Hybrid dual stream blender for wide baseline view synthesis

Nour Hobloss, Lu Zhang, Stephane Lathuiliere, Marco Cagnazzo, Attilio Fiandrotti

Hybrid Dual Stream Blender For Wide Baseline View Synthesis

Nour Hobloss[a,b], Lu Zhang[d], Stéphane Lathuilière[b], Marco Cagnazzo[b,a], Attilio Fiandrotti[b,c]

[a] *IRT b<>com, Rennes, France*
[b] *LTCI, Télécom-Paris, Institut Polytechnique de Paris, France*
[c] *Università di Torino, Dipartimento di Informatica, Italy*
[d] *Univ Rennes, INSA Rennes, CNRS, IETR - UMR 6164, F-35000*

Abstract

Free navigation of a scene requires warping some reference views to some desired target viewpoint and blending them to synthesize a virtual view. Convolutional Neural Networks (ConvNets) based methods can learn both the warping and blending tasks jointly. Such methods are often designed for moderate inter-camera baseline distance and larger kernels are required for warping if the baseline distance increases. Algorithmic methods can in principle deal with large baselines, however the synthesized view suffers from artifacts near disoccluded pixels. We present a hybrid approach where first, reference views are algorithmic-ally warped to the target position and then are blended via a ConvNet. Preliminary view warping allows reducing the size of the convolutional kernels and thus the learnable parameters count. We propose a residual encoder-decoder for image blending with a Siamese encoder to further keep the parameters count low. We also contribute a hole inpainting algorithm to fill the disocclusions in the warped views. Our view synthesis experiments on real multiview sequences show better objective image quality than state-of-the-art methods due to fewer artifacts in the synthesised images.

Keywords: View synthesis, View blending, Convolutional Neural Network, Hole inpainting.

1. Introduction

Free navigation systems allow users to freely browse a scene by arbitrarily changing viewpoint. The key benefit of such systems is providing a new immersive user experience and interactivity that goes beyond higher image quality and higher realism. A smooth transition between viewpoints would require to capture the scene from a large number of viewpoints. Because it is infeasible to acquire all possible viewpoints, only a few selected views are actually captured (with corresponding depth maps) [1, 2]. Missing viewpoints are usually synthesized exploiting the captured neighboring view. In the rest of this work, we will refer to the synthesized view as *target view*, and to the neighbors used for the synthesis as *reference views*.

Depth-Image-Based Rendering (DIBR) methods represent state of the art in view synthesis: they synthesize the target view exploiting the 3D geometry of the scene and the associated depth maps. First, reference views are warped to the target view position using their corresponding depth maps. Then, the warped images are blended together to finally produce the sought target view [3]. Usually, the quality of the synthesis process improves with the number of available references, as well as its complexity [4] [5]. Much of existing research focuses on the case where

only a pair of references are available for synthesis: similarly, in this work we will assume that only two references are exploitable for synthesis.

DIBR-based methods perform less favorably in the case of occlusions, *i.e.* when a pixel of the target view is occluded in both references [6]. Occluded pixels are usually inpainted by interpolating the available neighbors, somewhat mitigating the issue [7, 8, 9, 10, 11, 12]. For example, in [13, 11], multiple warped reference views are combined to fill holes in the target view. However, many of such methods still yield visible artifacts in the synthesized view, prompting for improved view synthesis schemes.

Learning-based approaches based on Convolutional Neural Networks (ConvNets) have been proposed with remarkable results. ConvNet based approaches have been proposed to learn the view synthesis process end-to-end taking care of all the steps of a typical DIBR method (view warping, blending and inpainting) [14, 15, 16, 17]. Such approaches limit not only the impact of occlusions, but also texture misalignment, color dis-harmonization, and different exposures among cameras [18]. In the case of large baselines, as it is often the case, large convolutional kernels are in fact required to handle the larger baseline [19]. Large kernels increase the network complexity at deployment time and makes the network prone to overfitting during training. In our previous work [20], we explored the idea of preliminary warping the reference views to the target position, then blending the warped views with the aid of a ConvNet.

We showed that a plausible novel target view can be obtained with a simple architecture, which outperforms traditional algorithmic methods (blending and inpainting) in most cases. However, the network showed reduced performance on more complex cases, mainly due to the limited generalization capacity of such a simple architecture.

The present work builds upon our previous research [20] retaining the ideas of a hybrid algorithmic-learning scheme where reference views are preliminary warped to the target position and using an inpainting method built around a median filter to handle occlusions. However, this work improves our previous approach under several aspects:

- We blend the warped views using a residual encoder-decoder architecture inspired by recent advances in image-to-image translation [21, 22]. Namely, we reformulate our problem as an image-to-image translation task, aiming to translate warped input real views into the target view.
- We provide the ConvNet in charge of blending the warped views both the warped and the inpainted views to allow the network to pick whatever is the best source for resolving each occluded pixel.
- Unlike image-to-image translation problems, we must deal with multiple reference views which entails one encoder for each view. To keep low the parameter count, we introduce a *flip-convolve-flip* scheme that allows sharing parameters between encoders operating on symmetric inputs.
- We improved the inpainting method built around a median filter to handle occlusions to fill all the remaining holes in the warped views.
- We experiment the use of binary masks as an alternative to the inpainted views inputs to reduce the network complexity.

We experimentally evaluate our proposed approach over multiple wide baseline multiview video sequences comparing with purely algorithmic and purely learnable state of the art approaches. Our experiments show that our architecture outperforms competitors in terms of image

quality for wide baseline sequences while striking a favorable balance between complexity and performance.

The rest of this document is organized as follows. Section 2 reviews the relevant literature and introduces the related background. Section 3 describes the proposed view synthesis method. Section 4 experimentally evaluates the performance of our method, including an ablation study. Finally, Section 5 discusses the learned lessons and discusses future research.

2. Related Works

View synthesis methods can be divided into algorithmic-based methods on one hand and learning-based methods on the other. In this section we describe the state of the art for both classes and we highlight the relative limitations. Finally, we recapitulate on the limitations of existing view synthesis methods that motivate our work.

2.1. Algorithmic-based methods

Depth-Image-Based Rendering (DIBR) methods operate by first warping at least two reference views to a target virtual view position exploiting depth maps and cameras' parameters. Then, warped reference views are blended to synthesize the target virtual view. We review here the most relevant representatives of such method.

2.1.1. MPEG VSRS and VVS

VSRS [23] is an early reference software for view synthesis released by the MPEG as a result of a challenge for 6 degrees of freedom (6-DoF) of the MPEG-I group. The VSRS scheme consists in warping the depth maps of the reference views to the target view position. Then the warped depth maps are used by the reference textures and to project the target view by a backward warping method, obtaining warped images. Finally, the warped views are blended by exploiting the warped depth maps. Despite its robustness, VSRS had a few drawbacks that caused its performance to flutter from content to content, especially when the synthesis is performed at the receiver, after decoding the compressed reference views.

VVS [3] is a later reference software for view synthesis released by the MPEG superseding VSRS. VVS is conceptually similar to VSRS, however it accounts for the compression artifact on view textures and depth maps at the receiver side. Namely, VVS improves the precision of the backward texture warping and better preserves edges via a conditional depth blending process, up-sampling the reference textures, and using reliability maps that indicate which pixels are safe to be warped to the target view position. Contrary to VSRS, VVS gives priority to foreground pixels during warping, and warps "triangles" instead of "points" to generate fewer occlusion artifacts in the warped view followed by a series of hole inpainting steps. VVS outperforms many view synthesis methods in reason of the high quality of the warped reference views. However, we notice that the method has difficulties to recover occluded regions in complex scenes (larger baseline, or highly detailed content) producing persistent artifacts in the synthesized view. Our investigations showed that errors accumulate all over the sequence of steps corresponding to the warped textures blending and the inpainting process.

2.1.2. Other algorithmic methods

Penner *et al.* [24] propose a soft 3D reconstruction of the scene that preserves depth uncertainty through each stage of 3D reconstruction and rendering, improving quality, continuity and robustness. Canclini *et al.* [25] estimate the infinite homography and assume it is sufficient to generate a novel view from uncalibrated reference images. To improve quality of synthesized views by DIBR Xu *et al.* [26] propose depth map misalignment correction and dilation. In coding scheme and to enhance compression performance, Xiao *et al* [27] suggest a scalable bit allocation between texture and depth views to enable synthesis virtual views in coding format. However, Yao *et al* [28] propose depth map down-sampling in order to minimize the view synthesis distortion. Lai *et al* [29] use a depth-reliability-map-based occlusion aware approach to create a segmentation mask, that indicates where the information in the novel synthesized view should be blended. Lin *et al* [30] propose a fast multi-view image rendering method that uses a pixel mapping information to derive a rendering image. This method reduces rendering time and memory size comparing to a conventional DIBR method, however it has the same effect in terms of synthesized image quality as DIBR. Chaurasia *et al* [31] introduce a new DIBR algorithm robust to missing or unreliable geometry by proposing a depth synthesis approach. They oversegment the reference images, creating superpixels of homogeneous color content which tends to preserve depth discontinuities.

However, the critical issues in all view synthesis algorithms remains in the occluded regions in the output image when rendering a scene from a novel viewpoint. These regions typically have to be filled using inpainting algorithms, which might yield unconvincing results.

2.1.3. Hole inpainting

A common issue with all the above methods is recovering occlusions in the reference views, i.e. areas that are occluded in all references, that show up in the synthesized view as visual artifacts. Resolving occlusions is a challenge per se and several approaches have been proposed, mainly based on hole inpaiting [7, 8, 9, 10, 11, 12, 32].

To minimize disocclusion holes in the synthesized novel view, Thatte *et al.* [33] proposed a statistical model that predicts the likelihood of missing data in synthesized images as a function of the viewpoint translation.

Li *et al.* [34] employed multiple views to synthesize output virtual views, by proposing a scheme of selective warping of complementary views developed by locating a small number of useful pixels for hole reduction.

In [11, 13], multiple reference views are warped and combined to generate a blended image. Other methods use the neighbor pixel color or the depth information to extrapolate or interpolate the occluded pixels [8, 35], or by pre-processing the warped depth maps [36].

In [6], Luo *et al.* extract the foreground objects in reference images and synthesize the background to be used to fill holes in the synthesized view, as they consider that occluded pixels have the same patterns as the background.

Yao *et al* [37] exploit the temporal correlation of texture and depth information to generate a background reference image, used to fill the holes associated to the dynamic parts of the scene.

2.2. Learning-based methods

Over the years, a number of learning-based approaches to view synthesis relying on ConvNets have been proposed.

Early approaches to view synthesis mainly addressed the problem of temporal frame rate upsampling, where missing pixels are interpolated temporal or spatial neighbors. Niklaus *et*

al. [38], for example, propose a convolutional architecture capable of joint motion estimation and pixel synthesis for temporal up-sampling. Liu *et al.* [15] propose a convolutional architecture that output is a 3D voxel flow field, used to sample the original input video with a volume sampling function to synthesize the final frame. The work in [19] deals with large object motion or occlusion by explicitly detecting occlusion leveraging the depth information to warp the temporally adjacent frames. Such approaches assume that the equivalent camera displacement between consecutive frames of the same view is small. So, they are intrinsically unsuited to the free viewpoint scenario we address due to the large baseline distance between cameras. Regmi *et al.* [39] use a homography as a pre-processing step to map the images between reference views and guide a generative adversarial networks to inpaint the missing regions in the novel view. However, their method mainly addresses the problem of generating images across street and aerial views.

Later on, view synthesis methods designed especially for free-navigation have been proposed. Flynn *et al.* [10] proposes a deep convolutional architecture that synthesizes the target view directly from the reference view pixels, avoiding the multiple pre-processing stages of traditional approaches. Such an approach requires a large number of images to be trained end-to-end in a supervised way. Hemad *et al.* [12] designed a ConvNet that takes four warped images with a global mesh of the scene as inputs, and a voxelized representation of the scene is used to accelerate the reference view selection. Each voxel indexes relevant triangles from the per-view meshes, which will be rendered to the target viewpoint. Flynn *et al.* [9] estimate a multi-plane image from sparse views that uses learned gradient descent. In this method, they use a set of four input views captured in a 2D array, and its convenience is in complexity related to the number of depth planes that increase with the maximum disparity in order to produce the multi-plane images. Inchang *et al.* [40], they proposed a neural network in charge of the refinement of a pre-synthesized view. However this method struggles to fix artifacts that look natural. Due to network complexity issues, such methods would have challenges to cope with persistent artifacts.

More recently, end-to-end view synthesis methods relying on a single image have appeared. In Xiaogang Xu *et al.* [41], the user specifies arbitrarily the desired new camera-pose, and their encoder-decoder characterizes the input image properties in terms of 3D-structure, color and texture, to hallucinate the image of a novel view. Olivia Wiles *et al.* [18] propose a novel end-to-end model using a single image at test time, trained on real images without using any ground-truth 3D information, they instead develop a differentiable point cloud renderer used to transform 3D point cloud of features to the target view. Other related works based on CNN for view synthesis are proposed in [15] [16]. Due to complexity problems faced when warping a reference view using a ConvNet and due to the lack of information on the 3D scene, such approaches would have issues to deal with view synthesis with large disparity. Unlike the previous learning-based view synthesis methods, Mildenhall *et al.* [42] opted for a non-convolutional deep fully-connected neural network referred to as a multiplayer perceptron, it takes as input the spatial location and the viewing direction coordinates to synthesize a 2D novel view. This technique achieves excellent results, but it has been applied to cases of small baseline.

2.3. Objectives and approach

Our state-of-the-art analysis suggest that none of the existing methods is free from drawbacks in the form of artifacts in the synthesized image. With algorithmic-based approaches such as VVS, view blending and hole inpainting yield artifacts in the synthesized view. With learning-based methods, problems in generalization and consistency can be traced back to the end-to-end view warping and blending process. While blending warped reference views allows handling

large baselines with small kernels, it does not solve the problem with the artifacts in the synthesized view. In our previous work [20], we took inspiration from convolutional architectures for image super resolution [43]. Conversely, in this work we take a different approach inspired by recent advances in image-to-image (I2I) translation [21, 22]. I2I translation consists in mapping one image to another and tackles problems such as image colorization, super-resolution and, to some extent, also view synthesis. Most I2I approaches rely on encoder-decoder architectures where the input image is first projected on a latent feature space by a convolutional encoder. Usually, the encoder relies on pooling layers or multiple-strided convolutional layers to reduce the spatial resolution of the feature maps. Next, such features may pass through a bottleneck layer that projects them over a feature space of (usually) lower dimensionality. Then, these features are projected back to the original pixel domain by a transposed convolutions decoder. The decoder usually employs transposed convolutions (or fractionnally-strided convolutions) to recover the original resolution of the translated image. The transposed convolution is used to conduct optimal up-sampling, it also has learnable parameters. A key challenge to tackle in our view synthesis scenario is i) how to encode into the latent features space the left and the right warped views and ii) how to blend the features in the latent space towards the synthesized view. A further key challenge for our application is how to prevent occlusions in the warped images from generating artifacts in the synthesised view, a problem that in I2I architectures is usually not present. In the next section, we propose a convolutional architecture for view synthesis that takes inspiration from I2I methods yet tackles challenges unique to view synthesis problems.

3. Hybrid Dual Stream Blender (HDSB)

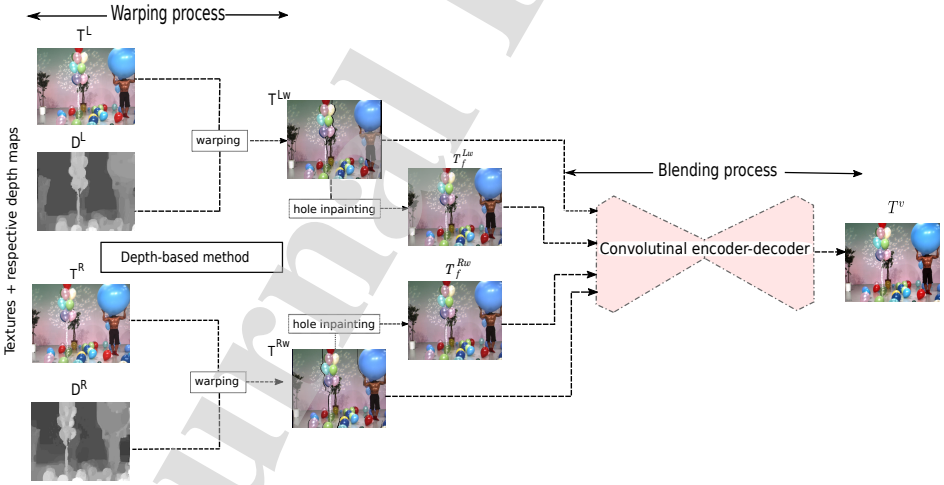


Figure 1: The proposed hybrid pipeline for wide-baseline view synthesis: reference views are first warped to the target position, disocclusions are inpainted and eventually blended by a ConvNet.

Fig. 1 illustrates the proposed wide-baseline view synthesis pipeline, where given two left and right input reference views I^L and I^R are composed by textures T^L and T^R, and depths D^L and D^R respectively. We aim at synthesizing texture T^v of the target view that lies in the middle between

the reference views. Towards this end, we propose a pipeline for view synthesis composed of a warping step followed by a hole filling step and a final blending step. First, the reference views are warped to the target position, producing warped references aligned with the target view to synthesize. Warping allows us to blend the warped images using a ConvNet with small kernels later on. Second, the hole inpainting consists in generating a filled texture (T_f^{Lw} and T_f^{Rw}) for each warped reference texture (T^{Lw} and T^{Rw}). Third and last, a ConvNet blends the left and right warped references (with filled textures T_f^{Lw} and T_f^{Rw}) to the target view T^v. The details of each step are detailed in the following.

3.1. Warping the reference views to the target position

The warping process consists in warping textures (T^{Lw}, T^{Rw}) to the target view position (T^v) with the aid of the depth maps. Namely, this step takes as input the two left and right reference views T^L and T^R and produces as outputs two warped reference views T^{Lw} and T^{Rw} to the same intermediate novel target view position T^v. The textures are back-projected to the target position as follows. Let pixels t_r and t_v be the projections of a same real world point denoted by X, and with coordinates $(u_r, v_r, 1)$, $(u_v, v_v, 1)$ respectively. Lets us consider K_r, K_v and R_r, R_v the respectively 3x3 intrinsic camera parameters and the 3x3 rotation matrix for each camera. Then, t_v can then be expressed as

$$t_v = K_v R_v (K_r R_r)^{-1} (z t_r + K_r R_r C_r) - K_v R_v C_v, \tag{1}$$

where z is the depth value. In particular, the two pairs of texture and depth $I^L = \{T^L, D^L\}$ and $I^R = \{T^R, D^R\}$ are up-sampled to half-pixel or quarter-pixel accuracy, in which the warping and interpolation steps are carried out.

In practical implementations of the above method, one has to decide the value to assign to pixels that are occluded in the reference views but visible in the target (disoccluded pixels). In practice, often those pixels are arbitrarily assigned a zero value, which entails a few drawbacks. In fact, a zero-valued pixel is ambiguous as it could represent either a non-occluded dark pixel or a disocclusion. While the ConvNet in charge of blending the warped views may learn this by itself, this would make more challenging the learning problem.

One possible solution is to provide an occlusion map for each warped texture as input to the network allowing a hypothesis on each pixel claiming whether yes or not it should be given a value, cf. Sec 4.3.2. An even better solution (at least, according to our experimental results) is to provide an additional version of the warped image where the disoccluded pixels are filled as much as possible with relevant information. We take this latter approach and we detail it the following section.

3.2. Hole inpainting

Due to the warping process of the reference view towards the target view, some visible area in the target view are invisible in the reference view and thus, they are represented as black holes in the warped views where each disoccluded pixel at a given position (r, c), $T^{Lw}(r, c)$ is set to zero. An example of the filled textures (T_f^{Lw} or T_f^{Rw}) is represented in Fig. 3.

The hole inpainting process fills in holes in textures (T^{Lw} and T^{Rw}) generated during warping as a result of pixel disocclusions. Namely, hole inpainting takes as input the left and right warped textures (T^{Lw} and T^{Rw}) and produces as output the corresponding warped, filled, textures (T_f^{Lw} and T_f^{Rw}) respectively. First, every disoccluded pixel at the position (r, c) in T^{Lw} (i.e, every

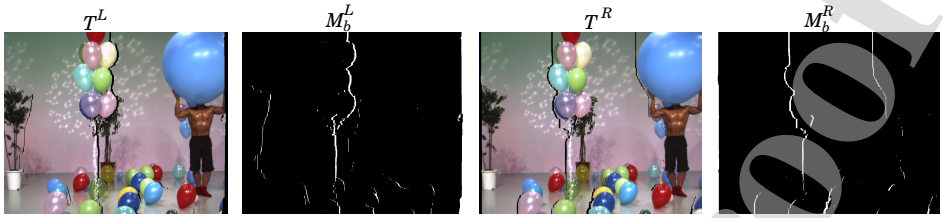


Figure 2: The two left and right warped input textures and their corresponding binary masks *Baloons* sequence.

$T^{Lw}(r,c) == 0$) is filled with the value of the co-located pixel at the position (r,c) in T^{Rw} if the latter is not occluded (*i.e* $T^{Rw}(r,c) \neq 0$). Then, the same procedure is followed to fill T^{Rw} using non-occluded pixels from T^{Lw}. However some pixels may still be occluded at this point when the new area in the target view is invisible in both left and right reference views.

To fill the remaining holes in T_f^{Lw} and T_f^{Rw}, we use for each remaining disoccluded pixel in T_f^{Lw} and T_f^{Rw} its non-occluded neighboring pixels. Therefore, we assume that the disoccluded pixels may appear only to the left or the right of foreground objects of the scene regarding the warping direction. Accordingly, we assign to each disoccluded pixel the median value of its five left or right neighboring valid pixels. Considering the remaining disoccluded pixels in T_f^{Lw} to be filled, each disoccluded $T^{Lw}(r,c)$ value is filled with the median value of the following five neighboring valid non-occluded pixels: $T_f^{Lw}(r-1,c)$, $T_f^{Lw}(r-1,c+1)$, $T_f^{Lw}(r,c+1)$, $T_f^{Lw}(r+1,c+1)$ and $T_f^{Lw}(r+1,c)$. A pseudo-code of this algorithm applied on the left warped image T^{Lw} is in Alg. 1. The resulting filled textures (T_f^{Lw}, T_f^{Rw}) go along within the warped textures (T^{Lw}, T^{Rw}) as inputs to the ConvNet responsible for the blending step described in the next section. In a nutshell, to fill holes in the left warped view, we leverage the knowledge that the holes will appear on the right side of the foreground objects, and we want to fill the missing information using the background. Thus, we sweep the image row after row, from left to right, filling holes pixel by pixel using the median over five right not occluded neighbours. We use the same technique to fill holes in the right warped image but instead we sweep the image from right to left, since the holes appears on the left of the foreground objects, using the median over the left five not occluded neighbors. Combining the two steps together, our technique allows reasonable results with larges holes. Finally, an approximation of the proposed method consists in using binary masks, in place of the filled textures, increasing speed and trading off visual quality.

Eventually, since generating filled warped textures is time and memory consuming, we propose here a lower complexity alternative based on occlusion maps. First, we generate a binary mask for each reference warped texture by labeling occluded pixels as 1, 0 otherwise. Fig 2 illustrates how occluded pixels in the reference image correspond to the visual black holes valued as 0. Therefore, as a low-complexity alternative, we propose to use the two reference warped textures and their corresponding binary masks (M_b^L, M_b^R) as inputs to our architecture, instead of the filled textures. In Sec. 4.3.2 we will use such method as a benchmark method to evaluate our proposed hole inpainting algorithm. C1

3.3. ConvNet-based blending

As a third and final step, the warped textures (T^{Lw}, T^{Rw}) and the relative filled counterparts (T_f^{Lw}, T_f^{Rw}) are blended together into a novel viewpoint using a ConvNet. We describe here the architecture of the CNN and the relative training procedure.

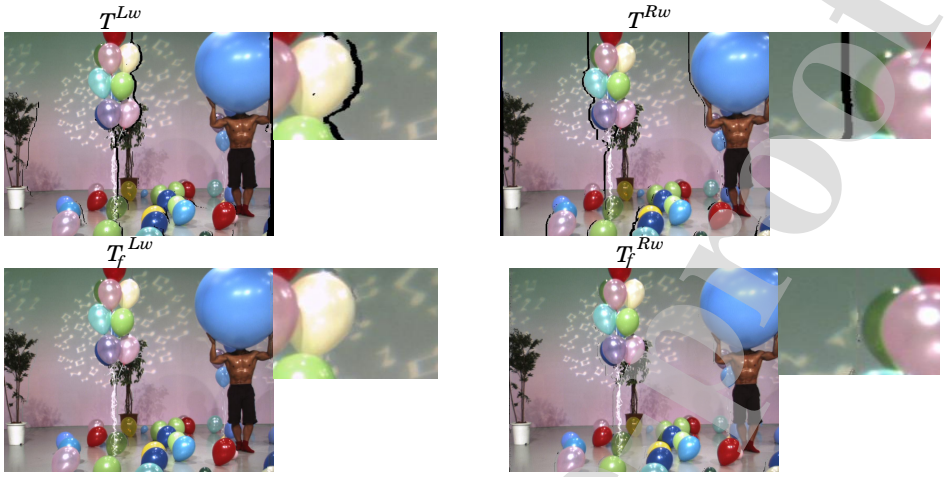


Figure 3: The two left and right warped input textures T^{Lw} and T^{Rw} and their corresponding filled textures T_f^{Lw} and T_f^{Rw} with details *Baloons* sequence.

3.3.1. ConvNet architecture

Recently, image-to-image translation architectures [21, 44] have shown that it is possible to map an image from a first visual domain to another image from a different domain. Indeed, we consider that the image-to-image problem is refined to view blending, so it may be reasonable to take inspiration from these architectures. However, there are some remarkable differences between image-to-image translation and the view blending problem considered here. First, we need to deal with a total of four inputs (the two warped reference textures and the relative filled textures) rather than with a single input image. Second, each pair of inputs (T^{Lw}, T_f^{Lw}) and (T^{Rw}, T_f^{Rw}) (cf. Fig 1), is characterized by a specific type of artifacts to be recovered. As one reference view is warped from the left side and the other from the right side, they do not share the same disocclusion problems. Therefore, our network should learn how to exploit such inputs together to obtain one synthesized view using and adapted architecture originally conceived for image-to-image translation task. Thus, we propose a specific encoder-decoder architecture that combines its four inputs and generates one output.

Our architecture is illustrated in figure 4 and is composed of three parts, the two *encoders*, the *blender*, and the *decoder*.

Encoders. Our ConvNet includes a pair of identical *encoders*, one for the left and one for the right view. We assume that all textures are 3-channels color images, e.g, in RGB or YUV format. The left encoder (the top encoder in Fig. 4) takes in input the left view composed of textures (T^{Lw}, T_f^{Lw}); the right encoder(the bottom encoder in Fig. 4) takes in input the right view made of textures (T_f^{Rw}, T^{Rw}). The role of each encoder is to project the views to a spatially-subsampled latent feature space. To this end, each encoder includes three convolutional layers with 64, 128 and 256 filters respectively of size 7x7 for the first layer and 3x3 for the last two, all followed by ReLU activations. Filters in the convolutional layers have 2-units stride, so that the feature maps in output of each layer are half the size of the feature maps in input to the layer. Eventually, the

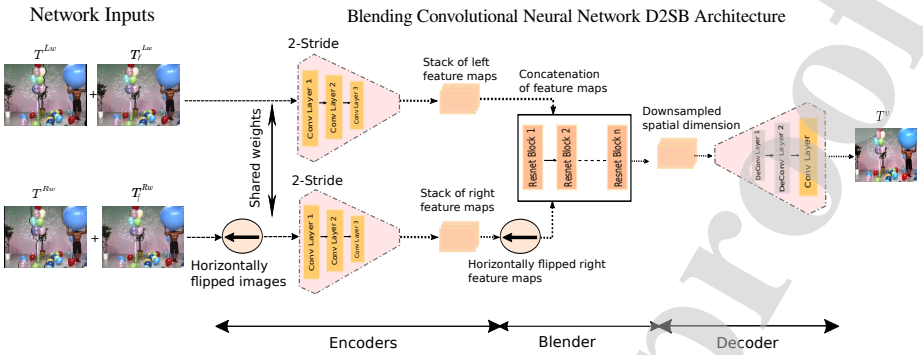


Figure 4: Deep Dual Stream Blender Architecture sharing parameters in the encoder stage.

feature maps output by the encoder are $\frac{1}{8}-th$ the size of the input textures. We found by extensive experimentation that such encoder topology achieves the best tradeoff between semantic depth and spatial resolution of the output feature maps.

Next, we propose to reduce the number of learnable parameters by sharing weights among encoders [45]. However, just sharing the same weights among the two encoders would be sub-optimal (as we experimentally verify in Sec. 4.3.4) since the left and right views do not share identical disocclusion artifacts. Indeed, occluded pixels in the left warped views will occur on the right side of objects, whereas in the right warped views occlusions will occur on the left side. That is, occlusion artifacts will show on the two opposite side of the objects in left and right views. Therefore, we propose a *flip-convolve-flip* approach that allows sharing parameters among encoders. First, we horizontally flip (mirror) the right view so that occluded pixels show on the right side of objects, as in the left view and as illustrated in Fig. 5. Then, the feature maps generated from the right encoder are horizontally flipped a second time. This produces feature maps that are semantically similar to those generated by the left encoder and can be easily merged by the bottleneck block later on while sharing parameters among encoders. While an extension of this scheme to the case of vertically arranged cameras by applying a vertical mirroring rather than horizontal may be theoretically envisaged, its discussion is out of the scope of this work which deals with arrays of 1-D horizontally arranged cameras. In Fig. 4, the mirroring operations are denoted by circled arrows.

Figure 5: Occlusions in the left view appear to the right of objects, occlusions in the right view to the left. When the right view is flipped, occlusions appear on the right of objects, as for the left view (*Newspaper* sequence).

Blender. The blender includes 6 residual blocks with 512 filters each and blends the feature maps extracted from the left and right views into a set of feature maps that holds a suitable representation of the desired target view. Our experiments showed that residual blocks are better suited than convolutional layers for the task of blending the downsampled feature maps into the target view. The blender block is our original answer to the problem of blending features in a latent, spatially subsampled, space originated from two different views of the same scene. By comparison, other image-to-image translation architectures deal with monoscopic images only, so they do not need to address this additional problem. In [20], features extracted from the input views did not undergo any spatial downsampling and they were only concatenated to synthesize the target view. We will experimentally show by ablation study the advantages of blending the features in a spatially subsampled feature space.

Decoder. The decoder finally synthesizes the virtual view at the target position exploiting the downsampled feature maps produced by the blender. The decoder includes 3 transposed convolutional layers with 256, 128 and 3 filters per layer of size 3x3 for the first two layers and 7x7 for the last layer. The decoder upsamples the low resolution feature maps produced by the blender component to a higher resolution. The first two layers are followed by ReLU activation function, while the output layer is followed by a hyperbolic tangent. We also stack a batch normalization layer after each convolutional layer as in ResNet blocks, as our experiments showed it speeds up the training process. Overall, our network produces in output a three-channels view where each view is expected to have approximately zero-mean and the pixel intensity is bounded in the $[-1, +1]$ interval by the output layer nonlinearity.

3.3.2. Training procedure

The network is trained in a fully supervised way on quintuplets of patches $(\widetilde{t}^{Lw}, \widetilde{t}^{Lw}_f, t^c, \widetilde{t}^{Rw}_f, \widetilde{t}^{Rw})$ extracted respectively from textures $(T^{Lw}, T^{Lw}_f, T^C, T^{Rw}, T^{Rw}_f)$, where and T^C is the ground-truth image to synthesize.

The training process for the proposed scheme may be hindered by the limited availability of suitable data. In facts, multiview plus depth video sequences are not easily produced or available, and most of our sequences are taken from the MPEG test material. However, we avoided the use of computer-generated (CG) video sequences. Typically, the characteristics of CG data are relevantly different from natural content. In CG data we have perfect depth-maps but also, depending on the rendering techniques, one can typically achieve a somewhat limited complexity of textures, noise levels are much lower than natural videos, and some phenomena are more difficult and more computationally intensive to be rendered (*e.g.*, non-Lambertian surfaces, subsurface scattering, *etc*.). In short, relying only on computer-generated data would not improve the training process so much if the methods are then to be used on natural data.

We randomly apply a vertical flip on each quintuplet as a form of data augmentation, to increase the diversity in our training samples. Our experiments showed that, due to the limited availability of suitable video contents, such augmentation method is fundamental to avoid overfitting on the training data. In order to keep the geometrical relationship between left, center and right patches, only a vertical flip is applied in this case. Before being provided in input to the network, patches are normalized so that the average per-channel pixel intensity has zero mean and unitary deviation.

Concerning the loss function to minimize at training time, we alternatively experiment with two options. The first function measures the distortion in the pixel space over a ground truth, while the latter aims instead at assessing image reconstruction quality as *perceived* by the user.

Pixel-based reconstruction loss.. For each pair of reference patches $(\widetilde{t}^{lw}, \widetilde{t}^{rw})$ provided in input, the network is trained to minimize the quadratic error between the network output t^v and the ground truth t^c. That is, at training time we minimize the loss function

$$L(w, t^v, t^c) = \frac{1}{n}\sum_{i=1}^{n}(t_i^v - t_i^c)^2, \tag{2}$$

where t_i^v and t_i^c are the i-th pixel of t^v and t^c, respectively. We train our network by back-propagating the gradient of the above error function and the network parameters are updated using the Adam algorithm [46].

Perceptual reconstruction loss.. Alternatively, we experiment with a perceptual loss function [47] to observe the *perceived* quality of the synthesized view. Perceptual loss relies on a feature extractor usually trained for image classification to compare two images based on their high-level features representations. Style transfer [21] experiments show that the perceptual loss training may achieves visually more pleasant images than per-pixel loss functions. Johnson *et al.* [48] first proposed the use of perceptual loss of image transformation tasks using a VGG16 trained on ImageNet [47]. The feature reconstruction loss is defined as

$$l_{feat}^{\phi}(\widetilde{y}, y) = \frac{1}{CHW}\|\phi(\widetilde{y}) - \phi(y)\|^2, \tag{3}$$

where l_{feat}^{ϕ} is the feature reconstruction loss from one layer of loss network ϕ of the content image $\widetilde{y}$ and content representation of the output image y. C is the number of filter in the input image, H and W are the height and the width of the input image. $\phi(\widetilde{y})$ and $\phi(y)$ are the feature representation of the content of the target image and the feature representation of the output of the target image respectively. Our image transformation network is thus trained using stochastic gradient descent to get weights that minimize the total loss, which is a weighted product of the feature reconstruction loss. The training procedure ends after the perceptual loss function measured over a validation set distinct from the training set stop decreasing.

4. Experiments and Results

In this section, we quantitatively and qualitatively evaluate our method HDSB in a comparative way and we perform ablation studies to validate each of our design choices.

4.1. Experimental setup

We experiment with well-known multi-view sequences commonly used in MPEG experiments as defined in the MPEG CTCs (Common Test Conditions) and detailed in Tab. 1 (views and depth maps are in uncompressed YUV format). Such sequences account for a wide range of content types with natural or artificial light, simple or complex objects motion, and different resolution. All sequences are captured with a linear 1D camera array, i.e. cameras axes are parallel, non-convergent. The inter-cameras distance is up to 55cm (e.g. *poznanstreet*) cf. Tab. 1 with a moderately long focal length (23mm on average) which makes the angle of view narrower and the overlap in the field of view smaller. Therefore, we notice large holes in the warped views even in our smallest baseline sequences, like the *"painter"* sequence, as illustrated in Fig. 6. The *Painter* sequence is captured with a 2D camera array, so we extract three linear, non-overlapping, camera setups of this scene for a total of 9 sequences. From each sequence we

Figure 6: Example of large holes (in green) in the *Technicolor painter* sequence with a baseline = 21cm.

extract 3 neighbor views from the first 100 frames: the left and right views are used as references (T^L, T^R) and the central view is used as target view (*i.e* ground-truth) T^C. For each sequence, we preliminary warp T^L and T^R to obtain T^{Lw} and T^{Rw} and then we generate the corresponding filled textures T_f^{Lw} and T_f^{Rw} using the methods described in the previous Sec. 3. Then, from each sequence we randomly extract 10k quintuplets of co-located patches $(\widetilde{t^{lw}}, \widetilde{t_f^{lw}}, t^c, \widetilde{t_f^{rw}}, \widetilde{t^{rw}})$. Each sequence is alternatively reserved for testing for a total of 10k testing patches. The other 8 sequences are used for training and validation purposes, for a total of 70k training patches and 10k validation patches. Such approach guarantees that the test sequence is always left out from the training set, i.e. there is no cross-contamination between train and test sets. All patches used for training are 64×64 as our experiments revealed it allows a favorable tradeoff between patch size and number of non-overlapping patches that can be extracted from the available training video sequences. Patches have fifty-fifty chance to be vertically flipped forming data augmentation. Our experiments reveal that a reasonable trade-off between performance and convergence time can be achieved using batches of 128 patches, and a learning rate of 0.0001, leading to the convergence of our learning algorithm after 100 epochs. In all our experiments, including losses experiments, we use the same hyper-parameters showing the best results. Concerning the parameters optimization algorithm, we rely on Adam, with weight decay = 0 and betas = $(0.9, 0.999)$. Our method is implemented in Pytorch and all the experiments are performed on a server with an NVIDIA RTX2080GPU.

Finally, in all our experiments we measure the quality of the synthesized view both using the PSNR and the SSIM metrics computed over the Y (luma) channel.

4.2. Comparison with prior works

In this subsection we compare our proposed view synthesis method with a number of references and provide a quantitative and qualitative analysis of the synthesized view quality.

4.2.1. Reference schemes

Many of the state-of-the-art methods listed in Sec. 2 are learnable end-to-end architectures in principle comparable with ours. However, while some require in input more than two reference views, others deal only with small baseline views [41], [18]. The pipeline proposed in [12] is comparable to our, however it relies on four warped reference textures and a global mesh of the scene as input to their blending/inpainting network and the source code to reproduce their method is not available. On the other hand, the video frame interpolation method [19] when

applied to our setup did yield very weak performances, we hypothesize because designed for different purposes purposes. For the above reasons, we compare with the following reference schemes:

CNN-VB is the method in [20], it shares with the approach proposed here the preliminary algorithmic warping followed by a learnable blending. Namely, first the the textures are warped to the target view position, then holes are disoccluded. Next, each warped view is first processed by two convolutional layers with 64 and 32 filters each sized 9×9 with ReLU activations. The feature maps produced in output are concatenated and processed by a two convolutional layer with 16 and 1 filters each sized 5×5, the output layer followed by a hyperbolic tangent activation. This architecture notably includes no pooling layers nor other feature downsampling methods and, albeit very primitive, validated the idea of preliminary warping the views to use smaller filters. For these reasons, we keep it as a very baseline reference.

CNN-VB+ is an improved version of CNN-VB that we purposely developed for comparing against HDSB as illustrated in Fig. 7. This scheme improves over CNN-VB in two aspects.
First, after warping and disoccluding the views, we compute two *reliability maps* that signal which pixels are the result of a disocclusion. The two maps are then provided in input to the ConvNet in charge of blending, that exploits the reliability maps as hints to improve the quality of the synthesized view.
Second, we double the depth (8 vs. 4 layers) of the ConvNet in charge of blending, while keeping the parameters count constant. Each pair of warped texture and reliability map is provided in input to a separate branch of four convolutional layers with 64, 50, 42, and 32 filters respectively, all sized 5×5 rather than 9×9. The last four convolutional layers include 50, 42, 32, and 3 filters respectively sized 3×3 in place of 5×5. It can be shown that each feature produced in output by these four convolutional layers enjoy the same receptive field as in the case of the CNN-VB architecture. Therefore, this scheme enjoys a deeper convolutional pipeline able to learn a potentially better representations of the input views albeit leaving untouched complexity and receptive field size. Finally, CNN-VB+ takes in input and outputs RGB rather than grayscale images, so to be able to compare with our proposed HDSB.

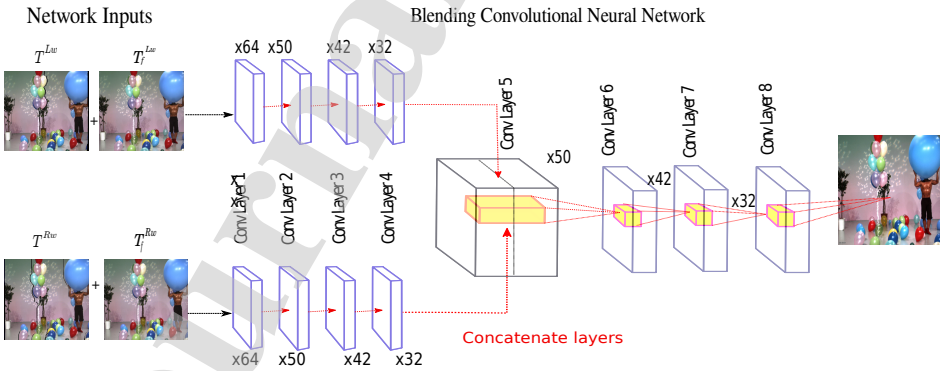


Figure 7: The CNN-VB+ architecture as an improved version of [20]. We use this architecture as a downsampling-less reference against HDSB.

VVS, the MPEG-I view synthesizer reference software described in Sec. 3.1. This method is not learning-based and is completely based on an algorithmic approach. VVS and HDSB share

the basic warping scheme, so VVS is a proper reference to assess the benefits of a learning-based blending stage over an algorithmic based approach.

Synsin, [18] is a purely learnable end-to-end view synthesis method. Synsin allows for synthesizing novel target views of a scene given a single image only, using generative adversarial networks (GAN) techniques and a new differentiable point cloud renderer. In our experiments, it is refined on our training set following the same procedure described for the other methods to allow for a fair comparison.

4.2.2. Results and discussion

In Tab. 2 and 3, we compare our proposed method HDSB with the references above. It is clear that Synsin shows weak performances in all sequences, in terms of visual quality and objective quality (cf. Tab. 2,Tab. 3 and Fig. 8). The results are linked to the use of only one reference image unwarped as input to the network, to generate a high-resolution novel view located at a long distance from the reference view. A possible explanation to this result could be that, this method is effective on small baseline cases and lower image resolution. Concerning CNN-VB and CNN-VB+, the latter improves over the former in all sequences, with gains in excess of 0.5 dB for *PoznanStreet* and *Lovebirds*. Similarly, CNN-VB+ outperforms MPEG VVS almost for all sequences, with a 0.8 dB gain for *Kendo*. We attribute such gains in part to the deeper convolutional architecture, in part to the introduction of the reliability maps. Concerning SSIM, CNN-VB+ outperforms CNN-VB on the average, albeit in some cases CNN-VB scores better. Anyway, a visual inspection of the synthesized view (Fig. 8) shows that even CNN-VB+ produces artifacts in the synthesized views, showing the intrinsic limits of this pooling-less architecture in view synthesis. In the following, we compare HDSB mainly against CNN-VB+, which is the best reference so far.

Tab. 2 shows that HDSB significantly outperforms all references by a significant margin (over 0.5 dB on the average). HDSB scores a top gain of 0.8 dB over CNN-VB+ for the *PoznanHall* sequence and improves by 0.6 dB for *PoznanStreet*. SSIM results show similar trends, with HDSB consistently outperforming every reference. We hypothesize that such gains are mainly due to the downsampling and upsampling of the feature maps performed by the encoder and the decoder respectively. In addition, we hypothesize that residual blocks in the bottleneck may have also a role in such gain. Concerning the visual assessment, Fig. 8 illustrates how HDSB improves over the three references. For *PoznanStreet*, for example, HDSB preserves the hanging lines on the image background, that are otherwise lost by the reference methods. Similarly, the shape of the black pole in the foreground looks much more like the ground truth as synthesized by HDSB. For *PoznanHall*, our method remarkably approaches the desired outcome in the reconstruction of the stairs and the handrail, or the clarity of the exit green plate on the wall.

Concerning computational complexity, inference and training times are as follows. The inference time of CNN-VB+ network is around few seconds ($2-3$ sec) per frame, and around 8 hours for the network to converge in the training process with the MSE loss over 100 epochs. The inference time of our network HDSB is on average 4.32s/frame, and the required training time the network to converge is 12 hours using MSE loss on 100 epochs. For both methods, training time drops by 3 times using the perceptual loss. Finally, Synsin inference time lies between 2 and 10 seconds per frame depending on image resolution, while training required 3 days to converge over 350 epochs. About the purely algorithmic VVS, the time to synthesize one frame is around 100 sec, where 83% of this time is due to inpainting/blending.

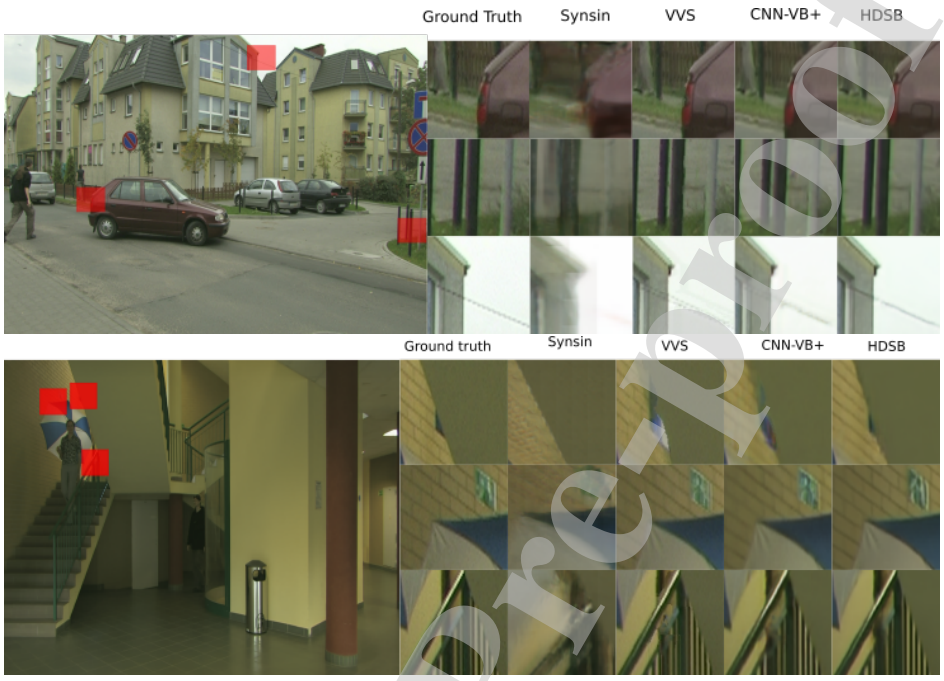


Figure 8: Details from *PoznanStreet* (top) and *PoznanHall* (bottom) sequences. First column is ground truth, second column is Synsin, third column is VVS, fourth column is CNN-VB+, and the last column is the proposed HDSB.

4.3. Ablation studies

In this subsection, we alternatively ablate one element from our HDSB architecture and we assess the effect on the synthesized view quality.

4.3.1. Encoder-decoder architecture

As a first ablation study, we explore the advantages of the encoder-decoder architecture with feature map downsampling implemented by HDSB. Namely, we modify HDSB avoiding to downsample the feature maps by reducing the stride of the filters in the convolutional layers to 1 pixel (we refer to this architecture as "Wo/EnDe"). In other words, Wo/EnDe is such that all the feature maps produced by the hidden convolutional layers have the same size as the input and output images. This architecture is composed of three convolutional layers, operating independently on the left and the right views with 64 filters each and kernel size 7×7, 3×3, and 3×3 respectively. The bottleneck is composed of 8 residual blocks, and the last convolutional layer is composed of 3 filters with 7×7 kernel size. The only difference with respect to HDSB is the removal of the down-sampling. Indeed we preserve the same number of convolutional layers, the residual blocks, and the number of the learnable parameters to study the impact of an encoder-decoder architecture on the training/testing processes.
We train and test this architecture from scratch according to the same procedure used for HDSB. However, due to the lack of downsampling, this architecture has a larger memory footprint which

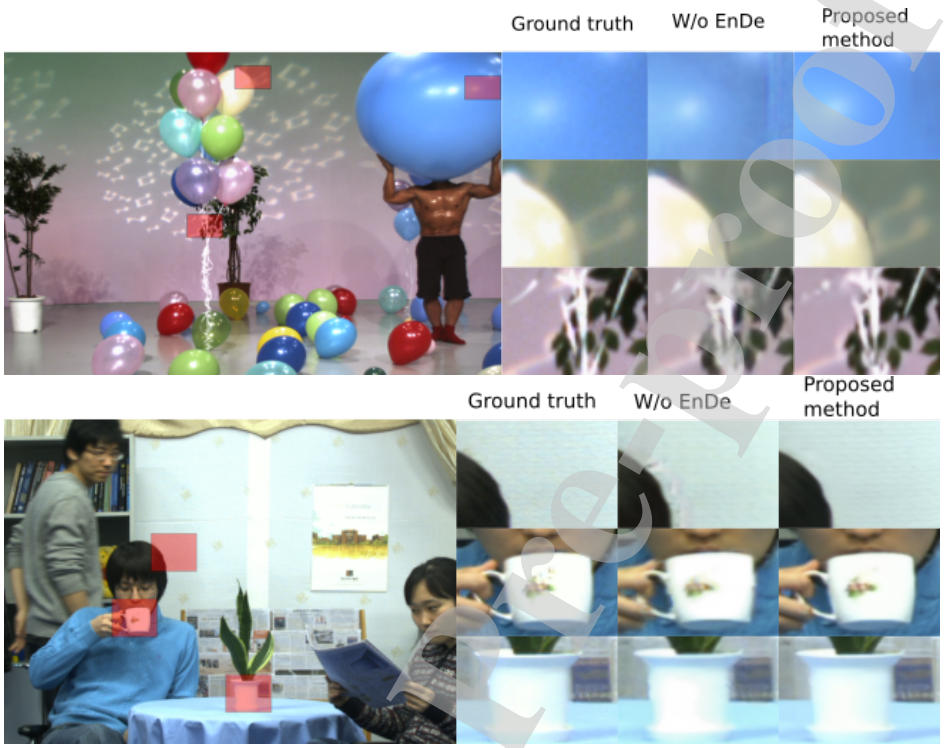


Figure 9: Ablation of the feature map downsampling: *Balloons* (top) and *Newspaper* (bottom). First column is the ground truth, second column is the architecture without encoder-decoder, and last column is our proposed architecture.

forced us to reduce the batch size from 128 to 80 samples. Preliminary experiments show that such architectures slows down the training process and increases the convergence time.

In Tab. 4, the experiments show how adopting an architecture without an encoder-decoder reduces in average the PSNR and the SSIM by almost 0.5 dB and 0.023 respectively on our test sequences. We illustrated the results in Fig. 9 for *Balloons* and *Newspaper* respectively. We notice that the architecture without encoder-decoder (refered to in Tab. 4 as Wo/EnDe) do not generalize well, and tends to overfit on the training data.

Indeed, HDSB our method focuses during training on fewer number of activation points to reduce redundancy in feature maps. It also yields the network output to be more tolerant for small translational changes in input images, which means that an encoder-decoder architecture can tolerate equivariances in input images produced due to the warping process. Indeed the two reference views are warped to the same target position, however they do not originally share the same lighting and angles conditions.

4.3.2. Hole inpainting experiments

We also elaborate on the effect of using different additional inputs to our network rather than the filled textures with the warped images. The filled textures T_f^{Lw} and T_f^{Rw}, were introduced in

our proposed approach to strengthen the inpainting task in the network, and thus by better filling the occluded pixels in the output texture, see Sec. 3.2.

We thus, experiment our architecture using the two reference warped textures and their corresponding binary masks (M_b^L, M_b^R) as introduced in Sec. 3.2, as inputs, and we compare it with our method (with filled textures).

We notice that using filled textures yields to better PSNR performances than using binary masks on all our test sequences in table 5. Whereas, for the SSIM we did not notice any distinction. However, the improvements achieved in terms of PSNR are not easily visible with the naked eye, and thus the visual quality difference is indistinguishable between the two methods on our test datasets. Notably, we significantly raise the computational speed and we lower the memory consumption by using binary masks. Therefore, for the sake of simplicity, we use the binary masks as inputs to our network in the following experiments.

4.3.3. Loss functions experiments

In this section we consider the effect of changing the loss function used to train our network. All the results presented previously in this paper come from our neural network trained using a per-pixel loss function, the MSE. We re-train the proposed architecture on the same training datasets, but we use instead the perceptual loss function, detailed in Sec. 3.3.2, essential for the training convergence. The experiment shows that high-quality visual images are generated when the perceptual loss is minimized . As well as it increases the SSIM on all the sequences, cf. Tab. 6, over 0.003 in average. With the perceptual loss function the computation of the loss between the output and the desired image is based on the image content and style rather than on the individual pixel values, and thus we expect that the PSNR will decrease. However, we notice that when the network is optimized towards another metric the PSNR decreases of 2 dB as shown in Tab. 6. Finally, in terms of complexity, the training with perceptual loss is three times faster than with MSE. In all our experiments in this work, we used the MSE loss function to evaluate and compare our results using both perceptual SSIM and non perceptual PSNR quality metrics. In the end, we observed that perceptual loss leads to visually pleasant synthesized images, while somehow reducing the training time. On the other hand, MSE is more appropriate when objectively benchmarking methods in PSNR terms.

4.3.4. Effect of the encoder architecture

Unlike image-to-image mapping monoscopic architectures that feature just one encoder, our neural architecture features one one encoder for the left view and one for the right view. Such design choice is motivated by the observation that disocclusion artifacts in the two warped views lie on opposite side of the objects. For the same reason, we proposed the *flip-convolve-flip* approach to be able to share parameters among encoders. We now experiment with two different encoder typologies as follows.

First, we consider a HDSB variant where we drop the two *convolve-flip-convolve* encoders with shared parameters in favor of a single encoder. This scheme leaves inaltered the number of learnable parameters in the network, however the encoder takes in input both left and right warped views and relative masks. That is, the encoder now faces the challenge of dealing with occlusions potentially on both sides of the objects. This scheme is referred to as *HDSB-1E* in the following.

Second, we consider another HDSB variant where the encoders do not share parameters, i.e. each encoder indipendently processes the left or the right view. This scheme has the potential to deliver better performance as the network includes more learnable parametrs and each encoder

learns specialized feature for each view. Obviously, in this case the right view is not mirrored anymore. This scheme is referred to as *HDSB-2E* in the following. We train and test both architectures as for HDSB, and we show the results of these experiments in Tab. 7.

Concerning HDSB-1E, the quality of the synthesized view is usually lower than HDSB. For example, for *Balloons* HDSB-1E scores 35.92 dB against 37.59 dB of HDSB, i.e. HDSB scores almost 0.5 dB higher. While for some sequences (e.g.: Kendo, PoznanHall) HDSB-1E scores marginally better, on the average HDSB scores almost 0.5 dB higher on the average. In terms of SSIM, HDSB-1E always score worse than HDSB. We recall that HDSB and HDSB-1E count the same number of parameters, nevertheless the inspection of the training curves shows that the loss function of HDSB-1E flutters more. We attribute such results to the complexity of the single encoder to deal with occlusions on both sides of the objects.

Concerning HDSB-2E, it outperforms HDSB-1E almost for any sequence yet it outperforms HDSB only for *Painter-1*. We recall that HDSB-2E counts twice as many parameters in the encoder as HDSB-1E. The analysis of the training curves show that HDSB-2E is more likely to overfit to the training data in reason of the higher parameters count. We hypothesize that if significantly more training sequences are available, HDSB-2E may have an edge over HDSB.

5. Conclusions

We presented a hybrid approach to wide baseline view synthesis where the warping is algorithmic while the blending is learnable and inspired by image-to-image convolutional architectures. Extensive experiments on real multi-view video sequences show better performance that pure-algorithmic approaches while avoiding the complexity of purely learning-based approaches and taught us some lessons. First, an encoder-decoder architecture improves over our previous super-resolution based method by projecting the input features over a spatially subsampled latent feature space. Second, the encoder complexity can be reduced by resorting to a smart *flip-convolve-flip* approach that allows us to share parameters among encoders reducing the network complexity. Third, providing additional filled textures to the blender helps preventing disocclusions-induced artifacts better than binary masks. Finally, experiments with perceptual loss show visually pleasant images, yet at the expense of a drop in objective visual quality. Our current research aims at exploiting the temporal information to improve the hole filling procedure and to impose temporal consistency among neighboring frames as an additional constraint at training and inference times.

References

[1] G. Petrazzuoli, M. Cagnazzo, F. Dufaux, B. Pesquet-Popescu, Using distributed source coding and depth image based rendering to improve interactive multiview video access, in: IEEE International Conference on Image Processing, Vol. 1, Bruxelles, Belgium, 2011, pp. 605–608.

[2] K. Müller, P. Merkle, T. Wiegand, 3-d video representation using depth maps, Proceedings of the IEEE 99 (4) (2010) 643–656.

[3] J. Jung, P. Boissonade, VVS: Versatile View Synthesizer for 6-DoF Immersive Video, working paper or preprint (Apr. 2020).
URL https://hal.archives-ouvertes.fr/hal-02541110

[4] M. Domański, A. Dziembowski, T. Grajek, A. Grzelka, K. Klimaszewski, D. Mieloch, R. Ratajczak, O. Stankiewicz, J. Siast, J. Stankowski, K. Wegner, Free-viewpoint television demonstration for sports events, ISO/IEC JTC1/SC29/WG11, Doc. MPEG M 41994 (2018) (2018).

[5] S. Li, C. Zhu, M. Sun, Hole filling with multiple reference views in dibr view synthesis, IEEE Transactions on Multimedia 20 (8) (2018) 1948–1959.

[6] G. Luo, Y. Zhu, Z. Weng, Z. Li, A disocclusion inpainting framework for depth-based view synthesis, IEEE transactions on pattern analysis and machine intelligence 42 (6) (2019) 1289–1302.

[7] F. Dufaux, B. Pesquet-Popescu, M. Cagnazzo, Emerging technologies for 3D video: creation, coding, transmission and rendering., John Wiley & Sons, 2013.

[8] C. Fehn, Depth-image-based rendering (dibr), compression, and transmission for a new approach on 3dtv, in: Stereoscopic Displays and Virtual Reality Systems XI, Vol. 5291, International Society for Optics and Photonics, 2004, pp. 93–104.

[9] J. Flynn, M. Broxton, P. Debevec, M. DuVall, G. Fyffe, R. Overbeck, N. Snavely, R. Tucker, Deepview: View synthesis with learned gradient descent, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 2367–2376.

[10] J. Flynn, I. Neulander, J. Philbin, N. Snavely, Deepstereo: Learning to predict new views from the world's imagery, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 5515–5524.

[11] M. Gotfryd, K. Wegner, M. Domański, View synthesis software and assessment of its performance, ISO/IEC JTC1/SC29/WG11, MPEG (2008) M15672.

[12] P. Hedman, J. Philip, T. Price, J.-M. Frahm, G. Drettakis, G. Brostow, Deep blending for free-viewpoint image-based rendering, ACM Transactions on Graphics (TOG) 37 (6) (2018) 1–15.

[13] Z. Liu, R. A. Yeh, X. Tang, Y. Liu, A. Agarwala, Video frame synthesis using deep voxel flow, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 4463–4471.

[14] S. Niklaus, F. Liu, Context-aware synthesis for video frame interpolation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 1701–1710.

[15] Z. Liu, R. A. Yeh, X. Tang, Y. Liu, A. Agarwala, Video frame synthesis using deep voxel flow, CoRR abs/1702.02463 (2017). arXiv:1702.02463.
URL http://arxiv.org/abs/1702.02463

[16] N. K. Kalantari, T.-C. Wang, R. Ramamoorthi, Learning-based view synthesis for light field cameras, ACM Transactions on Graphics (TOG) 35 (6) (2016) 1–10.

[17] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.

[18] O. Wiles, G. Gkioxari, R. Szeliski, J. Johnson, Synsin: End-to-end view synthesis from a single image, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 7467–7477.

[19] W. Bao, W.-S. Lai, C. Ma, X. Zhang, Z. Gao, M.-H. Yang, Depth-aware video frame interpolation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 3703–3712.

[20] N. Hobloss, A. Purica, A. Fiandrotti, M. Cagnazzo, R. Cozot, W. Hamidouche, A hybrid approach to wide baseline view synthesis with convolutional neural networks, in: 2019 International Conference on 3D Immersion (IC3D), IEEE, 2019, pp. 1–7.

[21] P. Isola, J.-Y. Zhu, T. Zhou, A. A. Efros, Image-to-image translation with conditional adversarial networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1125–1134.

[22] J.-Y. Zhu, T. Park, P. Isola, Efros, A. A., Unpaired image-to-image translation using cycle-consistent adversarial networks, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 2223–2232.

[23] M. Tanimoto, T. Fujii, K. Suzuki, N. Fukushima, Y. Mori, Reference softwares for depth estimation and view synthesis, ISO/IEC JTC1/SC29/WG11 MPEG 20081 (2008) M15377.

[24] E. Penner, L. Zhang, Soft 3d reconstruction for view synthesis, ACM Transactions on Graphics (TOG) 36 (6) (2017) 1–11.

[25] A. Canclini, F. Malapelle, M. Marcon, S. Tubaro, A. Fusiello, View-synthesis from uncalibrated cameras and parallel planes, Signal Processing: Image Communication 79 (2019) 40 – 53. doi:https://doi.org/10.1016/j.image.2019.08.012.
URL http://www.sciencedirect.com/science/article/pii/S0923596518309780

[26] X. Xu, L.-M. Po, K.-H. Ng, L. Feng, K.-W. Cheung, C.-H. Cheung, C.-W. Ting, Depth map misalignment correction and dilation for dibr view synthesis, Signal Processing: Image Communication 28 (9) (2013) 1023 – 1045. doi:https://doi.org/10.1016/j.image.2013.04.003.
URL http://www.sciencedirect.com/science/article/pii/S0923596513000593

[27] J. Xiao, M. M. Hannuksela, T. Tillo, M. Gabbouj, C. Zhu, Y. Zhao, Scalable bit allocation between texture and depth views for 3-d video streaming over heterogeneous networks, IEEE Transactions on Circuits and Systems for Video Technology 25 (1) (2015) 139–152. doi:10.1109/TCSVT.2014.2334011.

[28] C. Yao, J. Xiao, T. Tillo, Y. Zhao, C. Lin, H. Bai, Depth map down-sampling and coding based on synthesized view distortion, IEEE Transactions on Multimedia 18 (10) (2016) 2015–2022. doi:10.1109/TMM.2016.2594145.

[29] Y. Lai, X. Lan, Y. Liu, N. Zheng, An efficient depth image-based rendering with depth reliability maps for view synthesis, Journal of Visual Communication and Image Representation 41 (2016) 176 – 184. doi:https://doi.org/10.1016/j.jvcir.2016.09.015.
URL http://www.sciencedirect.com/science/article/pii/S1047320316302048

[30] J. Lin, W. Wang, J. Yao, T. Guo, E. Chen, Q. F. Yan, Fast multi-view image rendering method based on reverse search for matching, Optik 180 (2019) 953 – 961. doi:https://doi.org/10.1016/j.ijleo.2018.12.003.
URL http://www.sciencedirect.com/science/article/pii/S0030402618319272
[31] G. Chaurasia, S. Duchene, O. Sorkine-Hornung, G. Drettakis, Depth synthesis and local warps for plausible image-based navigation, ACM Transactions on Graphics (TOG) 32 (3) (2013) 1–12.
[32] L. Zhu, Y. Zhang, M. Yu, G. Jiang, S. Kwong, View-spatial–temporal post-refinement for view synthesis in 3d video systems, Signal Processing: Image Communication 28 (10) (2013) 1342 – 1357. doi:https://doi.org/10.1016/j.image.2013.08.005.
URL http://www.sciencedirect.com/science/article/pii/S0923596513001227
[33] J. Thatte, B. Girod, A statistical model for disocclusions in depth-based novel view synthesis, in: 2019 IEEE Visual Communications and Image Processing (VCIP), IEEE, 2019, pp. 1–4.
[34] S. Li, C. Zhu, M.-T. Sun, Hole filling with multiple reference views in dibr view synthesis, IEEE Transactions on Multimedia 20 (8) (2018) 1948–1959.
[35] J. Shade, S. Gortler, L.-w. He, R. Szeliski, Layered depth images, in: Proceedings of the 25th annual conference on Computer graphics and interactive techniques, 1998, pp. 231–242.
[36] P.-J. Lee, et al., Adaptive edge-oriented depth image smoothing approach for depth image based rendering, in: 2010 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), IEEE, 2010, pp. 1–5.
[37] C. Yao, T. Tillo, Y. Zhao, J. Xiao, H. Bai, C. Lin, Depth map driven hole filling algorithm exploiting temporal correlation information, IEEE Transactions on Broadcasting 60 (2) (2014) 394–404. doi:10.1109/TBC.2014.2321671.
[38] S. Niklaus, L. Mai, F. Liu, Video frame interpolation via adaptive convolution, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 670–679.
[39] K. Regmi, A. Borji, Cross-view image synthesis using geometry-guided conditional gans, Computer Vision and Image Understanding 187 (2019) 102788. doi:https://doi.org/10.1016/j.cviu.2019.07.008.
URL http://www.sciencedirect.com/science/article/pii/S1077314219301043
[40] I. Choi, O. Gallo, A. Troccoli, M. H. Kim, J. Kautz, Extreme view synthesis, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 7781–7790.
[41] X. Xu, Y.-C. Chen, J. Jia, View independent generative adversarial network for novel view synthesis, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 7791–7800.
[42] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, R. Ng, Nerf: Representing scenes as neural radiance fields for view synthesis, in: ECCV, 2020.
[43] A. Kappeler, S. Yoo, Q. Dai, A. K. Katsaggelos, Video super-resolution with convolutional neural networks, IEEE Transactions on Computational Imaging 2 (2) (June 2016).
[44] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, B. Catanzaro, High-resolution image synthesis and semantic manipulation with conditional gans, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 8798–8807.
[45] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, R. Shah, Signature verification using a" siamese" time delay neural network, in: Advances in neural information processing systems, 1994, pp. 737–744.
[46] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).
[47] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556 (2014).
[48] J. Johnson, A. Alahi, L. Fei-Fei, Perceptual losses for real-time style transfer and super-resolution, in: European conference on computer vision, Springer, 2016, pp. 694–711.
[49] I. organization of standardisation, Coding of moving pictures and audio,iso/iec jtc1/sc29/wg11 mpeg/w18076, macau,sar cn (10- 2018).

Algorithm 1: Hole inpainting algorithm exemplified for view T^{Lw}

Input: (T^{Lw}, T^{Rw})
Output: Filled textures (T_f^{Lw}, T_f^{Rw})

$ListLeft \leftarrow$ the list of occluded pixels in I^{Lw};
$ListRight \leftarrow$ the list of occluded pixels in I^{Rw};
for (r, c) *in* $ListLeft$ **do**
    if *if* (r, c) *is in ListLeft but not in ListRight* **then**
        copy right to left, and remove from ListLeft
    else if (r, c) *is in ListRight but not in ListLeft* **then**
        copy left to right and remove from ListRight;
end
for (r, c) *in ListRight* **do**
    if *if* (r, c) *is in ListRight but not in ListLeft* **then**
        copy left to right, and remove from ListRight
    else if (r, c) *is in ListLeft but not in ListRight* **then**
        copy left to right and remove from ListLeft;
end
for (r, c) *in* $ListLeft$ **do**
    if *neighbor pixels are valid pixels not in ListLeft* **then**
        $T_f^{Lw}(r, c)$ = Median[valid neighbor pixels in T_f^{Lw}]
    else
        continue
    end
end
for (r, c) *in ListRight* **do**
    if *neighbor pixels are valid pixels not in ListRight* **then**
        $T_f^{Rw}(r, c)$ = Median[valid neighbor pixels in T_f^{Rw}]
    else
        continue
    end
end

Table 1: The seven multiview video sequences used in our experiments (all sequences are 100 frames, average camera baseline is 24 cm).

Sequence	Characteristics	Setup	Resolution	(T^L, T^C, T^R)	(T^L, T^R)
Balloons	Indoor, colorful	1D	1024x768	(1,3,5)	20 cm
Kendo	Indoor, white smoke	1D	1024x768	(2,4,6)	20 cm
Newspaper	Indoor, people	1D	1024x768	(2,4,6)	20 cm
PoznanStreet	Outdoor, nature	1D	1088x1920	(2,4,6)	55 cm
PoznanHall	Indoor, building hall	1D	1088x1920	(5,6,7)	27.5 cm
Lovebirds	Outdoor, nature	1D	1024x768	(4,6,8)	15 cm
Painter-1	Indoor, art studio	2D	2048x1088	(0,1,3)	14-21 cm
Painter-5	”	”	”	(4,5,6)	”
Painter-9	”	”	”	(8,9,10)	”

Table 2: Quality of the synthesized view for the proposed and reference methods in terms of PSNR

Sequence	PSNR [dB]				
	Synsin[18]	VVS[49]	CNN-VB[20]	CNN-VB+	**proposed HDSB**
Balloons	20.84	37.07	36.92	37.18	**37.59**
Kendo	21.87	38.21	38.98	39.08	**39.19**
Newspaper	19.28	34.53	35.08	35.35	**35.65**
PoznanStreet	18.16	36.96	36.17	36.81	**37.44**
PoznanHall	18.65	37.26	37.43	37.50	**38.30**
Painter-1	20.45	37.86	37.88	38.01	**38.21**
Painter-5	20.32	38.04	38.12	38.18	**38.87**
Painter-9	20.17	36.36	36.44	36.51	**38.01**
Lovebirds	19.89	34.56	34.70	35.34	**35.54**
Average ± Std	19.9589 ±1.13	36.76 ±1.38	36.85 ±1.41	37.11±1.26	**37.64**±1.28

Table 3: Quality of the synthesized view for the proposed and reference methods in terms of SSIM

Sequence	SSIM				
	Synsin[18]	VVS[49]	CNN-VB[20]	CNN-VB+	**proposed HDSB**
Balloons	0.752	0.922	0.965	0.964	**0.978**
Kendo	0.826	0.972	0.976	0.981	**0.992**
Newspaper	0.875	0.930	0.950	0.951	**0.967**
PoznanStreet	0.728	0.922	0.932	0.956	**0.968**
PoznanHall	0.745	0.921	0.932	0.961	**0.966**
Painter-1	0.841	0.948	0.956	0.955	**0.970**
Painter-5	0.829	0.945	0.953	0.948	**0.961**
Painter-9	0.813	0.930	0.941	0.937	**0.952**
Lovebirds	0.789	0.909	0.933	0.951	**0.973**
Average ± Std	0.7998±0.049	0.933±0.02	0.949±1.16	0.956±0.12	**0.969**±0.01

Table 4: Quality of the synthesized view with our encoder-decoder architecture HDSB and without encoder-decoder architecture

Sequence	PSNR[dB]		SSIM	
	Wo/EnDe	**HDSB**	Wo/EnDe	**HDSB**
Balloons	36.90	**37.59**	0.943	**0.972**
Kendo	38.71	**39.19**	0.966	**0.985**
Newspaper	35.25	**35.65**	0.942	**0.962**
PoznanStreet	36.76	**37.44**	0.947	**0.965**
PoznanHall	37.30	**38.30**	0.960	**0.965**
Painter-1	38.04	**38.21**	0.958	**0.964**
Painter-5	38.39	**38.87**	0.944	**0.958**
Painter-9	37.58	**38.01**	0.933	**0.949**
Lovebirds	35.11	**35.54**	0.896	**0.969**
Average ± Std	37.12±1.27	**37.56**±1.35	0.943±0.02	**0.966**±0.01

Table 5: Quality of the synthesized view using the filled textures and the binary masks.

Sequence	PSNR[dB]	
	HDSB	Wo binary mask
Balloons	37.24	**37.59**
Kendo	39.12	**39.19**
Newspaper	35.53	**35.65**
PoznanStreet	37.22	**37.44**
PoznanHall	38.25	**38.30**
Painter-1	38.09	**38.21**
Painter-5	38.85	**38.87**
Painter-9	37.95	**38.01**
Lovebirds	35.41	**35.54**
Average ± std	37.56±1.35	**37.63**±1.36

Table 6: Quality of the synthesized view using two different loss functions during the training process.

Sequence	SSIM for HDSB		PSNR for HDSB	
	w/per-pixel loss	w/perceptual loss	w/per-pixel loss	w/perceptual loss
Balloons	0.972	**0.980**	**37.24**	35.84
Kendo	0.985	**0.989**	**39.12**	37.16
Newspaper	0.962	**0.968**	**35.53**	33.35
PoznanStreet	0.965	**0.966**	**37.22**	34.27
PoznanHall	0.965	**0.968**	**38.25**	36.41
Painter-1	0.964	**0.965**	**38.09**	37.03
Painter-5	0.958	**0.962**	**38.85**	37.18
Painter-9	0.949	**0.957**	**37.95**	36.22
Lovebirds	0.969	**0.971**	**35.41**	32.94
Average ± std	0.966±0.01	**0.969**±0.01	**37.56**±1.35	35.6±1.65

Table 7: Effect of the encoder architecture: HDSB-1E includes just one shared encoder for both views, HDSB-2E includes separated encoders for each view.

Sequence	PSNR			SSIM		
	HDSB	**HDSB-1E**	**HDSB-2E**	HDSB	**HDSB-1E**	**HDSB-2E**
Balloons	**37.24**	35.92	37.03	**0.978**	0.967	0.972
Kendo	39.12	**39.30**	39.20	**0.992**	0.990	0.985
Newspaper	**35.53**	35.24	35.46	**0.967**	0.965	0.962
PoznanStreet	**37.22**	36.3	37.18	**0.968**	0.958	0.965
PoznanHall	38.25	**38.32**	38.31	**0.966**	0.964	0.965
Painter-1	38.09	38.01	**38.18**	**0.970**	0.963	0.964
Painter-5	**38.85**	38.25	38.81	**0.961**	0.960	0.958
Painter-9	**37.95**	37.55	37.67	**0.952**	0.941	0.949
Lovebirds	**35.41**	34.51	34.92	**0.973**	0.962	0.969
Average	**37.52**	37.04	37.42	**0.970**	0.963	0.966

Highlights

- Free navigation systems allow users to freely browse a scene by arbitrarily changing viewpoint.
- Missing viewpoints are usually synthesized exploiting the captured neighboring view.
- Free navigation of a scene requires warping some reference views to some desired target viewpoint and blending them to synthesize a virtual view.
- Convolutional Neural Networks (ConvNets) based methods can learn both the warping and blending tasks jointly.
- Preliminary view warping allows reducing the size of the convolutional kernels and thus the learnable parameters count.
- A residual encoder-decoder can be used for image blending with a Siamese encoder to further keep the parameters count low.

AUTHORSHIP STATEMENT

Manuscript title: Hybrid Dual Stream Blender for wide baseline view synthesis

All persons who meet authorship criteria are listed as authors, and all authors certify that they have participated sufficiently in the work to take public responsibility for the content, including participation in the concept, design, analysis, writing, or revision of the manuscript. Furthermore, each author certifies that this material or similar material has not been and will not be submitted to or published in any other publication before its appearance in the *Hong Kong Journal of Occupational Therapy*.

Authorship contributions
Please indicate the specific contributions made by each author (list the authors' initials followed by their surnames, e.g., Y.L. Cheung). The name of each author must appear at least once in each of the three categories below.

Category 1
Conception and design of study: Nour Hobloss, Stéphane Lathuilière, Attilio Fiandrotti, _____;

acquisition of data: Nour Hobloss, Marco Cagnazzo, Attilio Fiandrotti, _____;

analysis and/or interpretation of data: Nour Hobloss, Marco Cagnazzo, Attilio Fiandrotti, Stéphane Lathuilière.

Category 2
Drafting the manuscript: Nour Hobloss, _____, _____, _____;

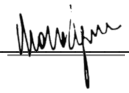
revising the manuscript critically for important intellectual content: Attilio Fiandrotti, Marco Cagnazzo, Stéphane Lathuilière, Nour Hobloss.

Category 3
Approval of the version of the manuscript to be published (the names of all authors must be listed):

Marco Cagnazzo, Attilio Fiandrotti, Stéphane Lathuilière, Nour Hobloss, Lu Zhang, _____, _____, _____, _____, _____.

Acknowledgements
All persons who have made substantial contributions to the work reported in the manuscript (e.g., technical help, writing and editing assistance, general support), but who do not meet the criteria for authorship, are named in the Acknowledgements and have given us their written permission to be named. If we have not included an Acknowledgements, then that indicates that we have not received substantial contributions from non-authors.

This statement is signed by all the authors (*a photocopy of this form may be used if there are more than 10 authors***):**

Author's name (typed)	Author's signature	Date
Stéphane Lathuilière		17 Mai 2021
Attilio Fiandrotti		17 Mai 2021
Marco Cagnazzo		17 Mai 2021
Lu Zhang		17 Mai 2021
Nour Hobloss		17 Mai 2021

Conflicts of Interest Statement

Manuscript title: Hybrid Dual Stream Blender for wide baseline view synthesis

The authors whose names are listed immediately below certify that they have NO affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers' bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing arrangements), or non-financial interest (such as personal or professional relationships, affiliations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript.

Author names:
Marco Cagnazzo
Attilio Fiandrotti
Stéphane Lathuilière
Nour HOBLOSS

The authors whose names are listed immediately below report the following details of affiliation or involvement in an organization or entity with a financial or non-financial interest in the subject matter or materials discussed in this manuscript. Please specify the nature of the conflict on a separate sheet of paper if the space below is inadequate.

Author names:

This statement is signed by all the authors to indicate agreement that the above information is true and correct (*a photocopy of this form may be used if there are more than 10 authors*):

Author's name (typed)	Author's signature	Date
Marco Cagnazzo		01/12/2020
Attilio Fiandrotti		01/12/2020
Stéphane Lathuilière		01/12/2020
Nour HOBLOSS		01/12/2020