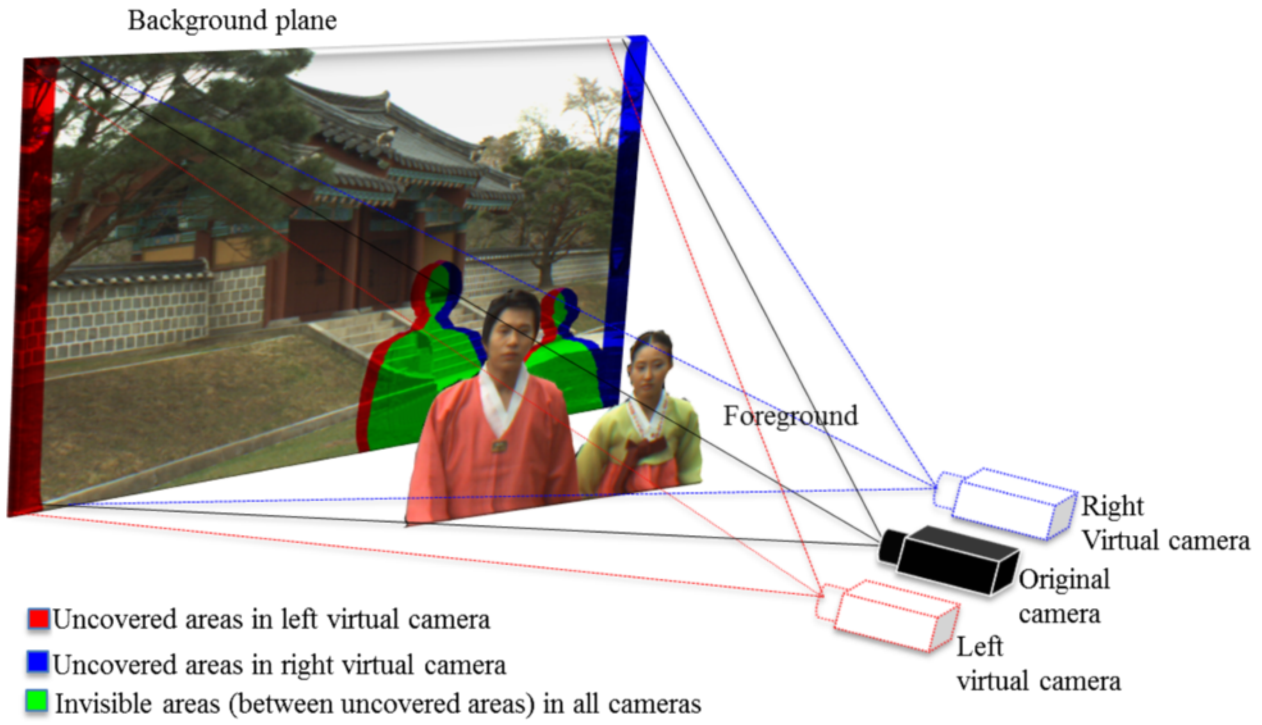
Background plane
Foreground
Right
Virtual camera
Original
camera
Left
virtual camera
🟥 Uncovered areas in left virtual camera
🟦 Uncovered areas in right virtual camera
🟩 Invisible areas (between uncovered areas) in all cameras

Fig. 1 Illustration of DIBR view extrapolation from V+D. Invisible uncovered areas in original camera are visible in the virtual cameras, so they need to be filled

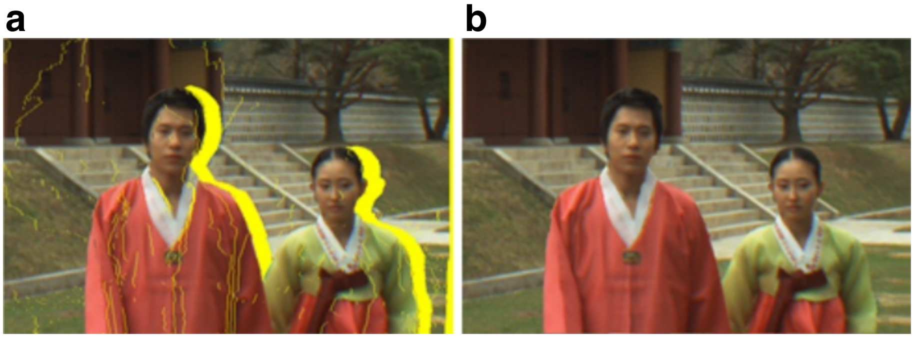

Fig. 2 View synthesis results. **a** Synthesized image (holes in *yellow color*). **b** Proposed synthesized image

image. However, post-processing techniques add complicated steps after warping in the warped image [6–9]. Image-domain warping methods are another way of reconstructing the virtual views using sparse disparity information [10, 11]. A number of methods combine both pre-processing and post-processing techniques to address rendering problems [8].

Several hole-filling methods have been proposed in the literature. In the context of virtual view synthesis from a single V+D, the hole-filling methods can be generally classified into two groups. In the first group, the data required to fill the hole is synthesized from the image itself. We term this group spatial domain hole-filling method. In the second group, the data required to fill the hole is recovered from both the spatial and temporal domain, so we term them temporal domain hole-filling methods.

Spatial domain hole filling: Spatial domain hole-filling methods can be classified into interpolation, extrapolation and image inpainting. Interpolation and extrapolation methods use the texture of neighbor pixels around the holes to fill missing areas in the warped images [12]. However, for large holes, the results yield annoying artifacts and unnatural textures. In contrast, inpainting methods use structural and textural properties of the image in the neighborhood of the hole to fill missing areas [13, 14]. The view synthesis reference software (VSRS) is used in the standardization activities of 3DTV and FTV and combines various techniques to produce virtual views [15]. For example, VSRS applies a diffusion-based inpainting method to fill any holes. In general, diffusion methods suffer from the introduction of blurred results in the virtual image, more so when the hole is large. In addition, VSRS suffers from limitations in how foreground-background is identified during inpainting, which results in holes being partially filled with foreground textures, and not only background as appropriate. To reduce this problem, Oh et al. [16] replaced the foreground pixels on the boundary with background values.

To overcome the limitations of these diffusion methods, exemplar methods combine both structural and textural properties when filling in missing areas [17]. In exemplar methods, a filling order is first computed on the hole boundary using the number of known pixels in a patch (confidence term) and structure in the neighborhood (data term). Next, a highest priority patch (target patch) is selected to be filled using the best patch (source patch) from the neighborhood using patch matching. Although the exemplar method has the advantage of recovering the structure, this priority order is not suitable for hole filling in the view synthesis context, since holes should be filled only from the background texture. Several methods have been proposed by improving filling priorities and adding various foreground-background classification techniques in the exemplar methods including the methods in [18–23]. Despite improvements, these methods still have spatial texture inconsistencies and moreover, they severely suffer from temporal inconsistencies (flickering).

Another approach to filling missing areas is to generate the hole information in the original view before warping, instead of filling the holes after DIBR. This approach is similar to methods working with the layered depth video (LDV) format. LDV contains occlusion information along with V+D and share many properties with the layered depth image (LDI) [24]. The method in [25] has been presented to generate this occlusion data from a stereo pair. Several ways to produce the necessary occlusion data from different camera set-ups are presented in [26]. However, an explicit way to produce LDI data from a single V+D is not presented. Recently, a method has been presented in [27] to generate LDI from V+D. Although this method has improved spatial consistency, it still suffers from flickering in the temporal domain.

Temporal domain hole filling: Video inpainting has been an active research field in computer vision for the past decade with increasing video editing applications. It makes sense to use the temporal true texture instead of the synthesized texture from the inpainting. Chen et al. proposed a method to recover the disocclusion information from temporal frames by using the motion vectors of H.264/AVC bit stream [28]. However, since motion vectors are sparse and do not correspond to the real motion, quality gains are limited. In other attempts, background sprites are generated by using the temporal frame information, after which disocclusions are filled and updated using the true texture and finally, inpainting is applied to fill holes in virtual views [29–32]. The method provided by Schmeing et al. requires a spatial hole filling for disocclusions caused by static foregrounds [30]. Regardless, reusing the synthesized background from the sprite and inpainting the remaining holes at the moving foreground causes spatial inconsistencies. Another sprite-based hole-filling method presented in [33–35] extracts the static background information using depth classification by probability analysis, a Gaussian mixed model and structural similarity index. Yet, these methods are prone to exhibit spatio-temporal inconsistencies for moving foreground objects. Hsu et al. proposed a method to construct the disocclusions using global optimization [36]. However, the results show spatial inconsistencies at the structured background due to the inpainting method used. Choi et al. used a method where the disocclusions are filled by finding the best patches in the past frames along with current frames [37]. This method yields spatially acceptable results but suffers from flickering artifacts particularly when it comes to the static foreground objects.

Although the above-mentioned temporal hole-filling methods show improvements and new advances, they still suffer from spatial and temporal inconsistencies. These inconsistencies consequently deteriorate the visual quality of the virtual view.

In this paper, we propose a spatio-temporal consistent view synthesis method for extrapolating virtual views. The proposed method produces a layered depth image by identifying the occlusions and inpainting them in the original view using temporal information and motion estimation. When the layered depth image is rendered to the virtual view, no disocclusions appear and views are consistent in the spatial and temporal domain.

This paper is organized as follows: Section 2 explains the proposed spatio-temporal consistent view synthesis with inpainting. The experimental set-up and the evaluation criteria are described in Section 3. Results and analysis of the proposed method are presented in Section 4. Finally, in Section 5, we discuss challenges and conclude the work.

2 Proposed view synthesis method

Temporal inconsistencies are the result of filling holes in a frame irrespective of temporal information. To reduce the temporal inconsistencies, we construct a static scene sprite over time and fill the holes by using a depth-based inpainting method. The sprite is formed such that the occlusion information in the original view is generated in a spatially and temporally consistent manner. Using the occlusion-free sprite, a LDI is generated such that no disocclusions will appear after warping.

In the sprite construction process, we have used a set of future frames, motion estimation, occlusion region, occlusion depth threshold and foreground-background classification. Future frames not only fetch the true information of the holes but also reduce temporal inconsistencies by adding a smoothing constraint. At each time instant t, the sprite makes use of information in all previous frames as well as future frames up to time instant $t + B_l$. The buffer length B_l depends on the applications and must be selected accordingly. Initially, the sprite is constructed by considering the information in the first B_l frames by employing the process described below in this section. Then, the sprite is updated with new information of frame $(t + B_l)$ at each iteration. That is, the sprite always contains information from the B_l future frames. After filling the sprite with this information, there are still holes to be filled. These holes are filled by employing the inpainting procedure described in Section 2.4. The inpainting procedure is rather time-consuming and is therefore carried out every B_l frame. Both for information in a new frame and for inpainted data, the process gradually transforms the existing information with new information such that it is fully replaced at time $t + B_l$. The identified occlusion, occlusion depth threshold and foreground-background classification help in finding the background information to fill the holes from neighboring frames and guide the inpainting process to fill from background. As the global foreground-background classification is not consistent over time, motion estimation helps to construct the static scene sprite. Next, the holes in the sprite are labeled using motion estimation such that holes that share information in the temporal domain are filled with previous inpainted information and the remaining holes are inpainted over a set of frames. This step not only guarantees the temporal consistency but also improves the spatial consistency at moving foregrounds.

The proposed view synthesis method using LDI is shown in Fig. 3. The process of the method is as follows: first, a spatio-temporal consistent LDI is generated. Next, the LDI is warped and combined with the warped image. The final step consists of hole-filling post-processing. The details of each part are presented in the following sections. The LDI generates the occlusion layers in the original view, where the occlusions and depth thresholds from V+D are identified using the work of [27].

2.1 Identification of occlusion and depth threshold

The LDI generation consists of identifying occlusion layers and inpainting them in the original view before warping. Given the properties of DIBR, occlusions belong to the background and are identified using the depth values and the projected locations of pixels in the virtual view.

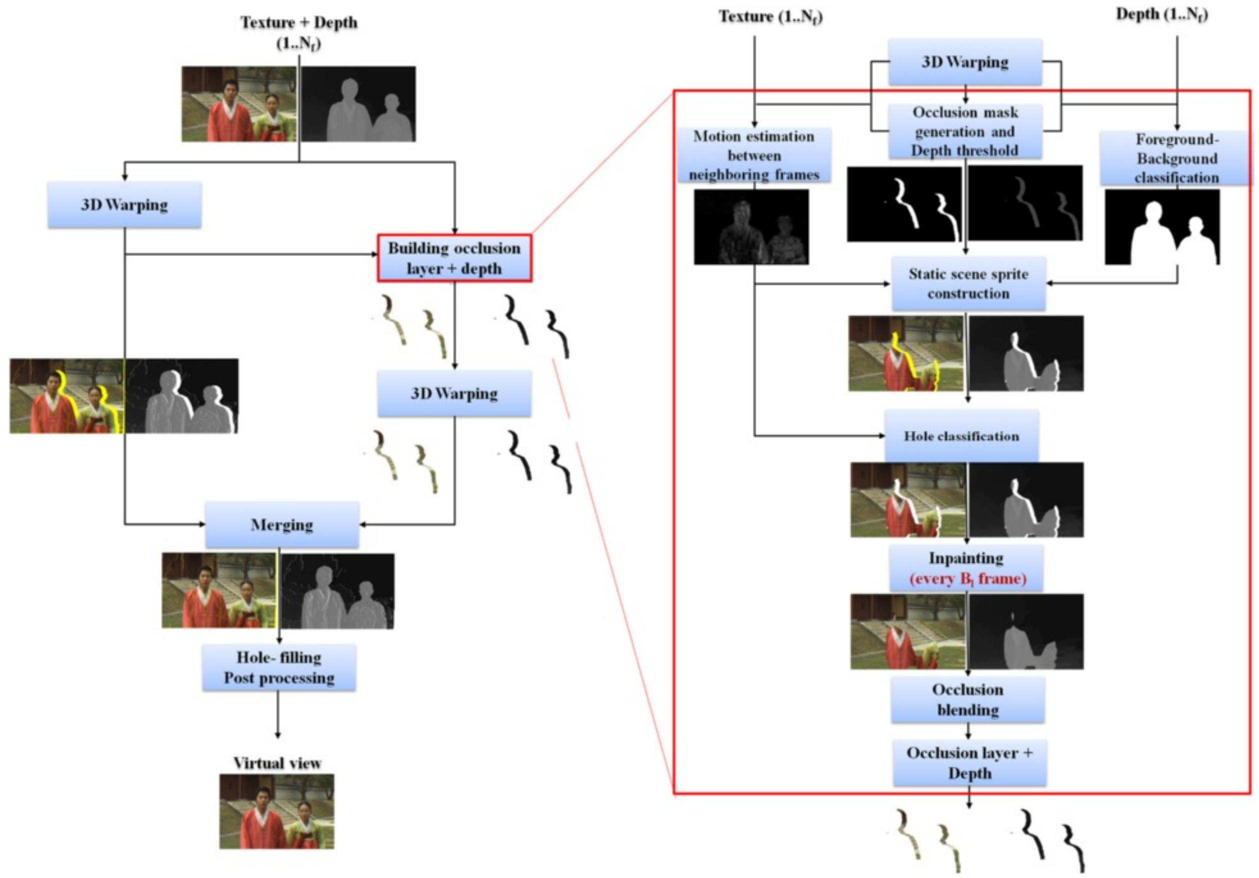
Texture + Depth
(1..N_f)
3D Warping
Building occlusion layer + depth
3D Warping
Merging
Hole- filling Post processing
Virtual view
Texture (1..N_f)
Depth (1..N_f)
3D Warping
Motion estimation between neighboring frames
Occlusion mask generation and Depth threshold
Foreground-Background classification
Static scene sprite construction
Hole classification
Inpainting
(every B_l frame)
Occlusion blending
Occlusion layer + Depth

Fig. 3 Proposed view synthesis method. Spatio-temporal consistent view snythesis using LDI and inpainting

The occlusions and the corresponding depth thresholds are identified as follows:

2.1.1 *Depth discontinuity pixel identification*

Disocclusions occur between pixels that are neighboring in the original image, have a distinct difference in depth, and that become separated in the warped image. For example, the occlusion behind a foreground object in the original image becomes visible as a disocclusion in the virtual image. A disocclusion appears to the right of the foreground object (see Fig. 4), when the virtual image is rendered from a camera positioned to the right of the original camera.

Let Λ be the warping operation that projects the original view image $\mathrm{I_o}$ to the virtual view image $\mathrm{I_v}$. We define a depth discontinuity pixel (DDP) to be a neighboring pixel pair $\{\mathbf{f}_\mathrm{o}, \mathbf{p}_\mathrm{o}\}$ positioned in the original view as shown in Fig. 4 that satisfy the following conditions:

$$\mathrm{DDP} = \{\mathbf{f}_\mathrm{o}, \mathbf{p}_\mathrm{o}\}\,;\, \|\mathbf{d}_\mathrm{v}\|_2 > \eta, \tag{1}$$

where η is an occlusion threshold, and $\mathbf{d}_\mathrm{v}$ is a displacement vector given by

$$\mathbf{d}_\mathrm{v} = \begin{pmatrix} d_x \\ d_y \end{pmatrix} = \mathbf{f}_\mathrm{v} - \mathbf{p}_\mathrm{v}, \tag{2}$$

where $\mathbf{f}_\mathrm{v}$, $\mathbf{p}_\mathrm{v}$ are projected pixels in $\mathrm{I_v}$:

$$\begin{aligned} \Lambda &: \mathbf{f}_\mathrm{o} \rightarrow \mathbf{f}_\mathrm{v}, \\ \Lambda &: \mathbf{p}_\mathrm{o} \rightarrow \mathbf{p}_\mathrm{v}. \end{aligned} \tag{3}$$

The above definition of DDP is presented within the context of a right warped view. It is straightforward to change Eq. (1) for a left warped image. The summary of the notation used in this paper is presented in Table 1.

In the next step, the pixels in the DDP are labeled foreground and background using their respective depth values. When there are multiple depth layers, DDPs might overlap and produce inconsistent foreground-background labeling, such that a pixel-labeled background in one DDP might be labeled foreground in another DDP. To circumvent inconsistent foreground-background labeling, we apply a simple labeling refinement step that selects either background or foreground based on the position difference of the projected pixels, that is, the pixel in DDP

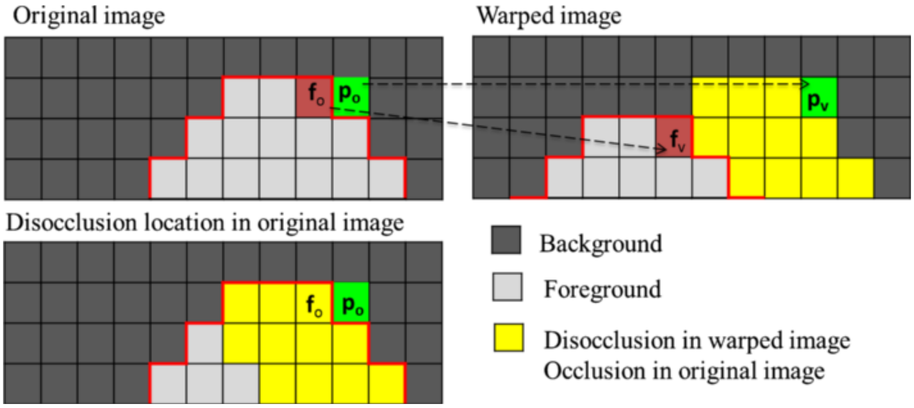

Fig. 4 Identification of occlusion location

is labeled as background when its projected pixels difference is less than its neighbor pixel in the overlapped DDP.

2.1.2 Occlusion identification

After identifying and labelling the DDPs, occlusions are located in the original image by horizontal and vertical displacements and their corresponding depth values. The occlusions at the DDP $\{\mathbf{p}_\mathrm{o}, \mathbf{f}_\mathrm{o}\}$ are identified as the following set:

$$\Omega_{O(\mathrm{DDP})} = \{\mathbf{p}; \mathbf{p} \in l \wedge (\mathrm{Z}(\mathbf{p}) > \xi)\} \tag{4}$$

where l is a line joining pixels $\mathbf{p}_\mathrm{o}$ and $\mathbf{r}_\mathrm{o} = \mathbf{p}_\mathrm{o} - \mathbf{d}_\mathrm{v}$, and ξ is an occlusion depth threshold, given by the arithmetic mean over the DDP depths:

$$\xi = \frac{\mathrm{Z}(\mathbf{f}_\mathrm{o}) + \mathrm{Z}(\mathbf{p}_\mathrm{o})}{2}, \tag{5}$$

The full occlusion in the image consists of all such pixels:

$$\Omega_O = \bigcup_{\text{all DDP}} \Omega_{O(\mathrm{DDP})} \tag{6}$$

An example of an occlusion is shown in Fig. 5d.

2.1.3 Occlusion depth threshold

Identified occlusions belong to the background and should be filled with consistent background information. Hence, two depth threshold images are created, Z_A and Z_B, which contain the average depth, and background depth values of the DDPs at occlusion areas, respectively:

$$\left.\begin{array}{l} \mathrm{Z}_A(\mathbf{q}) = \xi, \\ \mathrm{Z}_B(\mathbf{q}) = \mathrm{Z}(\mathbf{p}_\mathrm{o}), \end{array}\right\} \quad \forall \mathbf{q} \in l, \tag{7}$$

where l is a line defined in Eq. (4). Note that pixels on different lines l are calculated from different DDPs. Examples of Z_A and Z_B are shown in Fig. 5e, f.

Table 1 Notation

Variable	Description
f,p,q	Pixels in an image
I	Texture image
Z	Depth map
Ω	Hole region
$\delta\Omega$	Hole boundary
Φ	Source region (I − Ω)
Ψ	2D patch

2.2 Static scene sprite construction

The static scene sprite construction step builds a global sprite during the view synthesis process by collecting texture and depth information of the static scene from temporal frames. The global sprite consists of both the texture I_S and depth Z_S. The construction process continues for a set of frames. The static information from the set of frames is extracted and stored in the constructed sprites I_S and Z_S. Example of texture sprite at different time instants is shown in Fig. 6. The remaining holes in the sprites are filled using the inpainting process.

The global sprite formation is guided by the following parameter specific for the sprite: The occlusions Ω_{OS} indicate the holes, the occlusion threshold Z_{AS} indicates allowable depth values to fill the holes, the global foreground-background classification Ω_{FS} identifies the absolute foreground-background and finally the motion pixels Ω_{MS} indicate the moving objects.

We use a simple depth map segmentation on the whole image to identify the background in order to form the sprite with the background information [38]. The Otsu method selects an optimal threshold by maximizing the class variance and it works well when the entire distribution is segmented into two classes. We denote the identified foreground pixels as Ω_F.

We want all static information to be part of the sprite. Therefore, motion is estimated to detect the moving

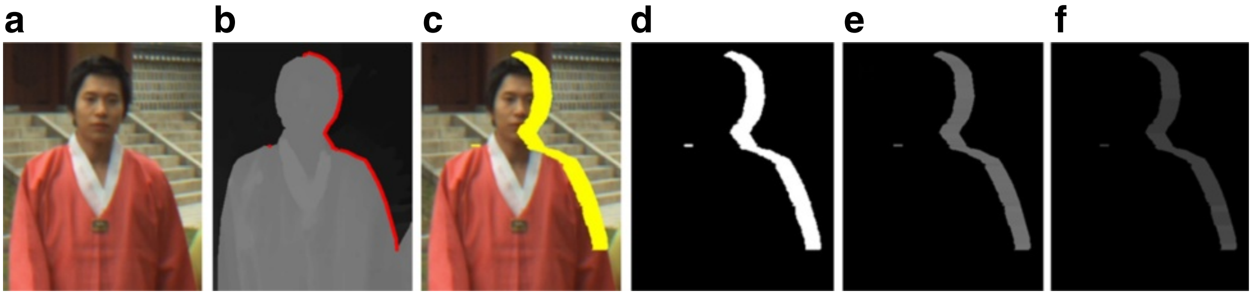

Fig. 5 Occlusion layer identification and depth threshold generation. **a** Original image. **b** Depth discontinuity pixels (DDPs) (in *red color*). **c** Occlusion in original image (in *yellow color*). **d** Occlusion region. **e** Average depth threshold image. **f** Background depth threshold image. Note that the depth values in the threshold images are scaled for illustration purpose

objects in the scene. The optical flow describes the apparent motion of objects, surfaces and edges between images, so we use the Lukas-Kanade algorithm, a simple and efficient method [39]. The motion pixels are given by

$$\Omega_M = \{\mathbf{p}; |\mathrm{V}(\mathbf{p})| > \rho\} \tag{8}$$

where $|\mathrm{V}(\mathbf{p})|$ is the absolute optical flow value at $\mathbf{p}$. ρ is the optical flow threshold, which allows to change the noise sensitivity of the optical flow calculations.

The static sprite is formed in three steps: (i) Hole regions in the sprite are filled with the future frame using the occlusion depth threshold Z_A. (ii) The foreground and moving information in the sprite are replaced by background, static foreground and motion information. (iii) Finally, the holes in the upcoming frame are located in the sprite using the depth threshold Z_A. In the sprite construction process of the static scene, the sprite is initialized with information from the first frame ($t = 0$).

$$\begin{aligned} \mathrm{I}_S &:= \mathrm{I}^0, \\ \mathrm{Z}_S &:= \mathrm{Z}^0, \\ \mathrm{Z}_{AS} &:= \mathrm{Z}_A^0, \\ \Omega_{OS} &:= \Omega_O^0, \\ \Omega_{MS} &:= \Omega_M^0, \\ \Omega_{FS} &:= \Omega_F^0. \end{aligned} \tag{9}$$

Once the sprite is initialized, the occlusions should be filled with new texture as it becomes available. Firstly,

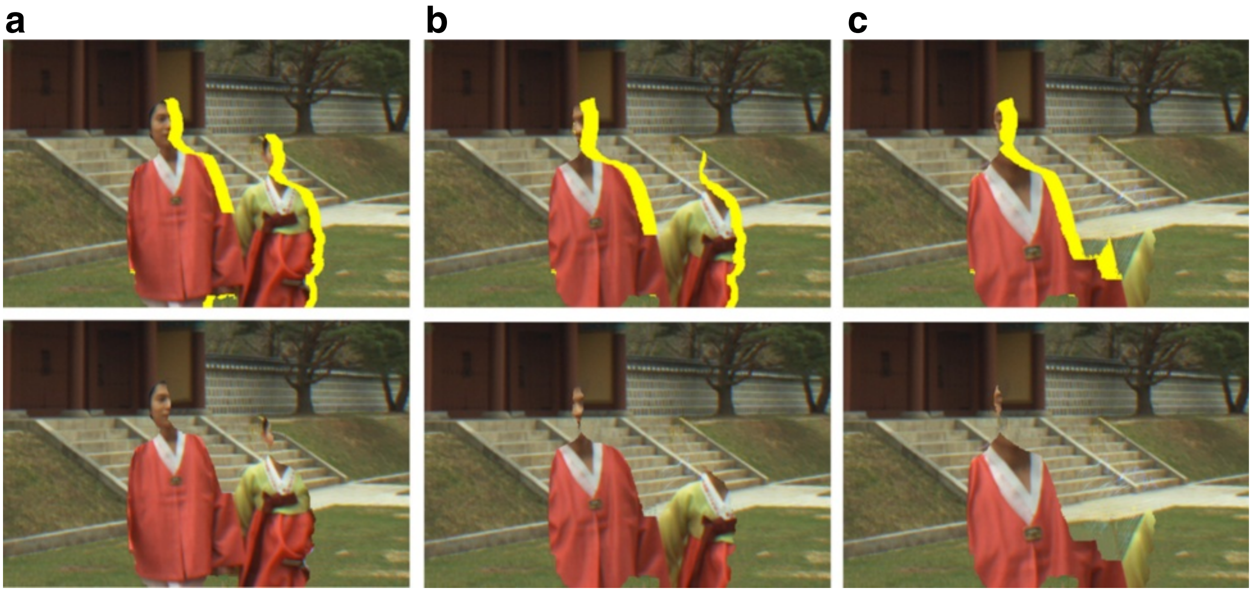

Fig. 6 Static scene sprites at different time instants. A static scene sprite is initialized with the first frame and updated with background information from subsequent frames at holes and foreground regions. Sprites after locating holes are in first row and sprites after inpainting are in second row; **a** at frame 6; **b** at frame 46; **c** at frame 96

the holes are filled with static background from the new frame:

$$\left.\begin{aligned} \mathrm{I}_S(\mathbf{p}) &:= \mathrm{I}^t(\mathbf{p}) \\ \mathrm{Z}_S(\mathbf{p}) &:= \mathrm{Z}^t(\mathbf{p}) \end{aligned}\right\} \forall \mathbf{p} \in \Omega_S \backslash \Omega_O^t; \mathrm{Z}^t(\mathbf{p}) < \mathrm{Z}_{AS}(\mathbf{p}) \tag{10}$$

The set of occlusion pixels in the sprite Ω_{OS} are then updated by removing the pixels considered in Eq. (10). Next, the moving object pixels and foreground pixels in the sprite are replaced by information in the new frame:

$$\left.\begin{aligned} \mathrm{I}_S(\mathbf{p}) &:= \mathrm{I}^t(\mathbf{p}) \\ \mathrm{Z}_S(\mathbf{p}) &:= \mathrm{Z}^t(\mathbf{p}) \end{aligned}\right\} \begin{aligned} &\forall \mathbf{p} \in \Omega_{MS} \cup \Omega_{FS}; \\ &\mathrm{Z}^t(\mathbf{p}) < \mathrm{Z}_S(\mathbf{p}) - \vartheta, \end{aligned} \tag{11}$$

where ϑ is depth tolerance, which is set according to the variations of the scene depth. The set of motion and foreground pixels in the sprite, Ω_{MS} and Ω_{FS}, are then updated with the corresponding pixel information in the new frame.

Finally, the average depth threshold in the sprite Z_{AS} is updated with the corresponding information of holes in the new frame Ω^t:

$$\mathrm{Z}_{AS}(\mathbf{p}) = \frac{\mathrm{Z}_S(\mathbf{p}) + \mathrm{Z}_A^t(\mathbf{p})}{2}, \forall \mathbf{p} \in \Omega_O^t; \mathrm{Z}_A^t(\mathbf{p}) > \mathrm{Z}_S(\mathbf{p}), \tag{12}$$

whereby the set of occlusion pixels in the sprite, Ω_{OS} is updated with the corresponding pixel information in the new frame.

An example of texture sprite at different time instants is shown in Fig. 6. The remaining holes in the sprites are filled using the inpainting process.

2.3 Hole classification

Holes in each frame are classified as static and dynamic by using the motion of the foreground objects (see Fig. 7e). Holes which appear at the moving foreground objects are labeled dynamic holes. These holes are filled with either true texture from consequent frames or by the inpainting process. Holes which appear at static foreground objects are labeled static holes. Using this classification, static holes are filled only once in the entire view synthesis process and are placed in the sprite to be reused when needed. In contrast, dynamic holes are inpainted over a set of frames in order for the inpainted texture to be spatially and temporally consistent with the background. As not all motion pixels belong to the foreground object that causes holes, only the pixels that belong to the dynamic holes are given by

$$\Omega_D = \left\{\mathbf{p}; \mathbf{p} \in \Omega_{O(n)} \wedge (\nu_n > \rho)\right\} \tag{13}$$

$$\nu_n = \frac{1}{\Gamma} \sum_{\mathbf{p} \in E_n} \max\left(\left|\mathrm{V}_{\mathbf{p}}\right|\right), \tag{14}$$

where $\Omega_{O(n)}$ is nth hole in a occlusion region. E_n is a set that contains the DDPs corresponding to the nth hole. ν_n is the average of maximum optical values in the patch centered at DDPs on the Ω_n in a frame (see Fig. 7b). Γ is the number of DDPs in E_n.

Once the hole classification for the frames in the buffer are obtained, then the dynamic holes in the sprites are given by $\Omega_{DS} = \Omega_{OS} \cap \Omega_D$.

2.4 Inpainting

The holes remaining in the sprite, after compensating the true texture from the temporal information, are divided into small and large holes. This is because small holes will have no inpainting priority due to the foreground presence within a patch. As a result, small holes will not be filled until after the large holes have been filled. This implies inconsistent selection and filling of large holes when they lie within a search region of a given hole to find a best match. Therefore, small holes with a size smaller than a patch area are filled using the interpolation method. Then inpainting is applied on the large holes, where the inpainting method is adopted from [27], which is derived from [17]. Note that the inpainting method is applied on the sprite instead of each frame. The steps of the inpainting process are as follows:

1. Identify the background boundary of the holes using the hole depth threshold Z_{AS}.

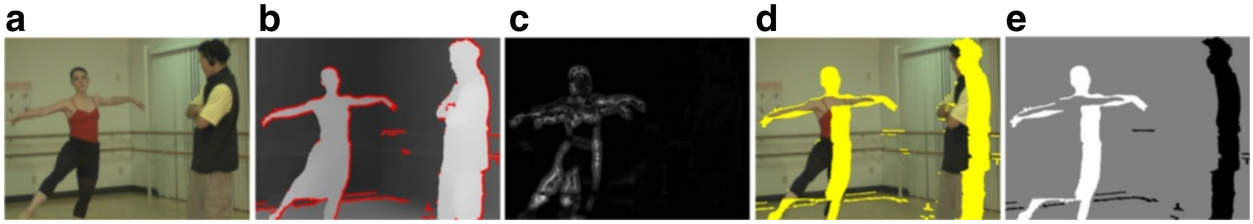

Fig. 7 Hole classification using optical flow. **a** Texture. **b** Depth image DDPs (in *red color*). **c** Optical flow image (*brighter region* indicates motion and *black regions* indicate no motion). **d** Occlusions in texture (in *yellow color*). **e** Hole classification region (*white areas* indicate dynamic holes and *black* indicates static holes)

2. Compute the priority on the background boundary using the curvature data term and depth-based confidence term and select a patch with the highest priority.
3. Find a good match to the target patch in both texture and depth. Fill holes in the target patch with a weighted combination of the best patches.
4. Update the terms and repeat the above steps until the holes in the sprite are filled.

In the next sections, we will give further details on the inpainting steps.

2.4.1 Foreground-background boundary extraction

The background boundary of the hole is identified by classifying the hole boundary into the foreground and background using the depth threshold image Z_{AS}. The classification proceeds as follows: Initially, the hole boundary $\delta\Omega$ is obtained by applying a Laplacian convolution on the hole region. Next, the hole boundary is classified into the foreground boundary $\delta\Omega_F$ and the background boundary $\delta\Omega_B$, i.e. $\delta\Omega = \delta\Omega_F \cup \delta\Omega_B$.

Let us begin by defining the maximal value of the occlusion threshold on the 2D patch at point $\mathbf{p}$, $\Psi_\mathbf{p}$, coinciding with the source region Φ:

$$\mu = \max\left(Z_{AS}(\mathbf{q})\,|_{\mathbf{q}\in(\Psi_\mathbf{p}\cap\Omega)}\right) \tag{15}$$

$$\delta\Omega_F = \delta\Omega \;\; \forall\mathbf{p} : Z_S(\mathbf{p}) > \mu, \tag{16}$$

$$\delta\Omega_B = \delta\Omega - \delta\Omega_F, \tag{17}$$

2.4.2 Priority

The priority term on the background boundary is the product of confidence and data terms:

$$P(\mathbf{p}) = C_z(\mathbf{p}) \cdot D_k(\mathbf{p}), \tag{18}$$

where $C_z(\mathbf{p})$ is the depth-based confidence term, and $D_k(\mathbf{p})$ is the curvature data term at $\mathbf{p}$.

Depth-based confidence term computes the percentage of background pixels in a patch.

$$C_z(\mathbf{p}) = \frac{1}{|\Psi_\mathbf{p}|} \sum_{\mathbf{q}\in\Psi_\mathbf{p}\cap\Phi} C(\mathbf{q}), \tag{19}$$

$$C(\mathbf{q}) = \begin{cases} 1, & \mathbf{q} \in \Phi_B; \\ 0, & \text{otherwise.} \end{cases} \tag{20}$$

where Φ_B is the background source region, which is given by

$$\Phi_B = \{\mathbf{p}; \mathbf{p} \in \Phi \wedge (Z_S(\mathbf{p}) < \mu)\} \tag{21}$$

Curvature data term finds the structure details in the neighborhood of the patch using the curvature, where it consists of both geometry and strength of the isophote. The data term gives the priority to the patches that consists of linear structures, which is given as

$$D_k(\mathbf{p}) = 1 - |k_\mathbf{p}|. \tag{22}$$

$$k_\mathbf{p} = \nabla \cdot \left(\frac{\nabla \mathrm{I}_\mathbf{p}}{|\nabla \mathrm{I}_\mathbf{p}|}\right), \tag{23}$$

where $k_\mathbf{p}$ is the curvature of the isophote through a pixel $\mathbf{p}$, $\nabla\cdot$ is the divergence at $\mathbf{p}$ that gives a total isophote flow from $\mathbf{p}$.

2.4.3 Patch matching and filling

To avoid the patch selection from the foreground during the inpainting process, the source region is classified into the foreground-background locally. The threshold for the classification is selected from the previously generated threshold image Z_{AS}. Similar to [23], the block matching is performed with only background pixels in the target patch by removing foreground pixels in order to avoid artifacts at foreground objects.

$$\Phi_F = \Phi - \Phi_B, \tag{24}$$

where Φ is the source region and subscripts F and B denote the foreground and background, respectively. Φ_F is the foreground source region, whose depth values are larger than the depth threshold μ. Thus, the block matching equation becomes

$$\Psi_\mathbf{q} = \arg \min_{\Psi_\mathbf{q}\in\Phi_B} \left\{ \left\|\Psi_{\hat{\mathbf{p}}} - \Psi_\mathbf{q}\right\|_2^2 + \beta \cdot \left\|Z_{\hat{\mathbf{p}}} - Z_\mathbf{q}\right\|_2^2 \right)\right\}, \tag{25}$$

where $\Psi_{\hat{\mathbf{p}}}$ and $Z_{\hat{\mathbf{p}}}$ are the target texture and depth patches, and $\Psi_{\hat{\mathbf{q}}}$ is the source patch. $\left\|\Psi_{\hat{\mathbf{p}}} - \Psi_\mathbf{q}\right\|_2^2$ is the sum of squared difference between known pixels in two patches and β is a weighting coefficient to equalize the effect of the depth and texture. Similar to [22], holes in the depth map are filled by copying the depth values from the best patch location. Holes in the texture image are filled with a weighted average of N patches of texture values.

2.5 Moving weighted averaging and blending

Although the static scene construction process recovers the true texture from the temporal information of the set of future frames, the scene and holes could change for each buffer. Thus, sudden changes could occur in the construction of the sprite. Sudden changes (flickering) appear in the texture content in the following cases: (i) available true texture in the current frame for the holes that are inpainted in the previous frame; (ii) holes in the current frame and previous frame are inpainted separately. To avoid flickering that occurs at the occlusion areas, a moving weighted average blending is introduced. This creates a smooth transition over frames at hole regions. The blending of textures is done as follows:

$$\mathrm{I}^{t+i}_{WS}(\mathbf{p}) = (1-\omega_i)\mathrm{I}^{t}_{S}(\mathbf{p}) + \omega_i \mathrm{I}^{t+B_l}(\mathbf{p}), \forall \mathbf{p} \in \Omega^{t}_{OS} \cap \Phi^{t+B_l} \tag{26}$$

$$\omega_i = \frac{t \mod B_l}{B_l} \tag{27}$$

where, $\mathrm{I}^{t}_{S}(\mathbf{p})$, $\mathrm{I}^{t+B_l}(\mathbf{p})$ are inpainted texture in the sprite at frame t and true texture in the frame $t + B_l$ at pixel $\mathbf{p}$. $\mathrm{I}^{t}_{WS}(\mathbf{p})$ is a blended texture image. Φ^{t+B_l} is the source region in frame I^{t+B_l} and Ω^{t}_{OS} is the hole region before filling the sprite at frame t. ω_i is the weight that linearly combines future frames, and i is an index number within the set.

By using the sprite, a spatially and temporally consistent layered depth image is generated, which is later used to recover the disocclusions after warping.

2.6 Warping layered depth image and merging

To avoid artifacts after projection, layers are warped using background depth values of occlusion areas, which are stored in Z_B. Then, the warped occlusion layers are merged with the warped image using Z-buffering (the pixel closest to the camera is selected when two pixels are projected at the same location).

2.7 Crack removal and hole-filling post-processing

By using the LDI from the sprite, spatial and temporal consistency is achieved at the disocclusions in the virtual view. However, certain types of holes could still be present after a virtual view is produced by warping, making hole-filling post-processing necessary. These holes are a result of the forward warping and are manifested as cracks and out-of-field areas. Typically, depth maps have significantly varying depth values surrounding the occlusion area, which will exhibit minor depth discontinuities in the filled area. These discontinuities are also manifested as disocclusions with an extent that is slightly larger than cracks but significantly smaller than conventional disocclusions. These artifacts are removed by using hole-filling post-processing.

Similar to the hole classification used in the sprite filling, small holes are filled by using the interpolation method. The remaining holes are filled by using the proposed inpainting method, operating without depth classification. This is because the threshold for classification is not identified, since the foreground is not available during the occlusion identification step. Out-of-field areas occur at image borders and are static and so they are filled only once and then reused for the subsequent frames in such a way that temporal consistency is achieved.

3 Experimental set-up

We use four MVD test sequences to evaluate the performance of the proposed view synthesis method. The test sequences are "Ballet" [40], "Book Arrival" [41], "Lovebird1" [42] and "Poznan Street" [43], which cover various texture and depth characteristics (see Table 2) to validate our method in different conditions. The hole size in the table is computed as follows: first, the percentage of hole pixels is calculated for each warped image; then the average of the calculated percentage is computed over the sequence. Note that only the static camera test sequence is used, since the proposed method is limited to static scenes. MVD sequences are used in the evaluation, for the purpose of measuring the quality of the virtual view using full reference objective metrics. We rendered extrapolated virtual views using V+D at the available real camera views, so that rendered views match the ground truth textures in the MVD, as a pre-requisite for measuring full reference quality evaluations.

In the quality evaluations of our proposed method, we use both objective measurements and visual inspection. As the proposed method is aimed at the spatial and temporal consistency of the virtual views, we divide the quality evaluation set-up into two parts; spatial consistency evaluation and temporal consistency evaluation.

1. Spatial consistency evaluation
2. Temporal consistency evaluation

Table 2 Test input data characteristics

Sequence name	Resolution	Frames	Camera arrangement	Texture background	Depth properties	Hole size (Average)
Ballet	1024×768	1–100	8 cameras with 30-cm spacing, 1D arc arrangement	Low structured	Large depth discontinuities and many layers	13.5 %
Book Arrival	1024×768	1–100	16 cameras with 6.4-cm spacing, 1D arrangement	Low structured	Small depth discontinuities and many layers	6.4 %
Lovebird1	1024×768	50–200	12 cameras with 3.5-cm spacing, 1D arrangement	Structured	Large depth discontinuities and few layers	4.3 %
Poznan Street	1920×1088	1–100	9 cameras with 13.5-cm spacing, 1D arrangement	Structured	Small depth discontinuities and many layers	5 %

3.1 Spatial consistency evaluation

Inpainting is an ill-posed problem, and so investigating the inpainting quality is as complex as the disocclusion problem. Hence, we used the commonly used evaluation methodology in the view synthesis context. We use full reference objective metrics peak signal-to-noise ratio of the luminance component (YPSNR) and mean structural similarity index (MSSIM) to evaluate the spatial consistency of virtual views. PSNR measures the absolute difference between images, whereas MSSIM measures structural similarities between images and closely corresponds to the perceptual visual quality [44]. In line with commonly used evaluation methodology of view synthesis, both metrics are applied on the full image.

3.2 Temporal consistency evaluation

Temporal consistency of the virtual view is evaluated by measuring the amount of flickering. We adopt the method in [45] where the amount of flickering is computed as

$$\tau_D^t = \frac{1}{\left|\Omega^t\right|}\sum_{\mathbf{p}\in\Omega^t}\left|\mathrm{I}^t\left(\mathbf{p}\right) - \mathrm{I}^{t-1}\left(\mathbf{p}\right)\right|, \forall t = 2\dots N_f \tag{28}$$

where τ_D^t is the inter-frame average absolute difference at holes in frame t, N_f is the number of frames, $\left|\Omega^t\right|$ is the number of hole pixels in the frame t and the flicker measured between the frames is given as

$$\text{F-score} = \left|\tau_{\mathrm{Dg}}^t - \tau_{\mathrm{Dv}}^t\right| \tag{29}$$

where τ_{Dg}^t and τ_{Dv}^t represent the inter-frame average absolute difference at holes of the ground truth and the virtual view at frame t, respectively. A lower F-score indicates less flickering. Note that flickering is only measured at the inpainted holes.

In addition to the objective spatial and temporal consistency measurements, synthesized views are also presented to allow qualitative visual comparison.

3.3 Complexity evaluation

The proposed method is implemented in MATLAB without any optimization, which is not the case for reference methods which are implemented in different platforms. However, few insights about the processing times are given. Note that the main objective is to improve the quality of the rendered view.

We use five of the state-of-the-art methods for comparison purposes. The comparison is carried out with various view synthesis methods that use diffusion and exemplar inpainting to fill holes in spatial and temporal domains. MPEG VSRS version 3.5 (M1) is a commonly known view synthesis method, employing several tools to handle artifacts, used to render virtual views [15]. Diffusion-based inpainting is one of the tools used for rendering. Other recent view synthesis methods presented by Ahn et al. [21] and Choi et al. [37] are based on exemplar inpainting. We denote these methods as M2 and M3, respectively. Method M2 applies inpainting by using only spatial information, whereas M3 applies inpainting by using both spatial and temporal information. Moreover, methods M2 and M3 are similar to our inpainting, since M2 and M3 uses exemplar-based inpainting. The remaining two methods are VSRS-alpha-Gist and VSRS-apha-Etri, which include different hole-filling methods and these methods are developed for view extrapolation scenarios [46, 47]. The methods VSRS-alpha-Gist and VSRS-apha-Etri are denoted as M4 and M5, respectively. All the state-of-the art methods are used in the traditional scenario, where hole filling is performed after warping to the virtual view. To make a fair comparison, and since we only have access to the low-resolution results of a few datasets from M3, we downscale the "Ballet" and "Poznan Street" sequence results during comparison.

In the notation used in the article, e.g. v4→v3 indicates that the original camera view 4 is warped to the virtual camera view 3. Parameter values used in the algorithm are empirically selected as follows: optical flow threshold $\rho = 0.05$ and depth tolerance $\vartheta = 10$ is used in the sprite construction step. The number of future frames $B_l = 5$ is used. Similar to [27], the following parameters are used in the inpainting process: the occlusion threshold $\eta = 5$ for a small baseline, a search region of 120×120 pixels, a patch

Table 3 Objective spatial quality evaluation

Sequence	YPSNR						**SSIM**					
	Proposed	M1	M2	M3	M4	M5	Proposed	M1	M2	M3	M4	M5
Poznan Street v3→v1	29.56	29.49	*29.60*	25.89	29.09	29.36	*0.8921*	0.8885	0.8892	0.8225	0.8858	0.8892
Ballet v4→v3	*30.77*	26.05	29.87	27.40	27.68	25.38	*0.9363*	0.9097	0.9268	0.8572	0.9205	0.9089
Ballet v3→v4	*29.32*	27.18	28.78	-	27.34	26.37	0.8499	0.8534	0.8456	-	*0.8590*	0.8511
Lovebird1 v6→v8	*27.63*	27.56	27.29	-	27.30	27.52	0.8693	*0.8747*	0.8653	-	0.8722	0.8733
Lovebird1 v8→v6	*28.56*	28.29	27.92	-	27.86	28.25	*0.8707*	0.8702	0.8651	-	0.8673	0.8693
Book Arrival v8→v10	*25.29*	25.00	25.10	-	18.75	25.51	0.8636	*0.8662*	0.8580	-	0.8472	0.8635

Values in italics indicates the highest score values

Table 4 Objective temporal quality evaluation: average F-score

Sequence	F-score					
	Proposed	M1	M2	M3	M4	M5
Poznan Street v3→v1	*0.33*	1.32	7.78	1.90	2.17	1.11
Ballet v4→v3	*0.65*	2.19	3.70	1.56	1.74	1.76
Ballet v3→v4	*0.54*	2.37	4.74	-	1.07	2.13
Lovebird1 v6→v8	*0.32*	2.72	6.47	-	2.05	2.93
Lovebird1 v8→v6	*0.50*	2.57	7.58	-	2.17	2.75
Book Arrival v8→v10	0.62	*0.33*	13.67	-	0.69	1.32

Values in italics indicates the highest score values

size of 11×11 pixels, number of best patches $N = 5$, and $\beta = 3$ in (25). The number of occlusion layers identified and inpainted is one.

4 Results and analysis

4.1 Spatial consistency evaluation

The objective spatial consistency evaluation results are presented in Table 3. Objective measurement results demonstrate that the proposed method performs better than the state-of-the art reference methods. In addition to the objective results, a set of images is shown in Figs. 9, 10, 11 and 12 for visual inspection. As method M1 fills the holes using the structured based inpainting without boundary classification, results exhibit foreground propagation and blurring artifacts (see row 2 in Figs. 9, 10, 11 and 12). Although method M2 used the background classification while inpainting the holes, holes are still filled with inconsistent background textures due to foreground-background classification (see row 3 in Fig. 9 at the woman's legs, in Fig. 11 at stairs and in Fig. 12 at wall clock). Method M3 uses the temporal information to fill the holes, however, there are still artifacts: such as the artifacts at the holes near moving objects (see row 4 in Fig. 9 at holes near the woman), the foreground propagation into the holes near moving objects and inconsistent filling at static objects (see row 4 in Fig. 10 at holes near the woman and the pole in front of the car). Methods M4 and M5 are aimed at filling holes in view extrapolation; however, holes are filled with smoothed textures and structure discontinuity (see rows 5 and 6 in Fig. 9, Fig. 10 rows 4 and 5 in Fig. 11 and Fig. 12).

The proposed method results in Figs. 9, 10, 11 and 12 show that holes are filled with spatial consistent background textures when the holes belong to the absolute background. However, the proposed method is limited when it comes to filling the holes when they occur between relative foreground and background. In this case, holes are filled with absolute background textures since the proposed method does not classify the absolute and

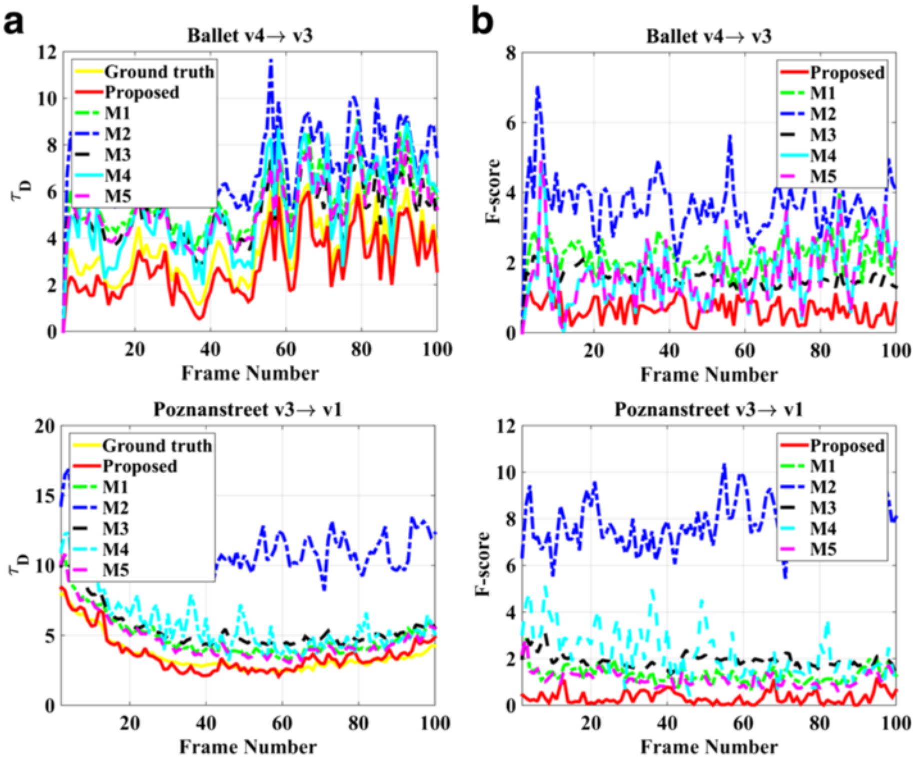

Fig. 8 Temporal objective quality evaluation. **a** Inter-frame average absolute difference at holes (τ_D). **b** Flickering score (F-score)

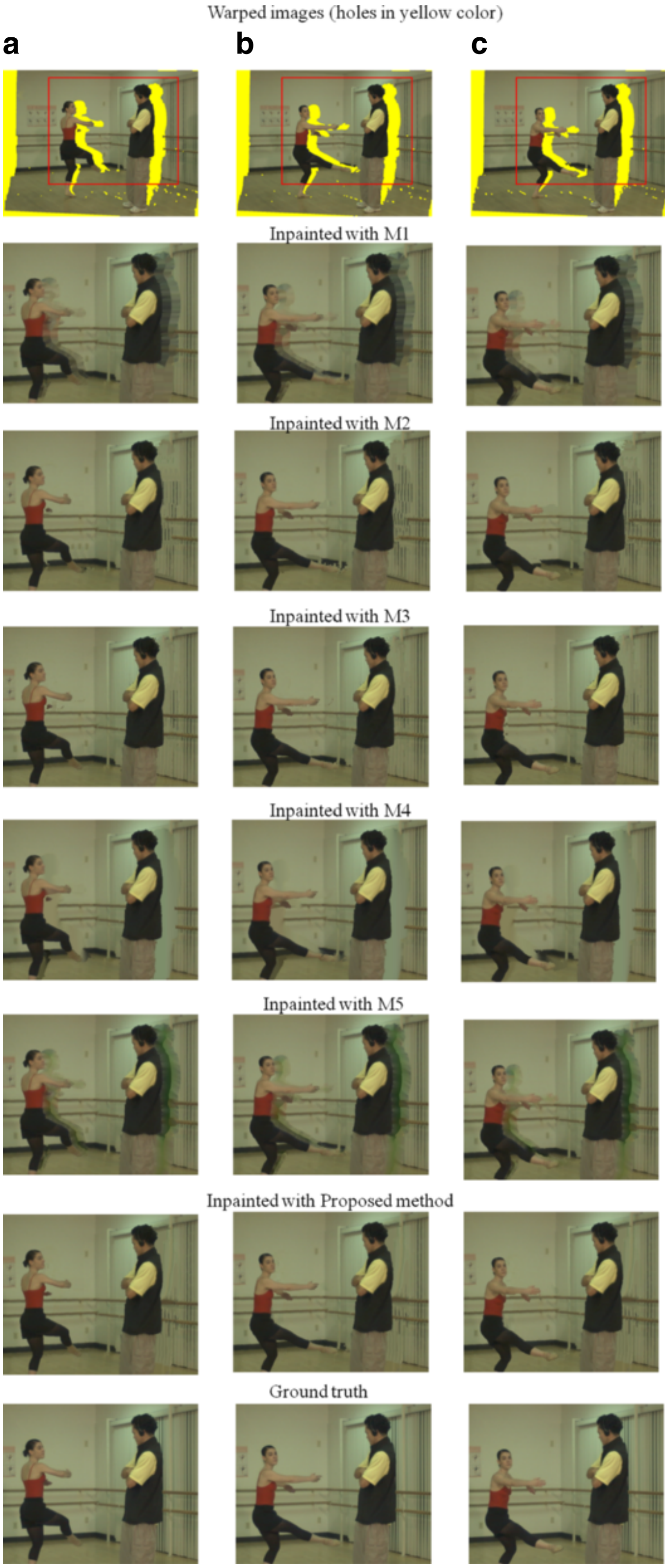

Fig. 9 Visual quality results of "Ballet" v4→v3 sequential frames. **a** Frame 66. **b** Frame 67. **c** Frame 68

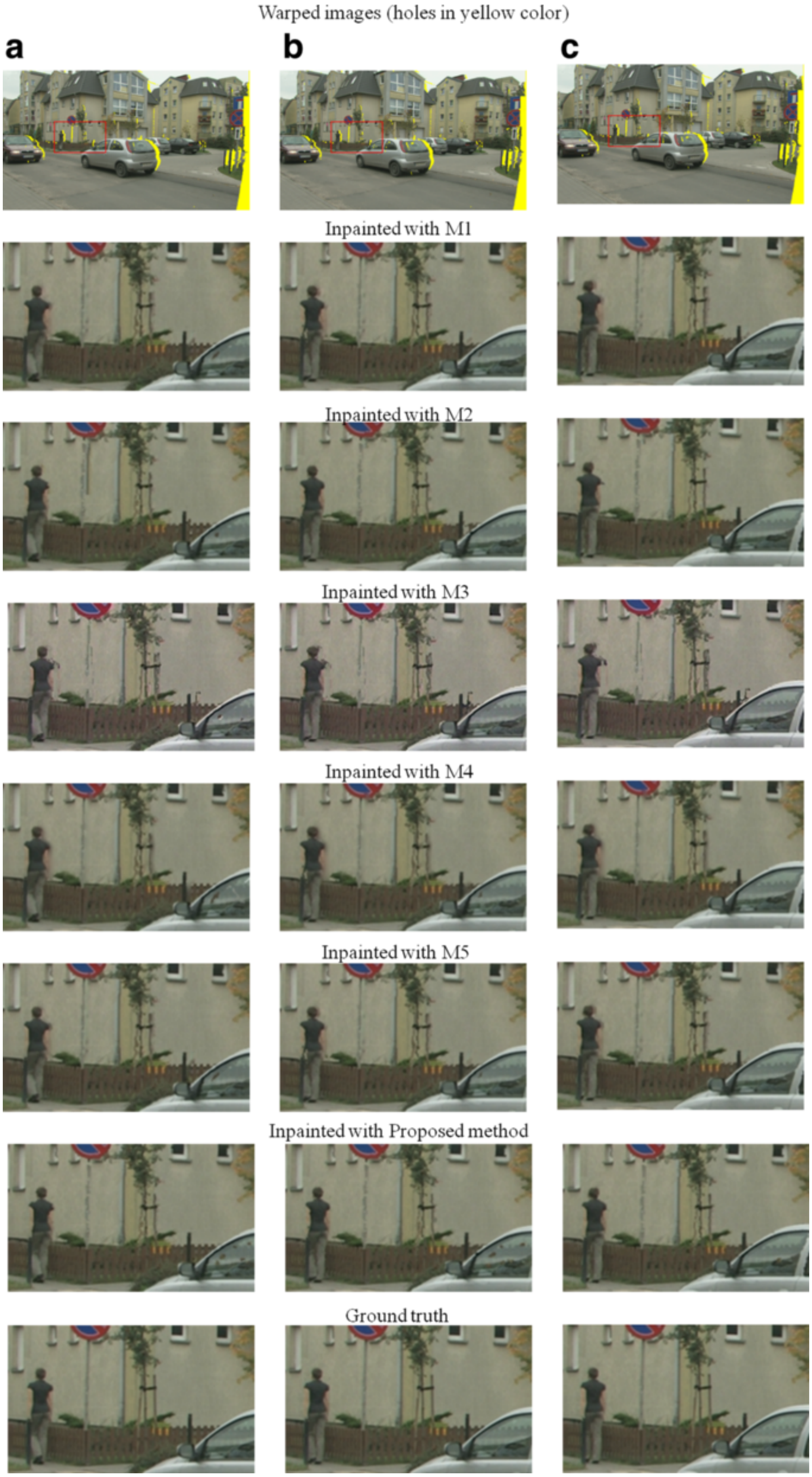

Fig. 10 Visual quality results of "Poznan Street" v3→v1 sequential frames. **a** Frame 18. **b** Frame 19. **c** Frame 20

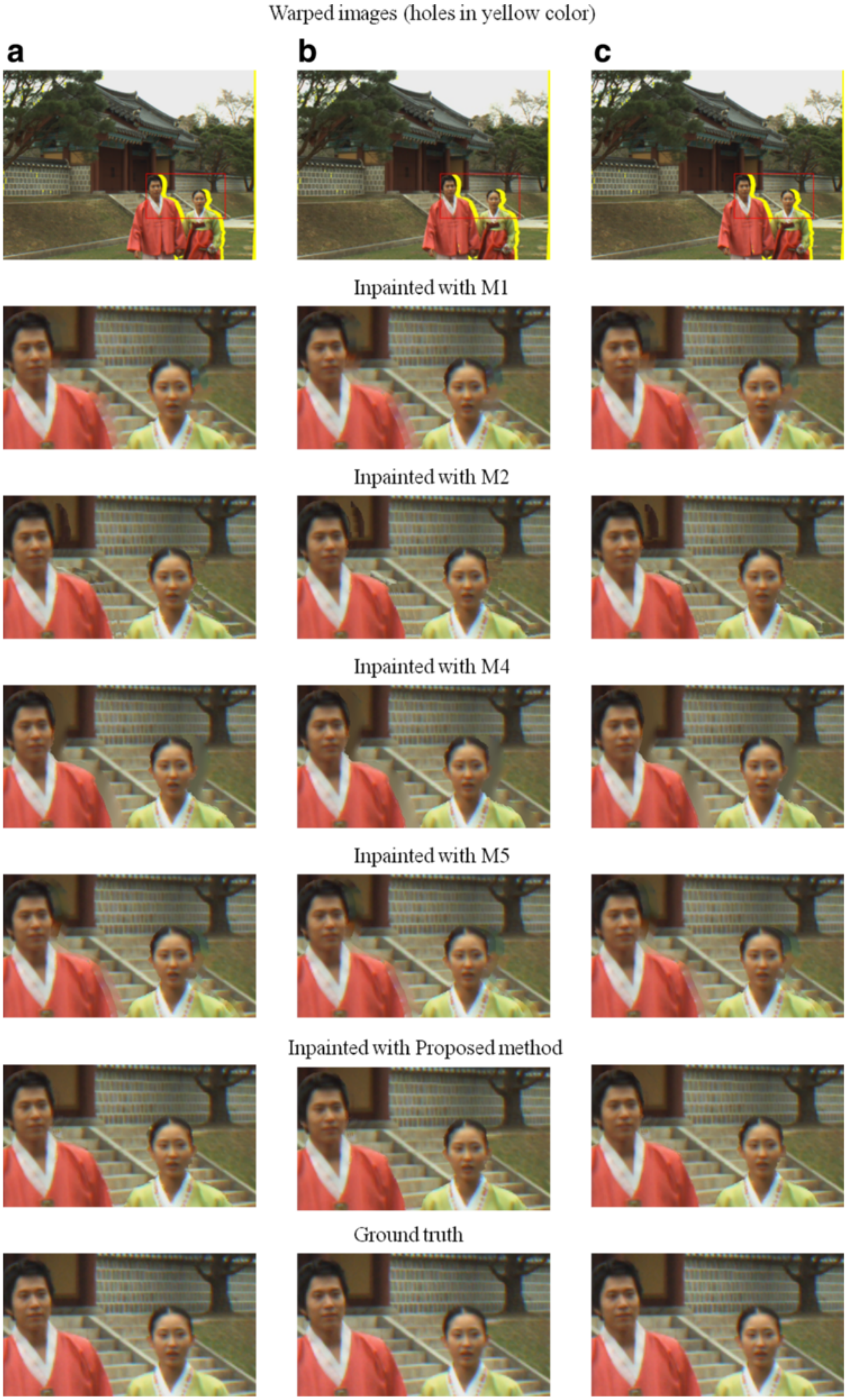

Fig. 11 Visual quality results of "Lovebird1" v6→v8 sequential frames. **a** Frame 111. **b** Frame 112. **c** Frame 113

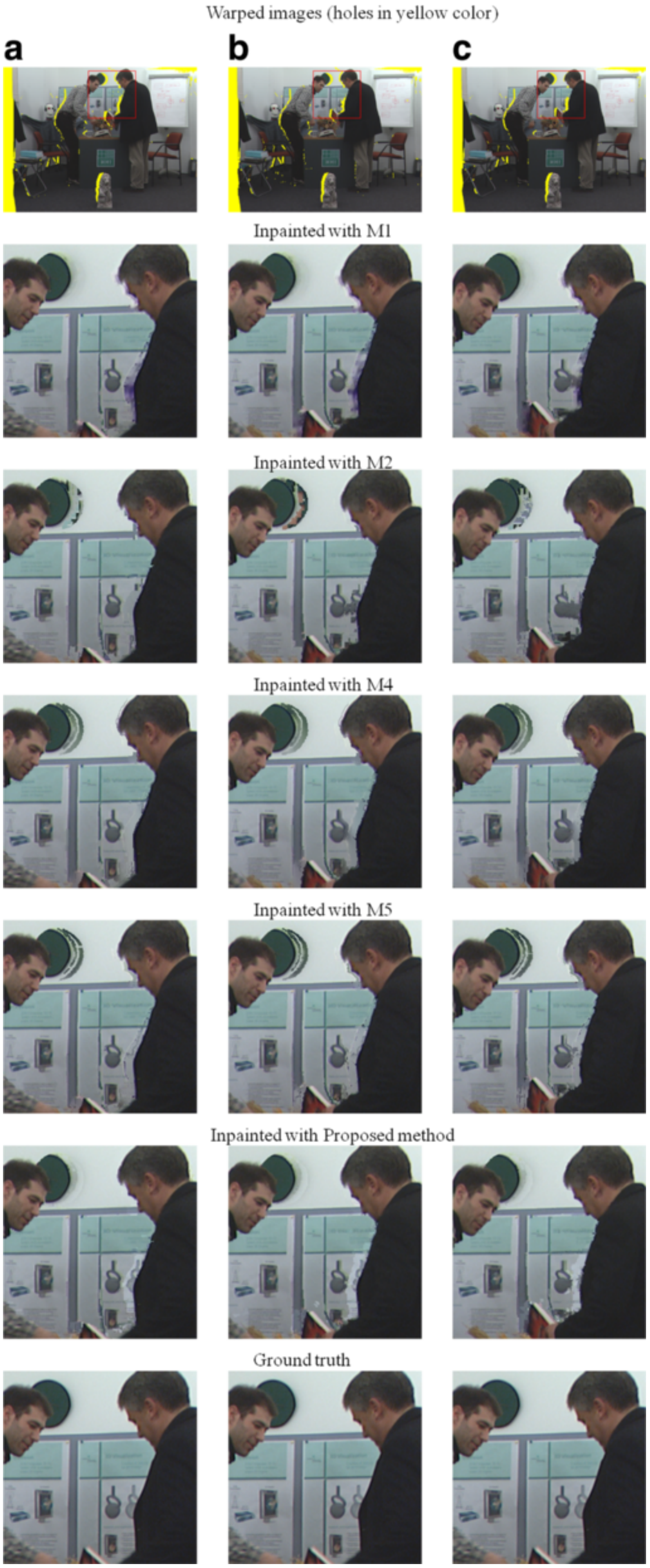

Fig. 12 Visual quality results of "Book Arrival" v8→v10 sequential frames. **a** Frame 57. **b** Frame 58. **c** Frame 59

relative background. We partially addressed this problem in our previous work by identifying the holes between relative foreground and background but only in the spatial domain [27].

4.2 Temporal consistency evaluation

Results from the objective temporal consistency measurements are presented in Table 4. In addition to the average F-score values, the inter-frame absolute difference at the holes and F-score of few frames are presented to show the similarity with the original signal (see Fig. 8). The average flickering score demonstrates that the proposed method performs better than the state-of-the art reference methods (consistent with the results from the spatial domain). Figure 8a shows that the result from the proposed method is similar to the ground truth data. In addition to objective measurements, the visual comparison clearly shows that the holes are temporally stable at the static holes (see row 7 in Fig. 9 at the holes near the man and row 7 in Fig. 10 at the holes near poles and the woman). Holes at the moving objects are also filled smoothly over frames (see row 6 in Fig. 11 at the holes near the woman, the edge of the stairs and in Fig. 12 at the holes near the man). Although flickering artifacts are significantly reduced at both static and moving objects' holes, apparent flickering is noticeable in the foreground of "Poznan Street" and less apparent in "Book Arrival" sequences due to erroneous depth. However, these are not well reflected by the F-score in the case of "Poznan Street" due to the flickering areas being significantly smaller than other disocclusions in the scène.

4.3 Complexity evaluation

Using the proposed method, it takes about 11 min to render a single frame of "Ballet" v4→v3, whereas using the reference method M1 takes 0.4 min, M2 takes 35 min, M3 takes 5 min, M4 takes 0.12 min and M5 takes 0.05 min, respectively. Note that M1, M3, M4 and M5 are implemented in C++, whereas M2 is implemented in MATLAB. Compared to the spatial hole-filling method M2, the computation time for the proposed method is reduced. This is because the holes at the static objects are inpainted once and the inpainted information is reused until they move, which means that there is no need to inpaint every frame for the holes at the dynamic objects. Because of the LDI generation and motion estimation and iterative inpainting, the proposed method is relatively slower than the reference methods.

5 Conclusions

We proposed a new view synthesis method with a spatial and temporal consistent layered depth image and depth-based inpainting, in order to improve the rendered view quality. The proposed method handles disocclusions in the virtual view in a spatial and temporal consistent manner. Furthermore, the proposed hole classification using the optical flow ensures the temporal consistency at static object holes by reusing the previously inpainted information. Spatio-temporal consistency is also ensured at moving objects' holes by using the true texture from temporal frames and inpainting. The computational time was reduced compared to reference methods by using the hole classification and reusing the inpainted textures to fill the holes at static objects. Moving weighted average blending between the textures within the hole regions reduced flickering and created a smooth transition. The objective and subjective test results consistently demonstrated that the proposed view synthesis method improved virtual view quality of the static background scenes. The results highlight the importance of the hole classification in improving the spatio-temporal consistency of extrapolated views.

The proposed method is aimed at static backgrounds and fills the holes with absolute background only, which means that the method is limited when it comes to handling the occlusion between relative foreground and background objects. Investigating possible improvements by identifying the relative foreground and background and building layered sprite is a matter for the future work. Furthermore, if the scene has illumination variations, reusing the inpainted information might cause artifacts. Investigations and finding a way to handle these problems is also a matter for future work. Identifying moving objects using other optimized and dense optical flow algorithms and finding the foreground objects locations in relation to the temporal domain is still challenging and is left for the future work. Optimizing the computation time is another issue to be dealt with in future investigations.

Additional file

Supplementary videos can be found at the paper web page www.miun.se/stc/Realistic3D/Muddala-2015.

Abbreviations

2D, two-dimensional; 3D, three-dimensional; 3DTV, three-dimensional television; FTV, free viewpoint television; DIBR, depth-image-based rendering; LDI, layered depth image; PSNR, peak signal-to-noise ratio; MSSIM, mean structural similarity index.

Competing interests

The authors declare that they have no competing interests.

Acknowledgements

This work has partly been supported by grant 20120328 of the KK Foundation, Sweden. We would like to thank Sunghwan Choi for providing their test results. We would like to thank Mitra Damghanian and Martin Köppel for their help. We would also like to acknowledge the reviewers for their valuable and helpful suggestions.

Received: 23 April 2015 Accepted: 14 February 2016
Published online: 29 February 2016

References

1. O Schreer, P Kauff, T Sikora, *3D Video Communication Algorithms, Concepts and Real-time Systems in Human Centered Communication*. (John Wiley & Sons Ltd., Chichester, 2005)
2. M Tanimoto, in *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW '06. Conference On*. FTV (free viewpoint television) for 3D scene reproduction and creation, (2006), pp. 172–172. doi:10.1109/CVPRW.2006.84
3. C Fehn, in *Proc. SPIE Stereoscopic Displays and Virtual Reality Systems XI*. Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV, (2004), pp. 93–104. doi:10.1117/12.524762
4. L Zhang, WJ Tam, in *IEEE Transactions on Broadcasting*. Stereoscopic image generation based on depth images for 3D TV, (2005), pp. 191–199. doi:10.1109/TBC.2005.846190
5. M Köppel, MB Makhlouf, M Muller, P Ndjiki-Nya, in *Visual Communications and Image Processing (VCIP), 2013*. Temporally consistent adaptive depth map preprocessing for view synthesis, (2013), pp. 1–6. doi:10.1109/VCIP.2013.6706436
6. K Muller, A Smolic, K Dix, P Merkle, P Kauff, T Wiegand, View synthesis for advanced 3D video systems. EURASIP J. Image Video Process. (2008). doi:10.1155/2008/438148
7. Y Mori, N Fukushima, T Yendo, T Fujii, M Tanimoto, View generation with 3D warping using depth information for FTV. Signal Process. Image Commun. **24**(1–2), 65–72 (2009). doi:10.1016/j.image.2008.10.013
8. D Tian, PL Lai, P Lopez, C Gomila, View synthesis techniques for 3D video. Proc. SPIE. **7443**, 74430–7443011 (2009). doi:10.1117/12.829372
9. SM Muddala, M Sjöström, R Olsson, S Tourancheau, Edge-aided virtual view rendering for multiview video plus depth. Proc. SPIE. **8650**, 86500–865007 (2013). doi:10.1117/12.2004116
10. N Plath, S Knorr, L Goldmann, T Sikora, Adaptive image warping for hole prevention in 3D view synthesis. Image Process. IEEE Trans. **22**(9), 3420–3432 (2013). doi:10.1109/TIP.2013.2268940
11. N Stefanoski, O Wang, M Lang, P Greisen, S Heinzle, A Smolic, Automatic view synthesis by image-domain-warping. Image Process. IEEE Trans. **22**(9), 3329–3341 (2013). doi:10.1109/TIP.2013.2264817
12. SM Muddala, M Sjöström, R Olsson, in *International Conference on 3D Imaging 2012 (IC3D)*. Edge-preserving depth-image-based rendering method, (2012). doi:10.1109/IC3D.2012.6615113
13. M Bertalmio, G Sapiro, V Caselles, C Ballester, in *Proceedings of ACM Conf. Comp. Graphics (SIGGRAPH)*. Image inpainting, (2000), pp. 417–424. doi:10.1145/344779.344972
14. A Telea, An image inpainting technique based on the fast marching method. J. Graphics GPU Game Tools. **9**(1), 23–34 (2004). doi:10.1145/344779.344972
15. Report on experimental framework for 3D video coding (2010). ISO/IEC JTC1/SC29/WG11 MPEG2010/N11631. Guangzhou, China
16. KJ Oh, S Yea, YS Ho, in *Proceedings of the 27th Conference on Picture Coding Symposium, Chicago, IL, USA*. Hole filling method using depth based in-painting for view synthesis in free viewpoint television (FTV) and 3D video, (2009), pp. 233–236
17. A Criminisi, P Pérez, K Toyama, Region filling and object removal by exemplar-based image inpainting. IEEE Trans. Image Process. **13**, 1200–1212 (2004). doi:10.1109/TIP.2004.833105
18. I Daribo, H Saito, A novel inpainting-based layered depth video for 3DTV. Broadcasting IEEE Trans. **57**(2), 533–541 (2011). doi:10.1109/TBC.2011.2125110
19. V Jantet, C Guillemot, L Morin, in *3D Research*. Joint projection filling method for occlusion handling in depth-image-based rendering, (2011). doi:10.1007/3DRes.04(2011)4
20. H Lim, YS Kim, S Lee, O C, JDK Kim, C Kim, in *Image Processing (ICIP), 2011 18th IEEE International Conference On*. Bi-layer inpainting for novel view synthesis, (2011), pp. 1089–1092. doi:10.1109/ICIP.2011.6115615
21. I Ahn, C Kim, A novel depth-based virtual view synthesis method for free viewpoint video. Broadcasting, IEEE Trans. **59**(4), 614–626 (2013). doi:10.1109/TBC.2013.2281658
22. SM Muddala, R Olsson, M Sjöström, Depth-included curvature inpainting for disocclusion filling in view synthesis. Int. J. Adv. Telecommun. **6**(3 & 4), 132–142 (2013)
23. SM Muddala, M Sjöström, R Olsson, in *3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), 2014*. Depth-based inpainting for disocclusion filling, (2014), pp. 1–4. doi:10.1109/TBC.2005.846190
24. JW Shade, SJ Gortler, L-W He, R Szelisk, in *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '98*. Layered depth images, (1998), pp. 231–242. doi:10.1145/280814.280882
25. B Barenbrug, R-PM Berretty, RK Gunnewiek, Robust image, depth, and occlusion generation from uncalibrated stereo. Proc. SPIE 6803, 68031–680318 (2008). doi:10.1117/12.765508
26. B Bartczak, P Vandewalle, O Grau, G Briand, J Fournier, P Kerbiriou, M Murdoch, M Müller, R Goris, R Koch, R van der Vleuten, Display-independent 3D-TV production and delivery using the layered depth video format. Broadcasting IEEE Trans. **57**(2), 477–490 (2011). doi:10.1109/TBC.2011.2120790
27. SM Muddala, Free view rendering for 3D video – Edge-aided rendering and depth-based image inpainting (2015). Doctoral Thesis No. 226, Mid Sweden University, Sundsvall, Sweden
28. K-Y Chen, P-K Tsung, P-C Lin, H-J Yang, L-G Chen, in *3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), 2010*. Hybrid motion/depth-oriented inpainting for virtual view synthesis in multiview applications, (2010), pp. 1–4. doi:10.1109/3DTV.2010.5506309
29. RK Gunnewiek, R-PM Berretty, B Barenbrug, JPM aes, Coherent spatial and temporal occlusion generation. Proc. SPIE. **7237**, 723713–72371310 (2009)
30. M Schmeing, X Jiang, in *3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), 2010*. Depth image based rendering: a faithful approach for the disocclusion problem, (2010), pp. 1–4. doi:10.1109/3DTV.2010.5506596
31. P Ndjiki-Nya, M Köppel, D Doshkov, H Lakshman, P Merkle, K Muller, T Wiegand, Depth image-based rendering with advanced texture synthesis for 3-D video. Multimedia, IEEE Trans. **13**(3), 453–465 (2011). doi:10.1109/TMM.2011.2128862
32. M Köppel, X Wang, D Doshkov, T Wiegand, P Ndjiki-Nya, in *19th IEEE International Conference on Image Processing, ICIP, 2012*. Depth image-based rendering with spatio-temporally consistent texture synthesis for 3-D video with global motion, (2012), pp. 2713–2716. doi:10.1109/ICIP.2012.6467459
33. W Sun, OC Au, L Xu, Y Li, W Hu, in *Image Processing (ICIP), 2012 19th IEEE International Conference On*. Novel temporal domain hole filling based on background modeling for view synthesis, (2012), pp. 2721–2724. doi:10.1109/ICIP.2012.6467461
34. C Yao, T Tillo, Y Zhao, J Xiao, H Bai, C Lin, Depth map driven hole filling algorithm exploiting temporal correlation information. Broadcasting, IEEE Trans. **60**(2), 394–404 (2014). doi:10.1109/TBC.2014.2321671
35. M Xi, L-H Wang, Q-Q Yang, D Li, M Zhang, Depth-image-based rendering with spatial and temporal texture synthesis for 3DTV. EURASIP J. Image Video Process. **2013**, 1–18 (2013). doi:10.1186/1687-5281-2013-9
36. H-A Hsu, C-K Chiang, S-H Lai, Spatio-temporally consistent view synthesis from video-plus-depth data with global optimization. Circuits Syst. Video Technol. IEEE Trans. **24**(1), 74–84 (2014). doi:10.1109/TCSVT.2013.2276699
37. S Choi, B Ham, K Sohn, Space-time hole filling with random walks in view extrapolation for 3D video. Image Process. IEEE Trans. **22**(6), 2429–2441 (2013). doi:10.1109/TIP.2013.2251646
38. N Otsu, A threshold selection method from gray level histograms. IEEE Trans. Systems, Man Cybernetics. **9**, 62–66 (1979). doi:10.1109/TSMC.1979.4310076
39. BD Lucas, T Kanade, in *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2. IJCAI'81, Vancouver, B. C., Canada*. An iterative image registration technique with an application to stereo vision, (1981), pp. 674–679
40. CL Zitnick, SB Kang, M Uyttendaele, S Winder, R Szeliski, High-quality video view interpolation using a layered representation. ACM Trans. Graph. **23**(3), 600–608 (2004)
41. I Feldmann, M Mueller, F Zilly, R Tanger, K Mueller, A Smolic, P Kauff, T Wiegand, HHI test material for 3D video (2008). ISO/IEC JTC1/SC29/WG11 MPEG 2008/M15413
42. GM Um, G Bang, N Hur, J Kim, YS Ho, 3D video test material of outdoor scene (2008). ISO/IEC JTC1/SC29/WG11/M15371
43. M Domanski, T Grajek, K Klimaszewski, M Kurc, O Stankiewicz, J Stankowski, K Wegner, Poznan multiview video test sequences and camera parameters (2009). ISO/IEC JTC1/SC29/WG11 MPEG 2009/M17050. Xian, China
44. Z Wang, AC Bovik, HR Sheikh, EP Simoncelli, Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004). doi:10.1109/TIP.2003.819861

45. M Schmeing, J Xiaoyi, in *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), 2011.* Time-consistency of disocclusion filling algorithms in depth image based rendering, (2011), pp. 1–4. doi:10.1109/3DTV.2011.5877201
46. G Bang, MS Ko, J Yoo, WS Cheong, G Um, N Hur, Boundary noise removal and common hole filling method for VSRS 3.5 (2011). ISO/IEC JTC1/SC29/WG11 Doc. m19356
47. C Lee, YS Ho, Results of view synthesis using modified hole filling methods (2010). ISO/IEC JTC1/SC29/WG11 Doc. m19281. Guangzhou, China