Deep Residual Architecture Using Pixel and Feature Cues for View Synthesis and Temporal Interpolation

Jinglei Shi, Xiaoran Jiang, Christine Guillemot

Deep Residual Architecture Using Pixel and Feature Cues for View Synthesis and Temporal Interpolation

Jinglei Shi, Xiaoran Jiang, Christine Guillemot *Fellow, IEEE*

***Abstract*—In this paper, we propose a deep residual architecture that can be used both for synthesizing high quality angular views in light fields and temporal frames in classical videos. The proposed framework consists of an optical flow estimator optimized for view synthesis, a trainable feature extractor and a residual convolutional network for pixel and feature-based view reconstruction. Among these modules, the fine-tuning of the optical flow estimator specifically for the view synthesis task yields scene depth or motion information that is well optimized for the targeted problem. In cooperation with the end-to-end trainable encoder, the synthesis block employs both pixel-based and feature-based synthesis with residual connection blocks, and the two synthesized views are fused with the help of a learned soft mask to obtain the final reconstructed view. Experimental results with various datasets show that our method performs favorably against other state-of-the-art (SOTA) methods with a large gain for light field view synthesis. Furthermore, with a little modification, our method can also be used for video frame interpolation, generating high quality frames compared with SOTA interpolation methods.**

***Index Terms*—Light field, view synthesis, pixel and feature cues, residual connection, video frame interpolation.**

I. Introduction

SCENE sampling rate plays a key role in determining the perception quality of visual content like videos or immersive content such as light fields (LF). However, the video or light field sampling rates can be limited due to technological limitations, e.g. the sensor readout speed, its resolution, the storage space or transmission bandwidth [1], [2]. A small video frame rate can lead to jerky motion while undersampled angular light field views have large parallax and can yield perspective discontinuities when navigating within the scene. Video frame rate conversion as well as light field view synthesis have therefore become widely investigated problems with specific solutions.

Existing video frame interpolation (VFI) approaches consist of synthesizing intermediate frame from given input frames and they can be roughly classified into three categories, i.e. optical flow [3]–[7] or feature flow-based schemes [8], kernel-based schemes [9]–[11] and phase-based ones [12], [13].

Light field view synthesis (LFVS) methods can similarly be roughly categorized as depth-dependent methods and depth-independent ones. Depth-dependent schemes [14]–[16] consist in first estimating scene depth which is then used for image warping. Kalantari *et al.* [14] use two cascaded networks respectively dedicated to depth prior estimation and color synthesis. Penner *et al.* [15] propose instead a soft 3D reconstruction method in which a confidence measure of depth helps for better handling boundary artifacts during the synthesis. Mildenhall *et al.* [16] slice the scene into Multi-Plane Images (MPIs) at different depths, then warp and fuse these planes according to the corresponding depth clues. Another method in [17] relies on Epipolar-Plane Images (EPI) shearing leading to a stack of sheared EPI which can be seen as a plane sweep volume as used in [15] and [16]. The above methods, that we refer here as depth-dependent methods, rely on geometry estimation for correctly warping images from one viewpoint to another. Other methods, that we refer here as depth-independent methods, do not require a prior geometry estimation. They instead retrieve views by exploiting signal priors such as sparsity [18] or smoothness of EPIs [19], [20]. The Neural Radiance Field (NeRF) concept has been introduced in [21] to model the mapping between the 5D spatial and angular coordinates of light rays emitted by the scene into its three RGB colour components and a volume density measure. This mapping is learned for each scene to be processed from densely captured images, using a multi-layer perceptron and assuming that the camera pose parameters are known. The model can then be efficiently used for synthesizing novel light field views. The neural radiance field learning has been further optimized in [22] to avoid prior computation of camera pose parameters, in [23] and [24] to enable faster inference as well as using a sparser set of input views, and to enable generalization to new scenes.

In this paper, we describe a deep residual network architecture that can be used for both light field view synthesis (LFVS) and temporal video frame interpolation (VFI). For the light field view synthesis problem, the proposed architecture takes a sparse subset of light field views and output a whole light field at any desired angular sampling rate. The proposed architecture builds upon the FPFR solution we proposed in [25]. Both approaches adopt the idea of merging a pixel-based view reconstruction and a feature-based reconstruction. However, the architecture in [25] suffers from limitations in terms of performance, slow convergence and training instability. We propose in this paper very effective improvements both in terms of architecture design and training schedule in order to address each of these limitations, as follows:

- We introduce a light-weight feature extractor (1.9M vs 2.4M parameters) which consumes less memory and can be trained end-to-end to extract deep features that are better suited for the reconstruction problem at hand. This

This project has been supported by the EU H2020 Research and Innovation Programme under grant agreement No 694122 (ERC advanced grant CLIM). The authors are with INRIA, Campus Universitaire de Beaulieu, 35042 Rennes, France. Contact: firstname.lastname@inria.fr

improves both the performance and the convergence of the entire network.
- We replace the convolutional layers of PixRNet and FeatRNet in [25] by residual convolutional blocks. On one hand, the residual blocks deepen PixRNet and FeatRNet, enhancing their synthesis capability. On the other hand, the residual connections in the convolutional blocks guarantee convergence speed of the networks (by avoiding gradient vanishing). These residual blocks hence also improve performance and network convergence.
- The weights of the optical flow estimator are optimized for the view synthesis problem or the temporal frame interpolation problem, in an un-supervised manner, i.e. without using ground truth optical flows or disparity maps. This operation is essential for the VFI task, since a flow estimator trained on synthetic data (for which ground truth motion is available), when applied to real-world frames, may lead to network collapse. The estimator optimized for view synthesis improves the training stability.

Note that there is a strong similarity between the LFVS and VFI tasks. Light field views can indeed be seen as frames of a video, in the special case of a static scene captured with a moving camera. Therefore, we further show how the proposed architecture can be adapted to the temporal video interpolation problem, where the scenes can be dynamic with moving objects.

We show that this novel architecture gives synthesized views of a higher quality compared with recent reference methods, such as DeepVS [14], Soft3D [15], LLFF [16], EPI [17], and FPFR [25], for both dense and sparse light fields. Our comparative analysis with the NeRF-based view synthesis method [21], for different input view configurations, show that the NeRF method performs quite well, provided the radiance field is learned from a sufficient number of input views, but that the proposed method gives better synthesis results for a small number of input views. This advantage is further analysed in light of storage or compression implications. We indeed show the PSNR-rate benefits of the proposed method compared with NeRF and with several compression methods. Finally, we show that the proposed architecture can also be used for video frame rate conversion with high quality which is quite competitive compared with the one obtained with reference methods, e.g., [10], [4], [11], [8] and [7], specifically dedicated to the temporal frame interpolation task.

II. Related work

Light field view synthesis and video frame temporal interpolation methods follow very similar principles. Although developed in parallel with methods specific to each problem, they can be similarly classified in two categories. One category of methods first estimates disparity maps or optical flows which are then used to warp input images in either the angular or temporal dimensions. A second category of methods does not rely on explicit depth or motion information but instead replies on signal priors that are specific to the input data characteristics.

A. Light field view synthesis

1) Depth-dependent schemes: Depth-dependent schemes follow the principles of image based rendering methods, as e.g. in [15], [26]–[28], which have been prevailing for many years in the field of view synthesis. Along the same lines, but using deep neural networks for both depth estimation and view synthesis, one finds the approach of Kalantari *et al.* [14] in which the authors sequentially connect two convolutional neural networks (CNNs) dedicated respectively to depth estimation and color fusion. The depth estimation network exploits cues that are well suited for dense light fields with small parallax. Hence, the method, mostly designed for dense structured light fields, fails in occluded regions, especially for sparse light fields having large baselines. Due to depth imprecision, the synthesized views also suffer from blurriness, tearing and ghosting effects. The method in [25] first estimate disparity using a learned optical flow estimator which is then used to warp not only the input views but also feature representations of these views. This leads to pixel-based and feature-based synthesized views which then merged using a learned soft mask. This method gives state-of-the-art results.

To cope with imprecision of disparity estimators, the concept of Multi-Plane Image (MPI) representation has been introduced in [29] for stereo views, and then used in [16] to render novel views from irregularly sampled views in unstructured light fields. In [16], an MPI representation is learned for each source view, from plane sweep volumes (PSVs), and the source MPIs are then warped and merged to synthesize the target view. MPIs exploit notions of visibility or transparency with alpha blending maps [29], [30]. In the same vein, Wu *et al.* [17] construct a stack of sheared EPIs, that can be seen as a PSV, which are then fused to reconstruct light fields. They train a CNN to give a score which is then used for the fusion of sheared EPIs.

2) Depth-independent schemes: The use of various signal priors has also been investigated for view synthesis, often regarding the problem as a problem of angular super-resolution. For example, the authors in [18] reconstruct dense light field views from a subset of samples by exploiting sparsity of the continuous Fourier spectrum. The authors in [19] apply a shearlet transform-based inpainting technique on EPIs to generate densely sampled light fields, EPI per EPI. Variational methods imposing some smoothness contraints on EPI have also been considered for view synthesis [31]. Similarly, the authors in [20] use a pseudo 4DCNN to upsample the input light field in the angular dimension. Boundary artifacts appear when the target views are far from the source views. The above schemes are effective on narrow-baseline light fields, but usually fail with those having large baselines. The authors in [32] describe a spatio-angular restoration network using 4D convolutions and high-resolution residual blocks, in order to extract spatial features that preserve geometrical properties. To further exploit scene geometry, and better handle occlusions and non lambertian scenes, the method in [21] optimizes a 5D neural radiance field representation, with volume density and view-dependent color at any location of a scene, from a

set of input images. The authors then use volume rendering techniques to sample the scene representation along rays, and render the scene from any viewpoint. This seminal work gave rise to several variants of NeRF learning methods, such as the so-called MVSNeRF [23] that aims at reconstructing radiance fields from a sparse set of views and via faster inference, or such as NeRF– [22] that avoids a prior estimation of camera pose parameters. Stereo radiance fields (SRF) are also introduced in [24] that can generalize the learning of the radiance field to new scenes.

B. Video frame interpolation

1) Flow-based interpolation: Flow-based methods consists in first estimating optical flows between given frames, and then in interpolating or extrapolating target frames along the motion vectors. This is the case of the Deep Voxel Flow (DVF) in [3] in which a network is trained to estimate the optical flows and produce a temporal mask for trilinear interpolation of the input frames. The authors in [4] use a first U-Net network to predict bi-directional optical flows which are then combined to approximate the intermediate optical flows. A second U-Net refines the intermediate optical flows as well as predicts soft visibility maps in order to warp and linearly merge the input frames. In [6], [7], the authors instead employ the state-of-the-art PWC-Net [33] for optical flow estimation. They then warp input frames and corresponding deep features to generate the interpolated frames. The problem of occlusion handling is further addressed in [5] by using depth information in addition to optical flows, together with local interpolation kernels. A recent method in [8] uses a multi-flow multi-attention generator to estimate feature flows instead of optical flows. The feature flows are then used to warp and interpolate feature maps of the input frames into the target position. Instead of considering only two input frames for interpolating an intermediate frame, the authors in [34] propose a framework utilizing three frames to remove tearing and ghosting artifacts of moving objects.

2) Flow-free interpolation: A second category of methods can be referred to as flow-free interpolation methods. This is the case of kernel-based methods [9], [10] which, instead of estimating optical flows, estimate a series of spatially-adaptive interpolation kernels used to convolve the input frames in order to produce the interpolated ones. Besides the effort for improving the estimation of kernels reported in [9], [10], the authors in [35] also propose a set of techniques that can be applied to kernel-based methods to further improve the performance. Kernel-based methods are computationally expensive and do not incorporate explicit mechanisms for occlusion handling. Please note that such spatially-adaptive filters and optical flows can nevertheless be combined as in [11]. Phase-based methods [12], [13] represent the motion as a per-pixel shift and operate phase modification for each pixel. However, these methods can not reach the same level of details as flow-based methods.

III. Background and notations

In the following we consider both video sequences and light fields. Considering a video sequence, we denote a frame captured at instant t as I_t in the following sections. A light field is represented by a 4D function $L(x, y, u, v)$ [36], where $(x, y) \in [\![1, X]\!] \times [\![1, Y]\!]$ and $(u, v) \in [\![1, U]\!] \times [\![1, V]\!]$ are respectively the light field spatial and angular coordinates. A light field can be interpreted as an array of views observed from different viewpoints, and we denote the view $L(x, y, u_0, v_0)$ observed from a certain viewpoint $\mathbf{i} = (u_0, v_0)$ as $I_{\mathbf{i}}$ (or I_{u_0,v_0}) for sake of notation simplicity.

IV. Methodology

A. Architecture overview

The proposed architecture aims at reconstructing a densely sampled light field from a subset of input views, thus performing angular view synthesis, and we show that it can also be used to reconstruct a video sequence from a subset of input frames. The overall architecture is depicted in Fig 1. For both problems, it is composed of the same three modules. The first module Flow estimation (blue) which will compute disparity maps for the angular view synthesis problem and optical flows for the frame temporal interpolation problem. The flow estimation module is followed by a pixel-based view synthesis module, PixRNet (orange), and by a feature-based view synthesis module, called FeatRNet (violet). The outputs of these two synthesis modules are then merged using a learned soft fusion mask generated by a network called FusNet (green).

Depending on the targeted problem, the application of the above architecture varies only in terms of input images and in the semantic of the estimated flows. In the case of light field angular view synthesis, the inputs of the architecture are 2×2 light field views. These four inputs, according to their relative positions, are respectively denoted as I_{tl} (top left), I_{tr} (top right), I_{bl} (bottom left) and I_{br} (bottom right) for convenience. In the case of video frame temporal interpolation, the inputs of the architecture are two temporally adjacent frames I_0 and I_1 and the architecture produces temporally intermediate frames. Depending on the targeted problem, the flows estimated by the Flow estimation module are disparity maps in the case of light field angular view synthesis, whereas are they are bi-directional optical flows in the case of temporal frame interpolation.

B. Flow estimation

The Flow estimation module predicts disparity maps for angular view synthesis or bidirectional motions for temporal interpolation.

1) Disparity maps: The notion of 'disparity' refers to the horizontal and vertical displacement of points in the image plane when moving from one viewpoint to the other one [37], [38].

Given four corner views $I_{tl}, I_{tr}, I_{bl}, I_{br}$, we can actually estimate two disparity maps for each view, one is from the horizontal view pair and the other is from the vertical view pair. Taking I_{tl} as an example, two corresponding disparity maps d_1, d_2 are obtained from two image pairs as follows:

$$d_1 = \text{FNet}(I_{tl}, I_{tr})_h, \tag{1}$$

$$d_2 = \mathcal{R}^{-1} \circ \text{FNet}(\mathcal{R}(I_{tl}), \mathcal{R}(I_{bl}))_h, \tag{2}$$

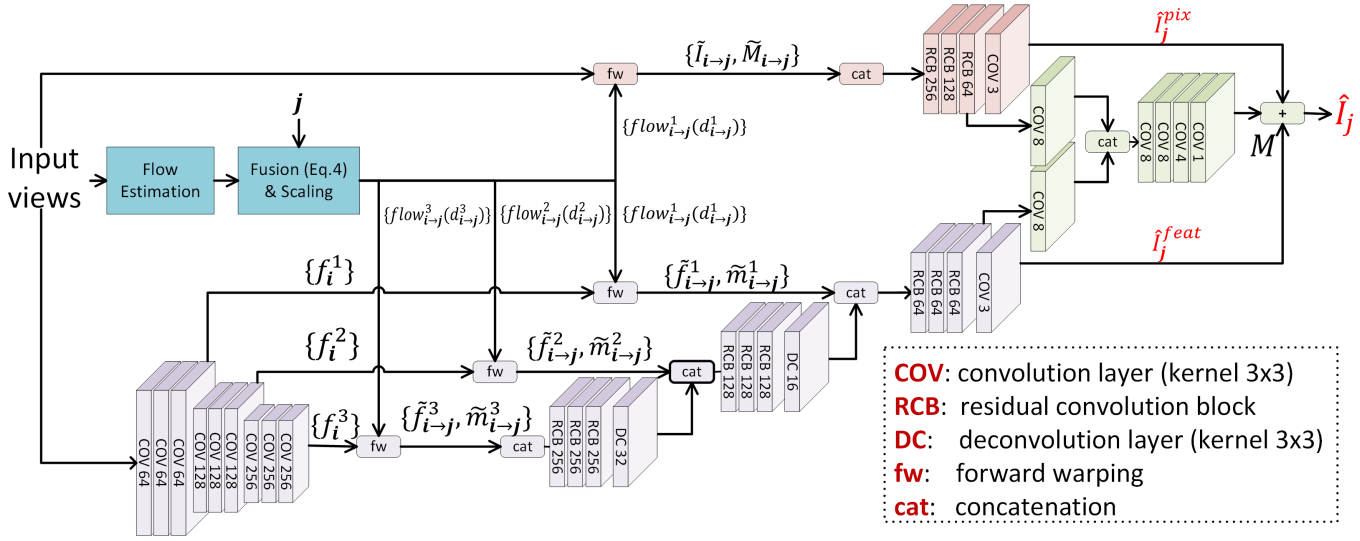


Fig. 1. Overview of the proposed deep architecture for both light field view synthesis and video frame temporal interpolation. In the case of the light field view synthesis problem, the input views are 4 sparse light field views (e.g. corner views) with $\mathbf{i} = \{tl, tr, bl, br\}$ being the index of the source view positions, and $\mathbf{j}$ the index of the target view position. In the case of video frame temporal interpolation, the input views are temporally adjacent views, with $\mathbf{i} = \{0, 1\}$ the index of source frame time instants, and $\mathbf{j} = t$ the target frame instant.

where FNet is a flow-estimation network, the subscript h refers to the horizontal component of the output optical flow. $\mathcal{R}$ and $\mathcal{R}^{-1}$ are respectively counterclockwise and clockwise rotation of $90°$, which permits us to convert vertical disparities to horizontal ones. In our work, we employ a state-of-the-art flow estimation network PWC-Net [33] as FNet, with a novel finetuning strategy that optimizes its weights for the targeted view synthesis or temporal interpolation tasks without using any ground truth optical flows. The training strategy hence differs from the one in [25], where we finetune PWC-Net with ground truth disparity maps between light field image pair, More details will be presented in IV-C2. Let us note that any other flow estimation network can likewise be used in our architecture.

Both d_1 and d_2 may contain estimation errors due to color inconsistencies or occlusions. Inspired by the work in [39], we further improve disparity accuracy by merging two disparity maps d_1 and d_2 into a single one d_{tl}. More specifically, based on each disparity map, other three views I_{tr}, I_{bl} and I_{br} are projected to the position of I_{tl} and the corresponding warping errors e_1 (for d_1) and e_2 (for d_2) are computed as:

$$e_n = \sum_{\mathbf{i} \in \{tr,bl,br\}} \sum_{RGB} (I_{tl} - \tilde{I}^n_{\mathbf{i}\to tl}), n = 1, 2, \tag{3}$$

where $\tilde{I}^n_{\mathbf{i}\to tl}$ represents image warped from current position $\mathbf{i}$ to top left corner using disparity d_n. The fusion of d_1 and d_2 is performed via value selection for each pixel $\mathbf{p}$:

$$n' = \arg\min_n e_n(\mathbf{p}), d_{tl}(\mathbf{p}) = d_{n'}(\mathbf{p}). \tag{4}$$

Similarly, we can obtain disparity maps d_{tr}, d_{bl} and d_{br} for the other three views.

2) Bidirectional optical flows: In the case of temporal interpolation task, the flow estimation module takes two input frames I_0 and I_1 and output a forward flow (from instant 0 to 1) and a backward flow (from instant 1 to 0):

$$flow_{0\to 1} = \text{FNet}(I_0, I_1), \tag{5}$$
$$flow_{1\to 0} = \text{FNet}(I_1, I_0). \tag{6}$$

The estimated optical flows have horizontal and vertical components. Moreover, as there is only one estimated motion for each view, further refinement by fusion is not required. Both flows are used for forward and backward warping when performing the temporal interpolation, as explained later in Section IV-C1.

C. View warping and Fusion

Based on the estimated optical flows or disparity maps, the input views and their feature maps are warped onto the target view position $\mathbf{j}$ via a differentiable forward warping operation while computing at the same time binary occlusion masks. The warped images and feature maps are respectively fed into the PixRNet and FeatRNet synthesis modules along with the occlusion masks to synthesize two views $\hat{I}^{pix}_{\mathbf{j}}$ and $\hat{I}^{feat}_{\mathbf{j}}$. The image $\hat{I}^{pix}_{\mathbf{j}}$ obtained through a pixel-wise synthesis in PixRNet, is accurate in homogeneous regions but tend to be blurred in highly textured regions and at object boundaries. This can be explained by the fact that pixels to be fused are not fully aligned in such regions due to disparity value inconsistencies among the input views. In the FeatRNet module, deep features covering different receptive fields are warped to the desired position and, thanks to a pyramidal reconstruction, the retrieved $\hat{I}^{feat}_{\mathbf{j}}$ is more accurate in textured regions. Finally, the FusNet module learns a soft mask M to merge $\hat{I}^{pix}_{\mathbf{j}}$ and $\hat{I}^{feat}_{\mathbf{j}}$ into an image $\hat{I}_{\mathbf{j}}$ which well reconstructs both homogeneous and textured regions.

1) Image forward warping: Backward warping (BW) has been a widely used operation for various image processing

tasks, e.g. image classification [40], depth image based rendering (DIBR) [14], [16] etc.It consists in projecting pixels from one source view to a target one, using the optical flow associated to the target view, hence requires the optical flow at the target position. We use instead a forward warping (FW) operation, which does not require flow estimation at the target position, and can moreover better handle occlusions via disparity-dependent interpolation. This FW operation proceeds as follows.

Let us assume that we would like to project a pixel $\mathbf{p} = (x_p, y_p)$ from a source view to a target view. In the FW procedure, given a source viewpoint $\mathbf{i}$, a target viewpoint $\mathbf{j}$ and a disparity map $d_\mathbf{i}$ in the case of light field view synthesis, or given optical flow $flow_{0\to1}$ between source frames I_0, I_1 and the target intermediate instant t in the case of video frame interpolation, the non-integer coordinates of the projected pixel $\tilde{\mathbf{p}} = (x_{\tilde{p}}, y_{\tilde{p}})$ can be computed as:

$$\tilde{\mathbf{p}} = \begin{cases} \mathbf{p} + (\mathbf{j} - \mathbf{i}) d_\mathbf{i}(\mathbf{p}), & \text{for LFVS} \\ \mathbf{p} + t \times flow_{0\to1}(\mathbf{p}), & \text{for VFI} \end{cases} \tag{7}$$

where the term $t \times flow_{0\to1}$ approximates the flow $flow_{0\to t}$ between the frame I_0 and the frame at time instant t, which is then used to warp pixels of frame I_0. The frame I_1 is similarly warped using $flow_{1\to t} = (1-t) \times flow_{1\to0}$.

The pixel value $\tilde{I}_{\mathbf{i}\to\mathbf{j}}(\mathbf{q})$ (or $\tilde{I}_{0\to t}(\mathbf{q})$) at integer coordinates $\mathbf{q} = (x_q, y_q)$ of the warped image will be interpolated as a weighted sum of nearby source pixels $I_\mathbf{i}(\mathbf{p})$ (or $I_0(\mathbf{p})$):

$$\widetilde{I}_{\mathbf{i}\to\mathbf{j}}(\mathbf{q}) = \frac{\sum_\mathbf{p} I_\mathbf{i}(\mathbf{p}) W(\mathbf{p}, \mathbf{q})}{\sum_\mathbf{p} W(\mathbf{p}, \mathbf{q}) + \epsilon}, \tag{8}$$

where $W(\mathbf{p}, \mathbf{q})$ is an interpolation weight term, and ϵ is a very small value for numeric stability. Three properties are considered during its design: 1-). this term should be totally differentiable for end-to-end learning. 2-) the further a pixel is from the target position, the smaller weight it will have for interpolation. 3-) pixel overlaps must be handled in the occluded regions. Based on these considerations, this term is finally proposed as the product of a distance term w_D and an overlap handling term w_O:

$$W(\mathbf{p}, \mathbf{q}) = w_D(\mathbf{p}, \mathbf{q}) w_O(\mathbf{p}), \tag{9}$$

where

$$w_D(\mathbf{p}, \mathbf{q}) = l(x_{\widetilde{p}}, x_q) l(y_{\widetilde{p}}, y_q), \tag{10}$$

with

$$l(x_1, x_2) = \begin{cases} (1 - |x_1 - x_2|), & \text{if } |x_1 - x_2| < 1; \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

The term w_O is used for handling overlapping warped pixels, since pixels from foreground and background are often overlaid in occluded regions. In the light field view synthesis problem, we use the disparity maps $d_\mathbf{i}$ of each input view to compute this weight. However, disparity maps are not available for the video frame interpolation problem, and since scene depth is difficult to estimate from a monocular video, we use the warping error for handling overlapping pixels.

More precisely, we use normalized disparity $d^*_\mathbf{i}$ or brightness error e^*_0 to calculate w_O as

$$w_O(\mathbf{p}) = \begin{cases} exp(-\lambda d^*_\mathbf{i}(\mathbf{p})), & \text{for LFVS,} \\ exp(-\lambda e^*_0(\mathbf{p})), & \text{for VFI.} \end{cases} \tag{12}$$

The exponential function gives more importance to pixels having smaller disparity value or brightness error and less importance to those with larger disparity value or brightness error. The brightness error e_0 of I_0 is calculated as

$$e_0 = \sum_{RBG} |I_0 - \mathcal{BW}(I_1, flow_{0\to1})|, \tag{13}$$

where I_0 and I_1 are the two input frames and $\mathcal{BW}$ denotes the backward warping operation.

Non occupied pixel positions on the target view after pixel warping are indicated as disoccluded positions via a binary mask $\tilde{M}$. However, to detect disoccluded pixels in a differentiable manner, and to handle interpolation in disoccluded positions, we also warp the weight map w_O. For each warped position, we thus have RGB and weight values. If there is no pixels in the neighbourhood, then w_D is 0 in all neighbourhood positions, indicating, in a differentiable manner, that the corresponding pixel position is a disoccluded pixel. The warped weights are also used to compute the interpolation weights.

In summary, by taking a source image along with its disparity map or optical flow and brightness error, the FW operation outputs a warped image and a binary disocclusion mask as follows:

$$\tilde{I}_{\mathbf{i}\to\mathbf{j}}, \tilde{M}_{\mathbf{i}\to\mathbf{j}} = \mathcal{FW}(I_\mathbf{i}, d_\mathbf{i}, \mathbf{i}, \mathbf{j}) \tag{14}$$

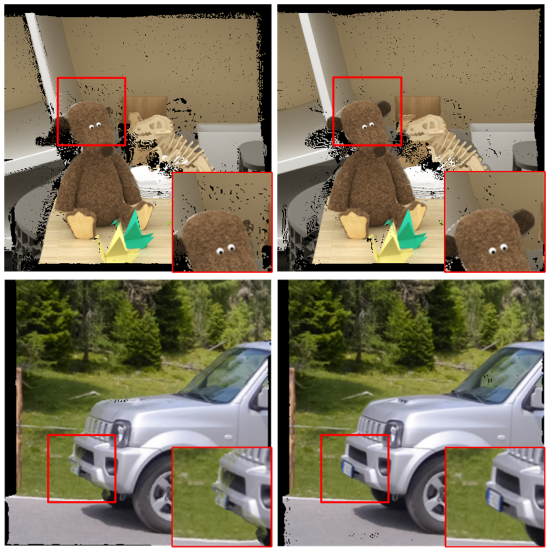
for LFVS, or

$$\tilde{I}_{0\to t}, \tilde{M}_{0\to t} = \mathcal{FW}(I_0, flow_{0\to t}, e_0) \tag{15}$$

for VFI. Both disparity and brightness error are effective cues in handling pixel overlaps, as the larger is the brightness error of a pixel, or the larger is its disparity value, the more likely it will be overlaid after warping. Similar brightness error-based image warping operation, called 'softmax splatting', can be found in [7]. And forward splatting operation is also applied in [41] to infer a layer-structured 3D representation of a scene from a single image.

2) Finetuning the flow estimation module: Depth-dependent LFVS approaches such as in [15], [25] and flow-based VFI schemes such as in [5]–[7], using an off-the-shelf disparity or optical flow estimation module, both rely on a good initialization of that module whether or not end-to-end finetuning is conducted later. However, most of these concerned networks are trained on synthetic datasets with ground truth disparities [37], [42] or optical flows [43], [44], and are not fully adapted to the view synthesis problem. Here, based on our aforementioned FW operation, we propose instead a simple but effective finetuning procedure that enables us to finetune the initial flow estimation module under view synthesis or frame-interpolation supervision, i.e. without any ground truth disparity maps or optical flows.

More specifically, in the case of LFVS, given four input views I_{tl}, I_{tr}, I_{bl}, I_{br} and corresponding disparity maps d_{tl},



(a).Warped view before FT (b).Warped view after FT

Fig. 2. Visual comparison of views before & after finetuning (FT). The views in the 1st row are from the light field dataset in [45], while those in the 2nd row are from the video frame interpolation dataset in [46]

d_{tr}, d_{bl}, d_{br} estimated by Flow estimation module, we aim at synthesizing the view at $\mathbf{j}$. We finetune the estimated flows, without using ground truth flows, by optimizing

$$argmin_\theta \sum_{\mathbf{i}\in\{tl,tr,bl,br\}} |(\tilde{I}_{\mathbf{i}\to\mathbf{j}} - I_{\mathbf{j}})\tilde{M}_{\mathbf{i}\to\mathbf{j}}|_1 \qquad (16)$$

where θ is the set of parameters of the FNet module, and $\tilde{I}_{\mathbf{i}\to\mathbf{j}}$, $\tilde{M}_{\mathbf{i}\to\mathbf{j}}$ are obtained by the FW operation in Eq. 14.

In the case of VFI, given the input frames I_0, I_1 and the associated estimated flows $flow_{0\to1}$, $flow_{1\to0}$, we interpolate a frame at an intermediate instant t, by optimizing

$$argmin_\theta \sum_{t'\in\{0,1\}} |(\tilde{I}_{t'\to t} - I_t)\tilde{M}_{t'\to t}|_1, \qquad (17)$$

where $\tilde{I}_{t'\to t}$, $\tilde{M}_{t'\to t}$ are obtained via Eq. 15, with $flow_{0\to t} = t \times flow_{0\to1}$ and $flow_{1\to t} = (1-t) \times flow_{1\to0}$.

Fig. 2 illustrates the warped images obtained before and after finetuning. Before finetuning, we can observe severe deformations in regions like bear's ear and licence plate that are challenging for the synthesis process. These artifacts are corrected thanks to the finetuning procedure.

3) Pixel and feature-based reconstruction: Thanks to the finetuned Flow estimation module, we can obtain disparity maps or optical flows optimized for synthesis. The synthesis proceeds in parallel in both pixel and deep feature domains.

Pixel-based reconstruction is performed by the PixRNet module, where, in the case of LFVS, using estimated disparity maps and the forward warping operation, four corner views are first projected to the target position $\mathbf{j}$ to generate the warped images $\{\tilde{I}_{tl\to\mathbf{j}}, \tilde{I}_{tr\to\mathbf{j}}, \tilde{I}_{bl\to\mathbf{j}}, \tilde{I}_{br\to\mathbf{j}}\}$ and the corresponding occlusion masks $\{\tilde{M}_{tl\to\mathbf{j}}, \tilde{M}_{tr\to\mathbf{j}}, \tilde{M}_{bl\to\mathbf{j}}, \tilde{M}_{br\to\mathbf{j}}\}$. The warped images and occlusion masks are then concatenated to be fed into the PixRNet module. Similarly, in the case of VFI, warped frames $\{\tilde{I}_{0\to t}, \tilde{I}_{1\to t}\}$ and masks $\{\tilde{M}_{0\to t}, \tilde{M}_{1\to t}\}$ are instead concatenated and fed into the PixRNet module to learn the reconstructed $\hat{I}_t^{pix}$.

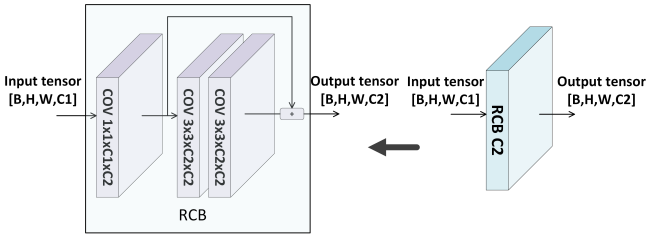


Fig. 3. Architecture of Residual Convolutional Block (RCB) with input tensor of dimension $[B, H, W, C1]$ and output tensor of dimension $[B, H, W, C2]$.

Disoccluded pixels are clearly indicated by the occlusion masks, which hence makes it easier for PixRNet to inpaint these disoccluded regions while fusing different views into the target view $\hat{I}_{\mathbf{j}}^{pix}$. Different from the LFVS approach in [25] that cascades four simple convolutional layers, we replace the first three layers with novel residual convolutional block (RCB) architecture illustrated in Fig 3. A RCB is composed of three convolutional layers, where the first layers with kernel size 1×1 convert $C1$ input channels to $C2$ channels. Then two cascaded convolutional layers with kernel size 3×3 learn a residual map acting on the input. Moreover, in comparison with the original design in [25], we quadruple the number of channels in RCB to further enhance its learning ability. The usage of RCB improves the reconstruction efficiency. More details will be investigated in Sec. VI-C.

Due to inconsistency between disparities or optical flows of input views, those warped views or frames are not fully aligned and the fusion may yield blurriness especially in highly textured regions and object contours. To overcome this limitation, we propose using an additional feature-based reconstruction module named FeatRNet, in which a trainable encoder is employed to flexibly extract deep features at three different resolutions later used in a pyramidal reconstruction step.

In comparison with the VGG19 encoder with fixed weights used in [25], our trainable encoder is 1-) more lightweight than the VGG19 encoder (1.9M parameters vs 2.3M parameters). 2-) more flexible to extract features suitable for synthesis. The feature maps of input views are extracted at three different resolutions as:

$$\{f_{\mathbf{i}}^1, f_{\mathbf{i}}^2, f_{\mathbf{i}}^3\} = FeatExt(I_{\mathbf{i}}), \forall \mathbf{i} \in \{tl, tr, bl, br\} \qquad (18)$$

where $FeatExt$ denotes the feature extraction operation and the extracted feature maps $f_{\mathbf{i}}^{s+1}$ are at half resolution compared with $f_{\mathbf{i}}^s$. Like in the PixRNet module, these feature maps are then warped to the desired position $\mathbf{j}$ using forward warping as

$$\tilde{f}_{\mathbf{i}\to\mathbf{j}}^s, \tilde{m}_{\mathbf{i}\to\mathbf{j}}^s = \mathcal{FW}(f_{\mathbf{i}}^s, d_{\mathbf{i}}, \mathbf{i}, \mathbf{j}). \qquad (19)$$

Warped features and masks at lower levels are first concatenated and fused by RCBs, and then $2\times$ upsampled by a deconvolutional layer to be fed to the next level. Finally, at the highest level, the warped feature maps and the occlusion masks along with upsampled features are merged to get the desired view $\hat{I}_{\mathbf{j}}^{feat}$. In case of VFI, the FeatRNet module will

take two frames as inputs and follow the same reconstruction process to obtain $\hat{I}_t^{feat}$.

Let us note that in [25], feature maps of the ground truth image are extracted using a fixed VGG19 encoder to supervise the feature reconstruction at each level of the FeatRNet module. With a trainable encoder, the extracted feature maps of the ground truth image vary with the learned encoder weights, and supervising the feature reconstruction with varying ground truth feature maps results in model performance oscillations during training. We therefore no longer supervise feature reconstruction and directly focus on the final reconstruction of $\hat{I}_{\mathbf{j}}^{feat}$ (or $\hat{I}_t^{feat}$) in the FeatRNet module. That also helps simplifying the training schedule as detailed later in Sec. V-C.

As the reconstructed views $\hat{I}_{\mathbf{j}}^{pix}$ and $\hat{I}_{\mathbf{j}}^{feat}$ (or $\hat{I}_t^{pix}$ and $\hat{I}_t^{feat}$) have complementary properties in homogeneous and highly textured regions, we want the synthesis to benefit from the advantages of both. A FusNet module is therefore employed to merge two reconstructed views into the final target view $\hat{I}_{\mathbf{j}}$ (or $\hat{I}_t$) with a soft mask M having value between 0 and 1 (forced by sigmoid activation) as follows

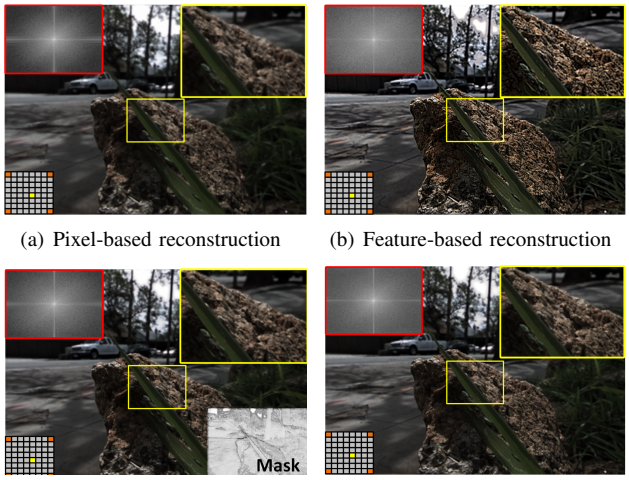
$$\hat{I}_{\mathbf{j}} = M\hat{I}_{\mathbf{j}}^{pix} + (1-M)\hat{I}_{\mathbf{j}}^{feat}. \tag{20}$$

To train the entire architecture, we propose a loss function composed of three reconstructed views as follows

$$\mathcal{L} = \gamma_1\mathcal{L}_l(\hat{I}_{\mathbf{j}}^{pix}, I_{\mathbf{j}}) + \gamma_2\mathcal{L}_l(\hat{I}_{\mathbf{j}}^{feat}, I_{\mathbf{j}}) + \gamma_3\mathcal{L}_l(\hat{I}_{\mathbf{j}}, I_{\mathbf{j}}), \tag{21}$$

where $\mathcal{L}_l$ represents a Laplacian loss [47] with 3 levels. $\{\gamma_1, \gamma_2, \gamma_3\}$ are weight hyperparameters.

Fig 4 shows the synthesized central views $\hat{I}_{5,5}^{pix}$, $\hat{I}_{5,5}^{feat}$, $\hat{I}_{5,5}$ and their Fourier transforms in the case of LFVS. And we can observe from Fig 4(a) and Fig 4(b) that pixel and feature-based reconstructions have complementary properties: the pixel-based reconstruction is more blurred and the energy in the Fourier domain is mainly concentrated in low frequencies, while the feature-based reconstruction is better in textured regions, and most of its energy resides in high frequencies. After fusing two views with a mask, we can see in Fig 4(c) that the final reconstructed view inherits the advantages of both pixel and feature-based reconstruction, and is close to the ground truth view in Fig 4(d) in both the color and Fourier domains.



(a) Pixel-based reconstruction (b) Feature-based reconstruction

(c) Fused view (d) Ground truth

Fig. 4. Visual comparison in both pixel and frequency domains between $\hat{I}_{5,5}^{pix}$, $\hat{I}_{5,5}^{feat}$,$\hat{I}_{5,5}$ and $I_{5,5}$ in the case of LFVS.

V. Implementation Details

A. Training data preparation

In the case of LFVS, we employ 78 synthetic light fields proposed in [45] and 16 light fields in 'Additional' category of [48] to form our synthetic training set. All light fields are of resolution $512 \times 512 \times 9 \times 9$. We also use 100 scenes proposed in [14], which are captured by a Lytro Illum camera with resolution $376 \times 541 \times 14 \times 14$, by extracting the 8×8 central views of each scene to form our real-world training set. Since real-world data contains complicated lighting conditions and artifacts caused by hardware limitations, making it distinct from synthetic data, we therefore prepare two models, one trained on synthetic data and the other further finetuned on real-world data. The finetuning of FNet and the training of the whole pipeline are conducted using the same dataset. As light fields from DLFD and SLFD in [45] and those in [14] captured by a Lytro Illum camera, are with different baselines, we therefore adopt different configurations when extracting views: for light fields in DLFD that have narrow baseline, four corner views are selected by an *angular distance* corresponding to view index difference $l \in \{5, 6, 7, 8\}$. For light fields in SLFD having wide baseline, corner views are extracted with $l \in \{2, 3\}$. When using the real-world training set,the corresponding angular distance is set to be $l \in \{6, 7\}$. The ground truth view is randomly selected within the square defined by corner views. We use randomly cropped patches of size 160×160 and a batch size of 8 in the training step.

For VFI, we adopt the widely used Vimeo90K dataset in [49] for training. This training set contains 51,313 video triplets with resolution 256×448. We use the intermediate frame as ground truth and other two frames as inputs, and randomly crop patches in size 160×160 during training, the batch size is set to 18.

B. Data augmentation

We also apply geometrical and chromatic transformations to increase the variety of the training data. The chromatic transformation is carried out by changing the contrast, color balance, brightness and gamma value. The contrast is sampled within the interval $[-0.8, 0.4]$, the color balance is adjusted with a multiplicative factor sampled within $[0.5, 2]$, the brightness is modified by adding a value drawn according to a Gaussian distribution with $\sigma = 0.2$, and the gamma value is sampled within the interval $[0.7, 1.5]$. For the geometrical transform, we randomly rotate the image by $0°$, $90°$, $180°$ or $270°$ counterclockwise and flip it on the left and right.

C. Training schedule

The training schedule of the model can be divided into an initialization step followed by an end-to-end optimization step. In the initialization step, we fix the weights in the FNet

module and set $\gamma_1 = \gamma_2 = \gamma_3 = 1$ in the loss function. The training will enable both PixRNet, FeatRNet and FusNet to generate views that approximate the ground truth view. We set the learning rate to 0.0001 for this step, and the whole pipeline is trained for 3000 epoches in the case of VSLF and for 30 epoches in the case of VFI.

Then, by setting $\gamma_1 = \gamma_2 = 0$ and $\gamma_3 = 1$ in the loss function, we end-to-end train the framework to optimize the final synthesized view, which makes PixRNet and FeatRNet to retrieve views with complementary properties. The learning rate in this stage is set to 0.00001 and we stop the training when there is no performance amelioration within 2000 epoches for LFVS or 20 epoches for VFI. The training takes about 4 days for LFVS and 5 days for VFI on a Nvidia Tesla V100 GPU with 32GB GRAM, the pipeline is implemented using the *tensorflow* framework. Our method has about 18.2M parameters (where 9.3M parameters are from PWC-Net), and it takes about 0.9s to synthesize an image of size $512 \times 512 \times 3$ (36.4M FLOPS).

VI. Experimental results

In this section, we evaluate the effectiveness of our proposed method for both LFVS and VFI tasks using various datasets.

A. Light field view synthesis

1) Synthetic LFs: We first validate the effectiveness of the proposed architecture using public synthetic light field datasets:

- densely sampled light fields *stilllife, buddha, butterfly, monasRoom* with resolutions $768 \times 768 \times 9 \times 9$ from the old HCI dataset [50], and *boxes, cotton, dino, sideboard* with resolutions $512 \times 512 \times 9 \times 9$ from new HCI dataset [48],
- sparsely sampled light fields *Toy_bricks, Elec_devices, Two_vases, Sculpture, Bear* with resolutions $512 \times 512 \times 9 \times 9$ from the INRIA_synth dataset [45]

We focus on the central 7×7 and 3×3 angular views in densely and sparsely sampled light fields respectively, synthesizing all views, given the four corner views.

Our approach is compared with state-of-the-art approaches, e.g. learning-based DIBR pipeline [14] (DeepVS), EPI-based view synthesis [17] (EPI), synthesis using multi-layer scene representation with learning [16] (LLFF) and without learning [15] (Soft3D), and with our previous pixel-feature based reconstruction scheme [25] (FPFR). During the test, all involved methods take four views as inputs, except LLFF, which take an extra fifth view as its released model requires at least five input views. We thus employ the horizontal immediate neighbour of I_{tl} as the fifth view. Furthermore, we directly use ground truth camera pose instead of estimated ones in LLFF to make sure the reconstruction quality only depends on the synthesis process. Given that training data plays a critical role in determining the performance of learning-based methods, we further finetuned pre-trained models of compared methods using the same training data as for our method for having a fair comparison.

Table I gives a quantitative evaluation of the synthesized central view in terms of PSNR, with all tested light fields arranged from dense to sparse according to the disparity range. In the last two columns, we show the performance of the vanilla version (Ours) and test-time augmented version (Ours*) of our method. The test-time augmentation is achieved by averaging eight reconstructed views, whose corresponding input views are rotated and flipped, after inverse rotation and flip processing. We can observe from the table that our novel pipeline outperforms the state-of-the-art methods with most scenes, and in average gives about 0.8dB gain against the best reference method. Besides testing on the central view that is the most distant from corner views, we further trace the PSNR curves when varying the target position for the different methods, Fig 6(a) shows the synthesis quality evolution in terms of viewpoints. We can observe that our method outperforms reference ones with a very large margin. Furthermore, in comparison with the FPFR network, our novel architecture gains in average 1dB for views distant from the corner views, and even more for views that are near the source input views.

2) Real-world LFs: We have also tested our method using real-world scenes from the testset of [14]. We feed four views to the tested methods except for LLFF, which takes a fifth view. As the ground truth camera pose is unavailable for real-world data, we hence convert the camera pose-based view transform to a disparity-based transform when constructing the PSV in LLFF. Table II shows the quantitative measures obtained for the central reconstructed views. A gain of 0.5dB is observed against FPFR, the best reference method.

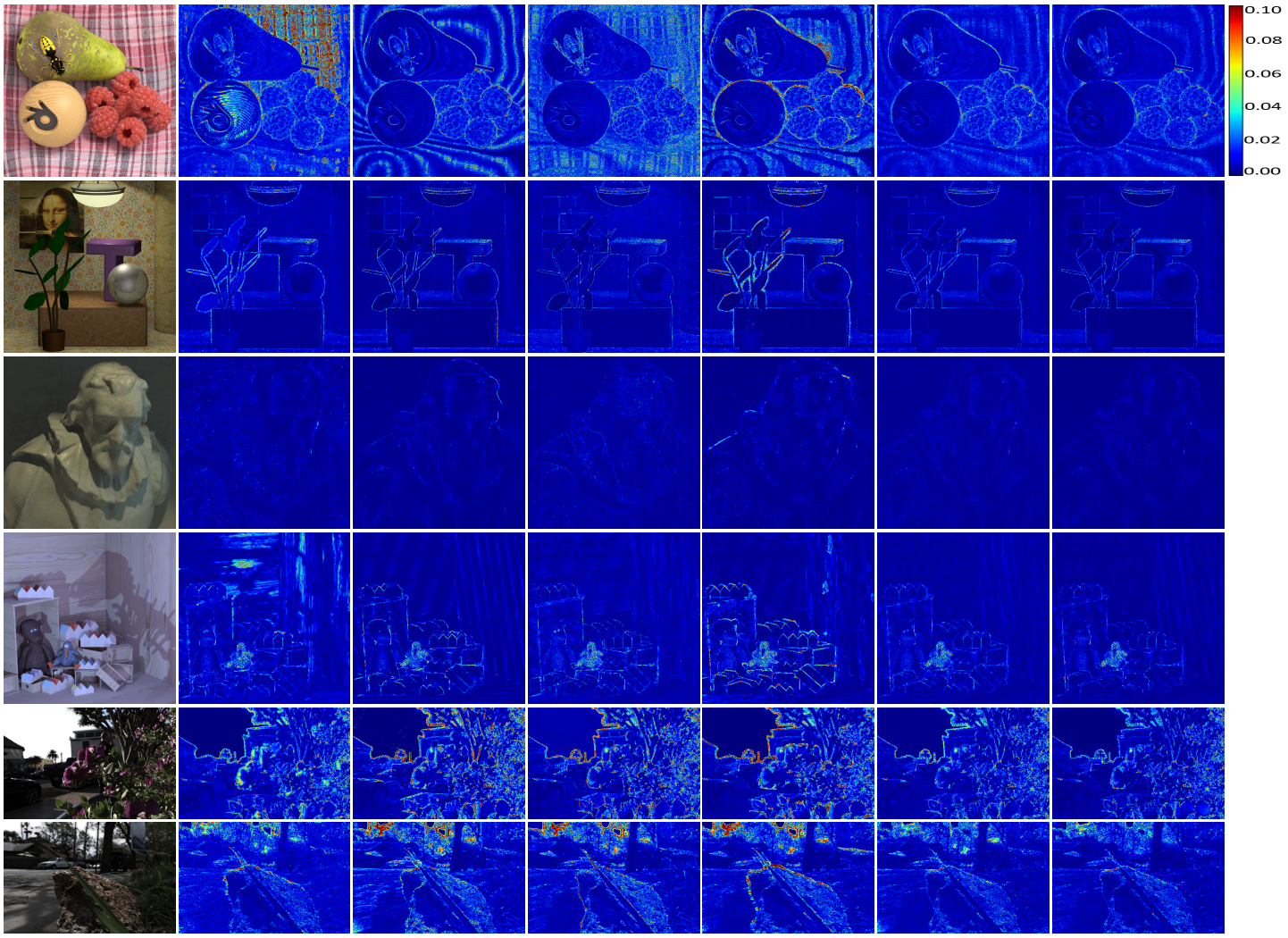
Fig. 5 shows reconstruction error maps for both synthetic and real-world light fields, we can observe that our method reconstructs views with less errors, especially on the object contours and subtle structures in highly textured regions. More experimental results are available in our project homepage: http://clim.inria.fr/research/TCI-VS-VI/index.html.

3) View extrapolation in LFs: Extrapolating views beyond the original field of view of the light field is a very challenging task, as less information on the target view is available in the input views. We evaluate the extrapolation capability of our method in comparison with two reference methods, namely the FDL approach in [51], the LLFF method [16] and our previous pipeline FPFR [25]. All tested methods take four corner views of the central 3×3 subset of light field views (red slashes in Fig. 6(b)) as inputs, except for the LLFF method which additionally takes the central view as a fifth input view (grey slashes in Fig. 6(b)), to synthesize a 9×9 light field. Fig. 6(c) shows the performances for each method. We can observe that our method yields better results than the reference methods. Wider is the baseline, more important are the gains with our approach. Let us note that all learning-based methods used in this test have been trained for the interpolation task, hence here we assess the inherent extrapolation capability of each method without any further finetuning.

4) Comparative evaluation with Neural Radiance Fields: In recent years, view synthesis methods have been proposed based on the concept of Neural Radiance Fields (NeRF). NeRFs represent a mapping between 5D spatial $((x, y, z)$

TABLE I
QUANTITATIVE RESULTS (PSNR) FOR THE RECONSTRUCTED CENTRAL VIEW OF THE SYNTHETIC TEST LIGHT FIELDS. THE CORRESPONDING DATASETS ARE INDICATED BY SYMBOLS: ⋆ [48], ◇ [45] AND † [50]. THE BEST AND THE SECOND BEST PERFORMANCES ARE SHOWN IN BOLD AND UNDERLINED.

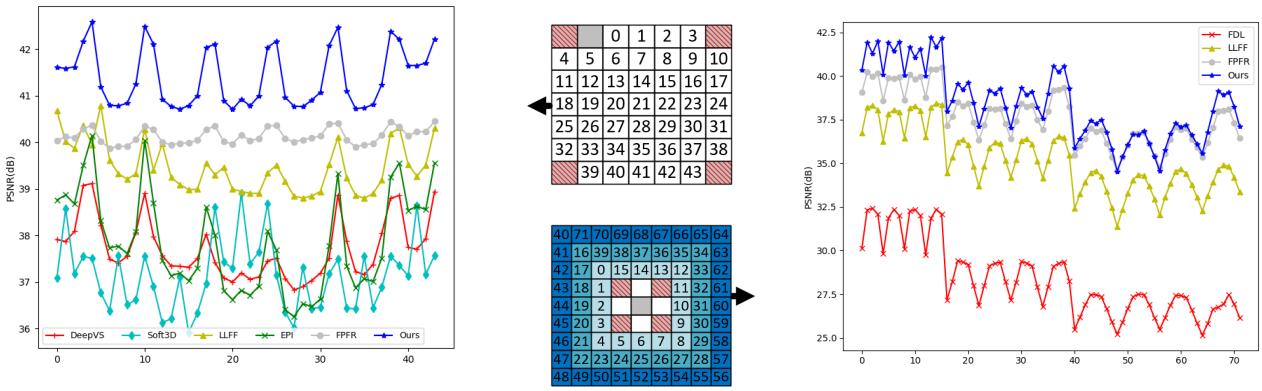
LFs	Disparity range	DeepVS [14]	Soft3D [15]	LLFF [16]	EPI [17]	FPFR [25]	Ours	Ours*
mona†	[-5,5] (10)	38.90	40.92	41.20	37.54	42.47	<u>43.26</u>	**43.57**
butterfly†	[-6,8] (14)	40.68	42.75	41.35	39.61	42.69	<u>43.52</u>	**43.79**
buddha†	[-10,6] (16)	41.08	41.86	40.66	40.05	42.78	<u>43.50</u>	**43.72**
cotton⋆	[-9,9] (18)	47.24	48.95	47.07	47.97	48.58	<u>50.48</u>	**50.59**
boxes⋆	[-7,13] (20)	33.64	32.14	**34.97**	31.65	33.86	33.41	<u>34.29</u>
dino⋆	[-10,10] (20)	38.41	41.69	41.26	38.44	42.66	<u>43.69</u>	**44.09**
sideboard⋆	[-10,12] (22)	30.91	30.23	32.33	27.30	31.85	<u>32.37</u>	**32.75**
Toy_bricks◇	[-1,22] (23)	28.90	36.58	37.98	31.46	38.84	<u>40.44</u>	**40.89**
Elec_devices◇	[-10,17] (27)	34.09	36.24	36.76	31.55	37.63	<u>39.25</u>	**39.50**
stilllife†	[-16,16] (32)	26.29	34.73	32.73	32.02	36.39	<u>37.20</u>	**37.74**
Lion◇	[-5,29] (34)	28.05	35.18	35.22	33.91	35.47	<u>35.66</u>	**35.75**
Two_vases◇	[-5,39] (44)	25.65	32.49	<u>35.82</u>	29.09	35.56	35.54	**36.26**
Sculpture◇	[-26,34] (60)	22.31	29.15	29.68	26.22	<u>30.09</u>	30.02	**30.12**
Bear◇	[-38,53] (91)	18.36	28.00	33.22	23.40	31.87	<u>33.41</u>	**34.22**
Average	-	32.49	36.49	36.43	33.59	37.92	<u>38.70</u>	**39.09**



Light field view DeepVS [14] Soft3D [15] LLFF [16] EPI [17] FPFR [25] Ours

Fig. 5. Reconstruction error maps of view $\hat{I}_{5,5}$ for different synthesis methods, with higher error value in red and lower error value in blue.

and angular (θ, ϕ)) coordinates of light rays, and their color (RGB) components and a volume density measure (σ). New views can be rendered from the NeRF model using volume rendering techniques. The model is optimized per scene, and



(a) PSNR for **interpolated** viewpoints (b) View indices (c) PSNR for **extrapolated** viewpoints

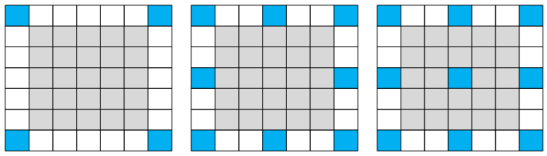
Fig. 6. Averaged PSNR curves for the different viewpoints. The 8 synthetic scenes are from the datasets [48], [50]. (a) Interpolation. (c) Extrapolation. (b) View indices for interpolation (top) and extrapolation (bottom). 4 input views (red slash) are used for DeepVS [14], EPI [17], Soft3D [15], FPFR [25], FDL [51] and our method, whereas 5 input views (grey) are used for LLFF [16].

TABLE II
QUANTITATIVE RESULTS (PSNR) FOR THE RECONSTRUCTED VIEW (5,5) WITH REAL-WORLD DATA (8×8 VIEWS) [14]. THE BEST PERFORMANCES ARE SHOWN IN BOLD AND THE SECOND BEST ONES ARE UNDERLINED.

LFs	DeepVS	Soft3D	LLFF	EPI	FPFR	Ours
Cars	31.53	27.68	29.06	28.17	32.05	**32.86**
Flower1	33.13	30.29	30.00	30.44	34.19	**34.63**
Flower2	31.95	30.52	28.90	29.26	33.80	**34.12**
Rock	34.32	32.67	32.60	32.46	36.48	**37.13**
Leaves	27.97	27.34	27.74	26.48	32.27	**32.65**
Seahorse	32.03	30.41	28.50	26.62	34.68	**34.95**
Average	31.82	29.82	29.47	28.90	33.91	**34.39**

the network weights can be seen as a representation of the scene radiance. More views are used to train the network (a multi-layer perceptron), more precise and complete is the Radiance Field, and better are the rendering results.

We therefore performed a comparative evaluation with three types of input view configurations: 1) four corner views (Fig. 7(a)). 2) eight border views (Fig. 7(b)). 3) nine views (Fig. 7(c)). The performance measures are averaged on all the other light field views that are synthesized. Table III shows the measured PSNR on 4 LFs from the benchmark considered in [48]. The quality of the views rendered from the learned NeRF model increases with the number of input views. However, when taking only the four corner views, and the corresponding ground truth camera poses, the quality of the views rendered using NeRF is inferior to the views synthesized with our method, except for one light field. Set aside the per-scene training complexity, this shows the limit of the approach when the number of input views is limited. The advantage of our approach in that case is particularly interesting in a compression context, since it leads to higher compression efficiency for the light field, as we show in the next section. The per-scene training complexity of NeRF has later been addressed in the state-of-the-art work of MVSNeRF [23] by proposing a method in which the trained network can be applied to novel scenes. However, as shown in Table III (column *'MVSNeRF-4'*), its generalizability is quite limited: synthesized views have low PSNR without per-scene finetuning. Despite the fact that



(a) 4 input views (b) 8 input views (c) 9 input views

Fig. 7. Three types of input view configurations for NeRF [21], blue squares are input views for training and gray squares are views to be inferred.

TABLE III
QUANTITATIVE EVALUATIONS (PSNR) FOR THE SYNTHESIZED VIEWS. METHOD NERF [21] IS TRAINED WITH DIFFERENT NUMBERS OF VIEWS (4,8,9, SHOWN IN FIG. 7). WE SHOW IN THE PARENTHESES THE PSNR AFTER SCENE-WISE FINETUNING FOR OUR METHOD AND MVSNERF [23]

LFs	Ours-4	MVSNeRF-4	NeRF-4	NeRF-8	NeRF-9
boxes	33.89 (38.84)	31.10 (35.32)	36.88	39.22	39.30
cotton	50.39 (50.84)	38.14 (47.94)	46.67	47.51	47.74
dino	44.04 (45.56)	26.94 (42.95)	40.29	43.63	43.83
sideboard	32.49 (35.38)	22.52 (34.25)	32.15	36.19	36.88
Average	40.20 (42.66)	29.68 (40.12)	38.99	41.64	41.93

per-scene finetuning can further improve the performance of MVSNeRF (PSNR in the parentheses), our method likewise achieves better PSNR after the same finetuning, which proves the superiority of our method against MVSNeRF.

5) Compression performance analysis: We consider a compression set-up in which the reference views, according to the configurations of Figure 7(a), are first encoded using the HEVC-inter coding standard (HM 16.10). The LFVS methods are then used for recovering the entire light field from the compressed/decompressed reference views. This LFVS-based compression pipeline is summarized in Fig. 8, and we use FPFR [25], NeRF [21], MVSNeRF [23] and our scheme as the LFVS method. Besides the LFVS-based pipeline, we also apply deep learning-based video sequence compression methods (HLVC [52], RLVC [53]) and the classical HEVC inter coding method to compress light fields. Fig. 9 shows the obtained PSNR-rate curves with the different LF compression strategies. Our method achieves better compression efficiency than other referenced methods, which means that our method

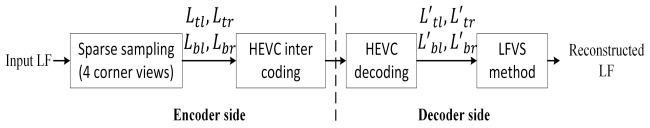


Fig. 8. LF compression pipeline based on LFVS method. We use FPFR [25], NeRF [21], MVSNeRF [23] and our proposed method as LFVS method in the pipeline.

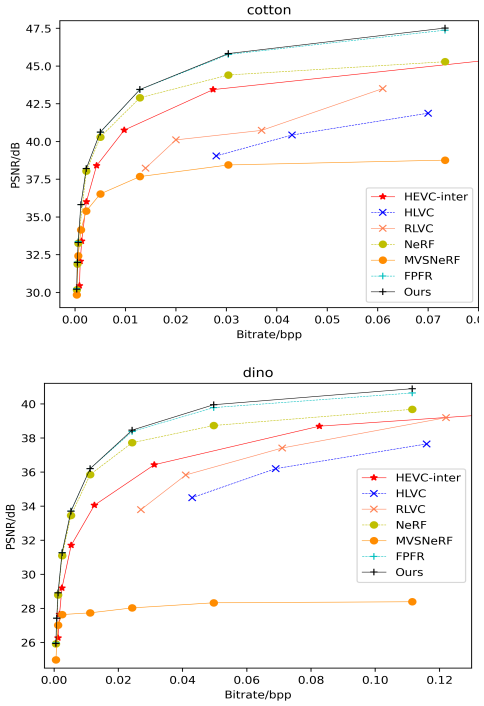


Fig. 9. Rate-distortion curves of scenes 'cotton' and 'dino' using different LF compression methods. Both methods FPFR [25], MVSNeRF [23] and ours are tested without per-scene finetuning, while NeRF [21] is trained for the test scene.

is more suitable for compression applications. Let us note that, although radiance field-based methods NeRF and MVSNeRF work well for view synthesis task itself, they either work poorly without per-scene finetuning (MVSNeRF) or require a long-time training (NeRF) for the compression task, which makes them less practical for the compression task.

B. Video Frame Interpolation

Besides angular view synthesis in light field, we also assess our pipeline for video frame temporal interpolation, using two datasets:

- The Vimeo90K datatset [49] that contains 3,782 video triplets with resolution 256×448, covering a large variety of scenes and actions. We interpolate the intermediate frames using the two adjacent frames as inputs.
- The Adobe240fps dataset [54] that is a real-world video set captured with a handheld camera, it is more challenging than Vimeo90K as it contains more motion blur. We extracted 777 frame triplets (about 10% of the total number of frames) with resolution 360×640 and synthesized the intermediate frames.

We compare the quality of the interpolated frames with the one obtained with state-of-the-art VFI methods, i.e. with MEMC-Net [11], which merges kernel and flow-based synthesis in its pipeline; FeFlow [8], which exploits a flow of features instead of pixels for motion estimation; and Softmax Splatting (SMSP) [7], a flow-based framework that employs pixels and features in the synthesis. For the MEMC-Net and FeFlow methods, We use the official authors source code and their pre-trained model. However, as the code and model are not available for SMSP, we re-implemented this method and trained it using the Vimeo90K training set following the instructions in the paper. We could not reproduce the results of the paper, therefore, we give in Table IV both results, the ones taken from the paper [7] (noted as SMSP*), and those obtained with our implementation.

TABLE IV
QUANTITATIVE EVALUATIONS (PSNR&SSIM) FOR THE SYNTHESIZED INTERMEDIATE FRAMES. THE BEST PERFORMANCES ARE SHOWN IN BOLD AND THE SECOND BEST ONES ARE UNDERLINED.

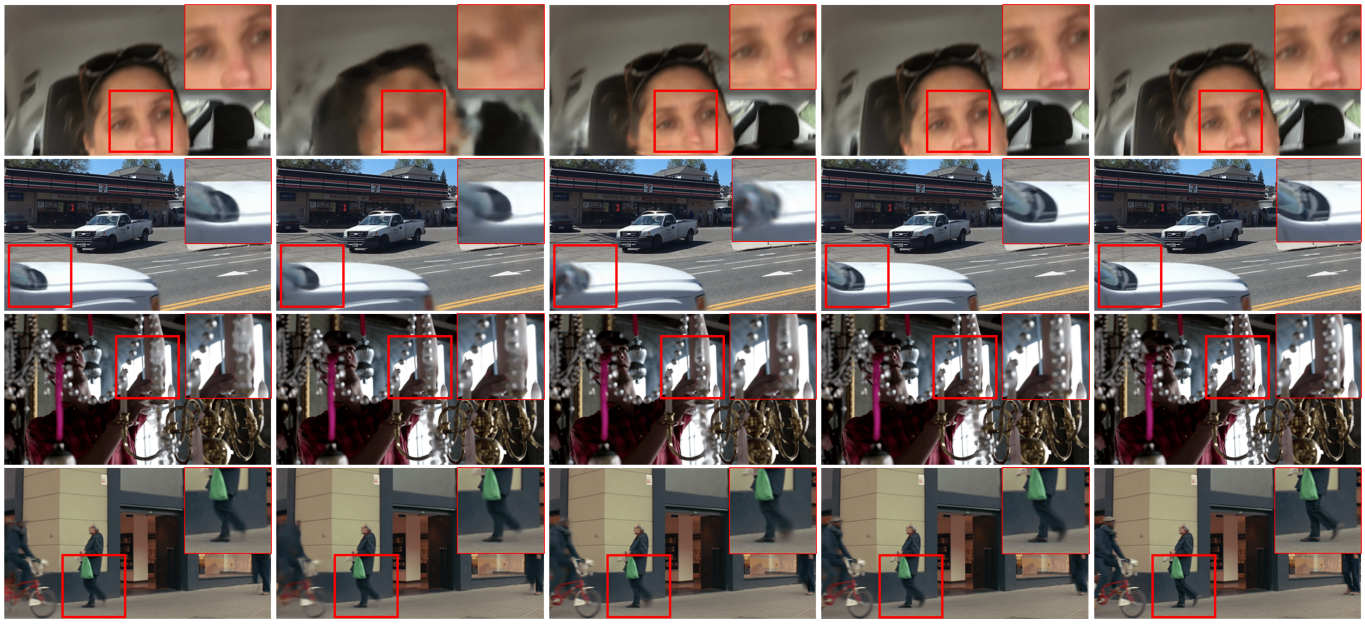
Methods	Vimeo90K [49]		Adobe240fps [54]	
	PSNR	SSIM	PSNR	SSIM
SuperSloMo [4]	30.92	0.932	28.44	0.897
SepConv [10]	33.80	0.956	31.16	0.923
MEMC-Net [11]	34.43	0.963	31.54	<u>0.927</u>
FeFlow [8]	35.09	0.963	31.50	0.925
SMSP [7]	35.51	<u>0.967</u>	<u>31.60</u>	<u>0.927</u>
*SMSP** [7]	**36.10**	**0.970**	-	-
Ours	<u>35.96</u>	**0.970**	**31.72**	**0.928**

Table IV gives the PSNR and SSIM performances for all tested methods, we can notice that our proposed architecture, although originally designed for angular view synthesis in light field, is well suited for temporal video frame interpolation. It can generate intermediate frames of a quality that is comparable to the one of the top-rank official SMSP method. The table shows that the proposed method outperforms state-of-the-art methods by a large margin. Fig 10 shows interpolated frames obtained with different methods. Compared with other methods, our architecture is able to retrieve high quality frames with subtle details. Let us note that when the input frames contain some blur, like in the 1st row, the reference methods may suffer from deformation or may fail to synthesize intermediate views, while our method is more robust and can still generate plausible results. Let us note that the proposed VFI method can be used to compress a video. The approach can indeed be applied in a similar way as the inter-prediction method based on deep frame interpolation networks in [55]. For each frame that can be coded with bi-directional inter-prediction from past and future frames, an additional reference frame can be provided by interpolating the original reference frames.

C. Ablation study

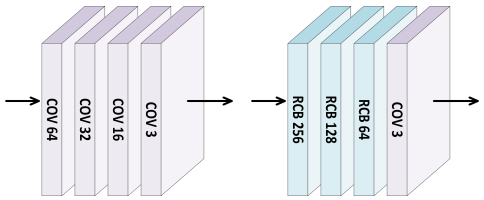
In this section, we carry out an ablation study to analyze the performance gain brought by the new architecture when compared with [25].

1) Residual convolutional block (RCB): We assess the performance gain obtained by using the RCB instead of simple convolutional layers. We consider the PixRNet module and fix the weights in the FNet module so that the quality of the synthesized views only depends on the synthesis block. We compare the three variants shown in Fig 11, where Fig 11(a)

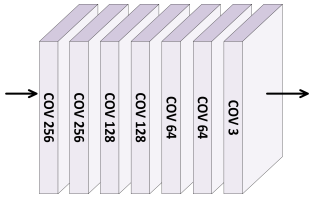


(a) MEMC-Net [11] (b) FeFlow [8] (c) SMSP [7] Ours GT

Fig. 10. Video frame interpolation results in comparison with (a) MEMC-Net [11], (b) FeFlow [8], (c) SMSP [7]. The upper two rows use frames from [54], the lower two rows use frames from [49]. We use official outputs of SMSP for the last two rows.



(a) PixRNet in [25] (b) PixRNet using RCBs



(c) PixRNet using CBs

Fig. 11. Three variants for PixRNet, with (a) fundamental architecture used in [25], (b) PixRNet based on RCB, (c) PixRNet based on CB (without residual connection).

shows the fundamental architecture adopted in [25], Fig 11(c) an enhanced architecture with more kernels and layers, which can also be considered as the non-residual connection version of Fig 11(b), and where Fig 11(b) is our current PixRNet architecture based on RCB. Their corresponding learning curves are compared in Fig 12(a), with all curves have been obtained using four HCI light field scenes in [48]. One can notice that adding kernels and layers like in Fig. 11(c) allows improving the network performance. Residual connections further ameliorate the synthesis quality.

2) Trainable encoder: Another improvement in our pipeline is the use of a trainable encoder, instead of a fixed VGG19 encoder. A lightweight trainable encoder that flexibly extracts feature maps in cooperation with the synthesis step effectively accelerate the network convergence. By fixing weights in the FNet module, we trace the learning curves of the FeatRNet module using two different encoders (see Fig. 12(b)). We can note that with a trainable encoder, the FeatRNet module quickly converges to a high PSNR value, which means that the extracted features are suitable for synthesis. While the features extracted with the VGG19 encoder originally designed for the classification task are less suitable for the reconstruction task, and yields slower network convergence.

3) Global architecture: Finally, Fig. 12(c) shows that the training of the proposed architecture, when compared with FPFR, converges faster. Tables I and II show that this novel architecture is quite efficient for both light field view synthesis and video frame temporal interpolation.

VII. Conclusion

In this paper, we have presented a learning-based architecture for both angular view synthesis for light field and video frame temporal interpolation. The proposed scheme employs a flow estimation network to predict flows between input images. The synthesis is then conducted in parallel in pixel-based and feature-based branches followed by a soft mask fusion to obtain the synthesized view. Compared with the FPFR solution originally designed for light field view synthesis, the novel architecture adopts residual convolutional blocks (RCBs) and a lighter weight trainable encoder that allows improving both training efficiency and synthesis performance for light field view synthesis. The architecture is further generalized to handle video frame temporal interpolation. The effectiveness of the proposed solution is demonstrated by comparing with state-of-the-art methods using both light field synthesis and video interpolation datasets. Experimental results show that our method tackles well both tasks with high quality results.

References

[1] B. Wilburn, N. Joshi, V. Vaish, E. Talvala, E. Antunez, A. Barth, A. Adams, M. Horowitz, and M. Levoy, "High performance imaging

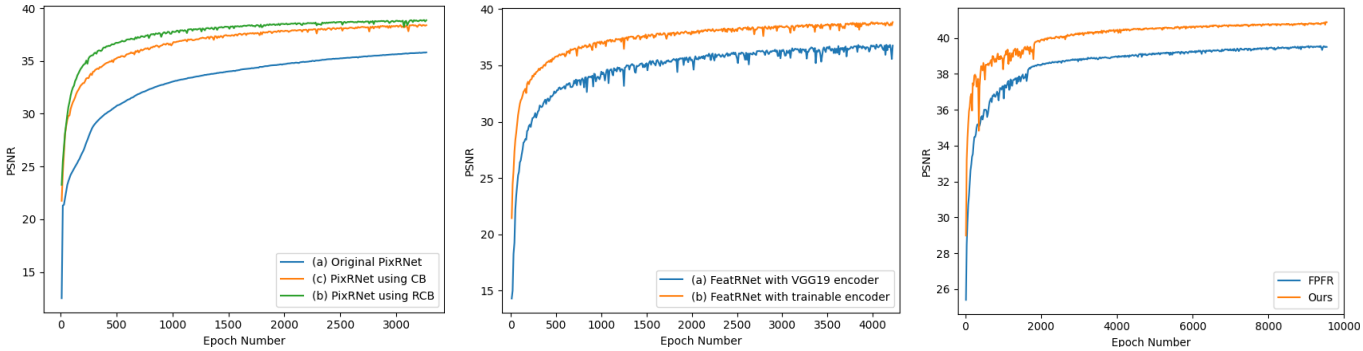


Fig. 12. Learning curves of (left) different architectures (PixRNet in [25], PixRNet using RCB, PixRNet using CB); (middle) different encoders (FeatRNet using VGG19 encoder, and FeatRNet using a trainable encoder); (right) FPFR and our new architecture

using large camera arrays," *ACM Trans. on Graphics (TOG)*, vol. 24, no. 3, pp. 765–776, Jul. 2005.

[2] R. Ng, M. Levoy, M. Brédif, G. Duval, M. Horowitz, and P. Hanrahan, "Light field photography with a hand-held plenoptic camera," *Computer Science Technical Report (CSTR)*, vol. 2, no. 11, pp. 1–11, 2005.

[3] Z. Liu, R. Yeh, X. Tang, Y. Liu, and A. Agarwala, "Video frame synthesis using deep voxel flow," in *IEEE Int. Conf. on Computer Vision (ICCV)*, 2017, pp. 4463–4471.

[4] H. Jiang, D. Sun, V. Jampani, M. Yang, E. Learned-Miller, and J. Kautz, "Super slomo: High quality estimation of multiple intermediate frames for video interpolation," in *IEEE. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 9000–9008.

[5] W. Bao, W. Lai, C. Ma, X. Zhang, Z. Gao, and M. Yang, "Depth-aware video frame interpolation," in *IEEE. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 3703–3712.

[6] S. Niklaus and F. Liu, "Context-aware synthesis for video frame interpolation," in *IEEE. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 1701–1710.

[7] ——, "Softmax splatting for video frame interpolation," in *IEEE. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 5437–5446.

[8] S. Gui, C. Wang, Q. Chen, and D. Tao, "Featureflow: Robust video interpolation via structure-to-texture generation," in *IEEE. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 14 004–14 013.

[9] S. Niklaus, L. Mai, and F. Liu, "Video frame interpolation via adaptive convolution," in *IEEE. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 670–679.

[10] ——, "Video frame interpolation via adaptive separable convolution," in *IEEE Int. Conf. on Computer Vision (ICCV)*, 2017, pp. 261–270.

[11] W. Bao, W. Lai, X. Zhang, Z. Gao, and M. Yang, "Memc-net: Motion estimation and motion compensation driven neural network for video interpolation and enhancement," *IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)*, 2019.

[12] S. Meyer, O. Wang, H. Zimmer, M. Grosse, and A. Sorkine-Hornung, "Phase-based frame interpolation for video," in *IEEE. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1410–1418.

[13] S. Meyer, A. Djelouah, B. McWilliams, A. Sorkine-Hornung, M. Gross, and C. Schroers, "Phasenet for video frame interpolation," in *IEEE. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 498–507.

[14] N. Kalantari, T. Wang, and R. Ramamoorthi, "Learning-based view synthesis for light field cameras," *ACM Trans. on Graphics (TOG)*, vol. 35, no. 6, pp. 193:1–193:10, 2016.

[15] E. Penner and L. Zhang, "Soft 3D reconstruction for view synthesis," *ACM Trans. on Graphics (TOG)*, vol. 36, no. 6, pp. 235:1–235:11, 2017.

[16] B. Mildenhall, P. Srinivasan, R. Ortiz-Cayon, N. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar, "Local light field fusion: Practical view synthesis with prescriptive sampling guidelines," *ACM Trans. on Graphics (TOG)*, vol. 38, no. 4, pp. 1–14, 2019.

[17] G. Wu, Y. Liu, Q. Dai, and T. Chai, "Learning sheared epi structure for light field reconstruction," *IEEE Trans. Image Process. (TIP)*, vol. 28, no. 7, pp. 3261–3273, 2019.

[18] L. Shi, H. Hassanieh, A. Davis, D. Katabi, and F. Durand, "Light field reconstruction using sparsity in the continuous fourier domain," *ACM Trans. on Graphics (TOG)*, vol. 34, no. 1, p. 12, 2014.

[19] S. Vagharshakyan, R. Bregovic, and A. Gotchev, "Light field reconstruction using shearlet transform," *IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)*, vol. 40, no. 1, pp. 133–147, 2018.

[20] Y. Wang, F. Liu, Z. Wang, G. Hou, Z. Sun, and T. Tan, "End-to-end view synthesis for light field imaging with pseudo 4DCNN," in *Eur. Conf. on Computer Vision (ECCV)*, 2018, pp. 333–348.

[21] B. Mildenhall, P. Srinivasan, M. Tancik, J. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *Eu. Conf. on Computer Vision (ECCV)*, 2020, pp. 405–421.

[22] Z. Wang, S. Wu, W. Xie, M. Chen, and V. Prisacariu, "NeRF−−: Neural radiance fields without known camera parameters," *ArXiv*, vol. abs/2102.07064, 2021.

[23] A. Chen, Z. Xu, F. Zhao, X. Zhang, F. Xiang, J. Yu, and H. Su, "MVSNeRF: Fast generalizable radiance field reconstruction from multi-view stereo," in *IEEE Int. Conf. on Computer Vision (ICCV)*, 2021.

[24] J. Chibane, A. Bansal, V. Lazova, and G. Pons-Moll, "Stereo radiance fields (SRF): Learning view synthesis for sparse views of novel scenes," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[25] J. Shi, X. Jiang, and C. Guillemot, "Learning fused pixel and feature-based view reconstructions for light fields," in *IEEE. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 2555–2564.

[26] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen, "The lumigraph," in *ACM Trans. on Graphics (TOG)*, 1996, pp. 43–54.

[27] G. Chaurasia, S. Duchene, O. Sorkine-Hornung, and G. Drettakis, "Depth synthesis and local warps for plausible image-based navigation," *ACM Trans. on Graphics (TOG)*, vol. 32, no. 3, pp. 1–12, 2013.

[28] Z. Zhang, Y. Liu, and Q. Dai, "Light field from micro-baseline image pair," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3800–3809.

[29] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely, "Stereo magnification: Learning view synthesis using multiplane images," *ACM Trans. on Graphics (TOG)*, vol. 37, no. 4, pp. 65:1–65:12, 2018.

[30] J. Flynn, I. Neulander, J. Philbin, and N. Snavely, "Deepstereo: Learning to predict new views from the world's imagery," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 5515–5524.

[31] S. Wanner and B. Goldluecke, "Variational light field analysis for disparity estimation and super-resolution," *IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)*, vol. 36, no. 3, pp. 606–619, Aug. 2013.

[32] N. Meng, H. So, X. Sun, and E. Lam, "High-dimensional dense residual convolutional neural network for light field reconstruction," *IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)*, 2019.

[33] D. Sun, X. Yang, M. Liu, and J. Kautz, "PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 8934–8943.

[34] J. Choi, J. Park, and I. Kweon, "High-quality frame interpolation via tridirectional inference," in *IEEE Winter Conf. on App. of Comput. Vis. (WACV)*, 2021, pp. 596–604.

[35] S. Niklaus, L. Mai, and O. Wang, "Revisiting adaptive convolutions for video frame interpolation," in *IEEE Winter Conf. on App. of Comput. Vis. (WACV)*, 2021, pp. 1099–1109.

[36] M. Levoy and P. Hanrahan, "Light field rendering," in *Proc. 23rd Annu. Conf. Comput. Graph. Interact. Techn.(CCGIT)*, 1996, pp. 31–42.

[37] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4040–4048.
[38] Z. Liang, Y. Feng, Y. Guo, H. Liu, W. Chen, L. Qiao, L. Zhou, and J. Zhang, "Learning for disparity estimation through feature constancy," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2811–2820.
[39] X. Jiang, J. Shi, and C. Guillemot, "A learning based depth estimation framework for 4D densely and sparsely sampled light fields," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 2257–2261.
[40] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 2017–2025.
[41] S. Tulsiani, R. Tucker, and N. Snavely, "Layer-structured 3D scene inference via view synthesis," in *Eur. Conf. on Computer Vision (ECCV)*, 2018, pp. 302–317.
[42] E. Ilg, T. Saikia, M. Keuper, and T. Brox, "Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation," in *Eu. Conf. on Computer Vision (ECCV)*, 2018.
[43] D. Butler, J. Wulff, and M. Black, "A naturalistic open source movie for optical flow evaluation," in *Eu. Conf. on Computer Vision (ECCV)*, 2012, pp. 611–625.
[44] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. V. D. Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *IEEE Int. Conf. on Computer Vision (ICCV)*, 2015.
[45] J. Shi, X. Jiang, and C. Guillemot, "A framework for learning depth from a flexible subset of dense and sparse light field views," *IEEE Trans. Image Process. (TIP)*, vol. 28, no. 12, pp. 5867–5880, Dec 2019.
[46] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. V. Gool, M. Gross, and A. Sorkine-Hornung, "A benchmark dataset and evaluation methodology for video object segmentation," in *IEEE. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.
[47] P. Bojanowski, A. Joulin, D. Lopez-Paz, and A. Szlam, "Optimizing the latent space of generative networks," *arXiv preprint arXiv:1707.05776*, 2017.
[48] K. Honauer, O. Johannsen, D. Kondermann, and B. Goldluecke, "A dataset and evaluation methodology for depth estimation on 4D light fields," in *Asian Conf. on Computer Vision (ACCV)*, 2016, pp. 19–34.
[49] T. Xue, B. Chen, J. Wu, D. Wei, and W. Freeman, "Video enhancement with task-oriented flow," *Int. J. Computer Vision (IJCV)*, vol. 127, no. 8, pp. 1106–1125, 2019.
[50] S. Wanner, S. Meister, and B. Goldluecke, "Datasets and benchmarks for densely sampled 4D light fields," in *Conf. on Vision, Modeling & Visualization (VMV)*, 2013, pp. 225–226.
[51] M. L. Pendu, C. Guillemot, and A. Smolic, "A fourier disparity layer representation for light fields," *IEEE Trans. Image Process. (TIP)*, vol. 28, no. 11, pp. 5740–5753, Nov 2019.
[52] R. Yang, F. Mentzer, L. Van Gool, and R. Timofte, "Learning for video compression with hierarchical quality and recurrent enhancement," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.
[53] ——, "Learning for video compression with recurrent auto-encoder and recurrent probability model," *IEEE J. Sel. Topics Signal Process. (JSTSP)*, vol. 15, no. 2, pp. 388–401, 2021.
[54] S. Su, M. Delbracio, J. Wang, G. Sapiro, W. Heidrich, and O. Wang, "Deep video deblurring for hand-held cameras," in *IEEE. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1279–1288.
[55] J. Bégaint, F. Galpin, P. Guillotel, and C. Guillemot, "Deep frame interpolation for video compression," in *IEEE Data Compression Conference (DCC)*, Mar. 2019, pp. 1–10.