Temporal-MPI: Enabling Multi-Plane Images for Dynamic Scene Modelling via Temporal Basis Learning

Wenpeng Xing and Jie Chen ✉

Department of Computer Science, Hong Kong Baptist University
{cswpxing, chenjie}@comp.hkbu.edu.hk

Fig. 1. Reconstruction quality demonstration of the proposed Temporal-MPI for the testing sequences from the Nvidia Dynamic Scene dataset [37]. The dynamic visual effects can be viewed in Adobe PDF Reader.

Abstract. Novel view synthesis of static scenes has achieved remarkable advancements in producing photo-realistic results. However, key challenges remain for immersive rendering of dynamic scenes. One of the seminal image-based rendering method, the multi-plane image (MPI), produces high novel-view synthesis quality for static scenes. But modelling dynamic contents by MPI is not studied. In this paper, we propose a novel Temporal-MPI representation which is able to encode the rich 3D and dynamic variation information throughout the entire video as compact temporal basis and coefficients jointly learned. Time-instance MPI for rendering can be generated efficiently using mini-seconds by linear combinations of temporal basis and coefficients from Temporal-MPI. Thus novel-views at arbitrary time-instance will be able to be rendered via Temporal-MPI in real-time with high visual quality. Our method is trained and evaluated on Nvidia Dynamic Scene Dataset. We show that our proposed Temporal-MPI is much faster and more compact compared with other state-of-the-art dynamic scene modelling methods.

Keywords: Multi-plane image, neural basis learning, novel view synthesis.

1 Introduction

Recent advancements on novel view synthesis have shown remarkable results on immersive rendering of static scenes using neural scene representations, such as Multi-plane Images (MPI) [40,35,17] and Neural Radiance Fields (NeRF) [18,4]. Neural basis expansion [35] and Plenoctree structures [38] have been recently proposed to further improve the rendering quality and efficiency. However, challenges still remain in modelling dynamic scenes, which require additional capacity to capture variations along time dimension.

To model dynamic contents, efforts have been made in training time-conditioned NeRF models [11,10,24]. Although photo-realistic view-synthesis results can be produced by these time-conditioned neural rendering methods, they normally require millions of ray-casting style queries during rendering, resulting in serious rendering delay and low frame rate. So, there is a popular branch of research on improving the rendering efficiency of neural scene representations by extracting the learned content into compact data structure, such as tree-based structure [38], or with occupancy priors [14] stored for efficient sampling. Another line of image-based rendering research, the MPI, focuses on rendering real-world forward-facing contents. MPI is highly efficient for real-time rendering due to its pre-computed 2.5D RGB-α volumes. In order to render dynamic scenes via MPI, pre-calculating and saving all time-instance MPIs is a straight-forward but engineering-oriented solution for time-space rendering. However, this method lacks temporal coherence and is expensive to save the bulky data incurred. *3DMaskVol21* [12] renders an image at a given timestamp by fusing a background MPI and instantaneous MPI using a 3D mask volume, which takes temporal-coherent information learned to be the background MPI. But generating these three volumes causes delay on rendering and heavy work-load on caching. In comparison, our proposed method can generate arbitrary time-instance MPIs from one Temporal-MPI within mini seconds, which is much more efficient for real-time rendering and compact in storage.

In this paper, we propose a novel efficient representation for dynamic scenes, Temporal-MPI, for space-time immersive rendering. Different from previous methods [11,12,37] which rely on pre-trained optical flow model [6], ground-truth background images [12], pre-trained depth estimation model [25] or dynamic-static masks [37] as additional premise, we aim at creating a self-contained pipeline. In addition, our method does not need to explicitly store time-instance MPIs, which greatly decreases the requirement for storage space and being computationally efficient.

2 Related Work

Novel view synthesis. Novel view synthesis is a long standing research issue that aims at synthesising novel views of a scene given arbitrary captured images, and has become one of the most popular classes of research topics in computer vision. Early researches on Light fields (LF) [21] represented the scene as a 4D Plenoptic

Function [16] $L(x, y, s, t)$, where (x, y) represents spatial coordinates and (s, t) represents angular coordinates. The spatial-angular correlations embedded in LF images can be exploited for applications of depth estimation [7,20], super-resolution [8] and novel view rendering [30,9]. With recent advancements in deep learning, different learning-based scene representations were proposed. Novel views can be synthesized from monocular input, they are *SynSin20* [34], *3D Photo20* [28], *WorldSheet21* [5] and *MPIs20* [33]. They share a common rationale, which is integrating the learning of geometry and appearance from rendering loss. The second branch of research is using multi-view inputs that allow machine learning models to reason the scene's geometry using epipolar geometry and triangulation, the scene can be learned as a volume representation either explicitly or implicitly. Neural/implicit volume representations can encode the scene as a continuous volumetric function, they are Deep Voxels [29] and NeRF [18]. In addition to above continuous volumetric functions, a scene can be decomposed into a layered representation [27], they are MPI [40] and its followers [17,33,35,12]. Although these methods can produce photo-realistic results, they can only model and render static scenes. The next key step of view rendering is rendering dynamic scenes.

Neural spatial and temporal embedding for novel-view synthesis. A successful novel view synthesis requires accurate modeling of a scene's geometry. Modeling the geometry of non-rigid scenes with dynamic contents are ill-posed, and were tackled by reconstructing dynamic 3D meshes where priors like temporal information [1,32] or known template configurations [2,19]. Yet, these methods require 2D-to-3D matches or 3D point tracks. Thus, limiting their applicability to real world scenes or simulated scenes with complex textures.

Under the context of space-time view synthesis, adding time parameters to the input of static scene's representations is a straightforward implementation. There are time-conditioned warping fields in *D-NeRF21* [24], scene flow fields in *NeuralFlow21* [11] and radiance fields in *Neural3DVideo21* [10]. More specifically, *D-NeRF21* [24] added a time-conditioned deformation network to predict the time-dependent positional offsets to deform the canonical NeRF into a time-instance shape. *NeuralFlow21* [11] used temporal photometric consistency to encourage the time-conditioned NeRF to be learned from monocular videos. Neural3DVideo21 [10] also transformed NeRF into a space-time domain, and achieved frame-interpolation by interpolating time latent vectors. However, the time-consuming rendering process of above NeRF-style methods limit their capabilities to real-time applications. Directly warping images to novel views according to depth is an efficient view-synthesis pipeline. *DynSyn20* [37] combined multi-view and single-view depths to generate temporal consistent depths for dynamic views warping. However, their method has two drawbacks: first, it requires foreground masks that separate static and dynamic contents; second, their method can not handle occlusions well. *3DMaskVol21* [12] proposed a method of generating dynamic MPI with a 3D mask volume to alleviate artifacts around the integration boundary of background and instantaneous MPIs. However, their method requires two-step training and background images. Thus, limiting

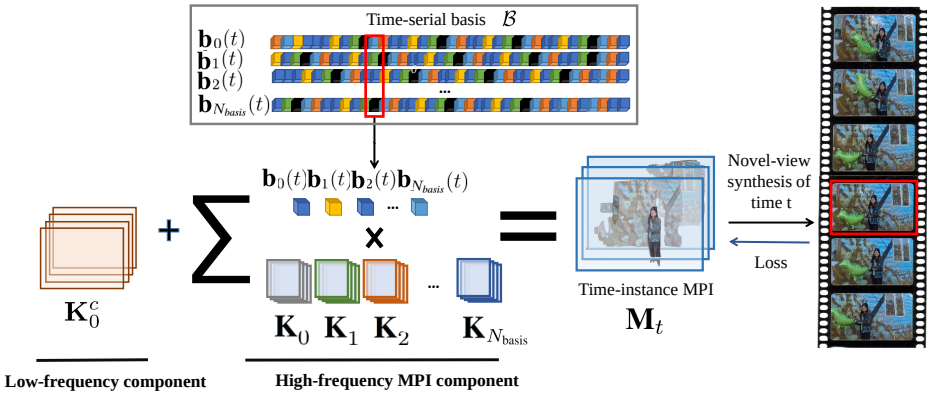


Fig. 2. Overall pipeline. The proposed Temporal-MPI contains three parts: low-frequency component $\mathbf{K}_0^c$, temporal basis $\mathcal{B}$ and high-frequency coefficients $\{\mathbf{K}_n\}_{n=1}^{N_{\text{basis}}}$. The alpha and color values in time-instantaneous MPI $\mathbf{M}_t$ are recovered as linear combinations of bases $\mathcal{B}$ and high-frequency coefficients $\{\mathbf{K}_n\}_{n=1}^{N_{\text{basis}}}$, and adding low-frequency component $\mathbf{K}_0^c$ from Temporal-MPI $\hat{\mathcal{M}}$. The color in the corresponding frame is rendered from time-instantaneous MPI $\mathbf{M}_t$ as MPI's alpha composition in Equation (1). The overall pipeline is differentiable and optimized per scene by pixel rendering loss.

their general capabilities. Compared with NeRF-style methods, *DynSyn20* and *3DMaskVol21*, our method is efficient on rendering and compact on storage.

Neural learnable basis. Our method is closely related to basis learning [13]. In signal processing, data often contains underlying structure that can be processed intelligently by linear combinations of *subspaces*. Tang et al. [31] learned subspace minimization for low-level vision tasks, such as interactive segmentation, video segmentation, optical flow estimation and stereo matching. PCA-Flow[36] predicted video's optical flows as a weighted sum of the basis flow fields. We take inspiration from these works, and learn coefficients to combine globally shared time-wise subspace to draw instantaneous MPIs.

3 Approach

Given a set of synchronized multi-view videos $\{\mathbf{I}_t^k\}$ of a dynamic scene, where $t = 1, 2, \cdots T$ are the frame number, and $k = 1, 2, \cdots, K$ are camera indices, our goal is to construct a *compact* 3D representation which enables *real-time* and *novel-view* synthesis of the dynamic contents at a given time $t \in [1, T]$. To achieve the goal, one naive option is to calculate and save a set of separate MPI $\mathcal{M} = \{\mathbf{M}_t \in \mathbb{R}^{H\times W\times D\times 4}\}_{t=1}^{T}$ for every video frame. This, however, will be extremely memory- and computation-inefficient (generating $\mathcal{M}$ incurs more than $225\times T$ MB data and around 2 seconds delay when rendering at VGA resolution [12]). As such, rather than having to calculate and save MPIs for all video frames in advance or having to calculate an MPI on-the-run, we investigate a novel

Temporal-MPI representation with learned temporal basis to compactly encode high-frequency variation throughout the entire video. An overall pipeline of our approach is shown in Fig. 2.

In the following, we will first briefly introduce the vanilla MPI representation in Section 3.1. Then, the temporal basis formulation will be elaborated in Section 3.2, and the novel-view temporal reconstruction in Section 3.3.

3.1 The Multi-plane Image Representation

Being one of the seminal representation frameworks for 3D content embedding and novel-view synthesis, Multi-plane Images (MPI) learn a layered depth decomposition of the scene from a set of multi-view references [40,17,40]. Following the MPI's illustration in Nex [35], let D denote the number of depth layers in a MPI, with the dimension of each layer being $H \times W \times 4$, where H and W denote the height and width of the MPI layers, 4 denotes 3-channel RGB and 1-channel alpha α. So we denote an MPI representation as $\mathbf{M} = \{\mathbf{C}_d, \mathbf{A}_d\}_{d=1}^{D}$, where $\mathbf{C}_d \in \mathbb{R}^{H\times W\times 3}$ are multiple layers of 3-channel RGB images and $\mathbf{A}_d \in \mathbb{R}^{H\times W\times 1}$ are one-channel alpha images, d denotes the depth plane index.

Synthesizing novel-views $\hat{\mathbf{I}}$ based on the MPI $\mathbf{M}$ involves two steps: first, warp all depth planes in the MPI homographically from a reference view to a source view; and second, render pixels using alpha-composition [23] over each layer's color:

$$\hat{\mathbf{I}} = \mathcal{O}(\mathcal{W}(\mathbf{A}), \mathcal{W}(\mathbf{C})), \tag{1}$$

here $\mathcal{W}$ denotes the warping operator, and $\mathcal{O}$ denotes the compositing operator. The compositing operator $\mathcal{O}$ is defined as:

$$\mathcal{O}(\mathbf{A}, \mathbf{C}) = \sum_{d=1}^{D} \mathbf{C}_d \mathcal{T}_d(\mathbf{A}), \tag{2}$$

$$\mathcal{T}_d(\mathbf{A}) = \mathbf{A}_d \prod_{i=d+1}^{D} (1 - \mathbf{A}_i). \tag{3}$$

where $\prod_{i=d+1}^{D}(1-\mathbf{A}_i)$ are accumulated transmittance, $\mathcal{T}_d$ are opacity. The output of $\mathcal{O}(\mathbf{A}, \mathbf{C})$ are final rendered colors. Both the composition $\mathcal{O}$ and the warping $\mathcal{W}$ operations are differentiable, thus allowing the representation $\mathbf{M}$ to learn the geometry and color information from final pixel rendering loss.

3.2 Temporal Basis Formulation

At a given time instance $t \in [1, T]$, we denote the time-instance MPI as $\mathbf{M}_t$. In order to render the entire novel view sequence at sequential timestamps, a set of time-instance MPIs $\mathcal{M} = \{\mathbf{M}_t \in \mathbb{R}^{H\times W\times D\times 4}\}_{t=1}^{T}$ are needed to generate. Based on the afore-analyzed reasons, we cannot exhaustively calculate and save $\mathcal{M}$. We propose a novel Temporal-MPI representation which is able to encode the

rich 3D and high-frequency variation information throughout the entire video as compact temporal basis, and in the meantime, preserve high rendering efficiency for real-time novel-view synthesis. To achieve this, we divide the goal into two tasks, i.e., (i) learning the low-frequency color components as explicit parameters, and (ii) learning the high-frequency variation over a set of temporal basis.

Explicit Parameter Learning for Low-frequency Component Low-frequency contents in a video constitute the low-frequency part of the total energy along the time dimension, which can be well-captured and modeled explicitly by time-invariant parameters. By treating all the frames of the multi-view video $\{\mathbf{I}_t^k\}_{t,k}$ as source views equally and ignoring their respective frame indices, we can directly learn the multi-plane time-invariant RGB color parameters $\mathbf{K}_0^{\mathrm{c}} \in \mathbb{R}^{H\times W\times D/8\times 3}$ using the pixel rendering loss. $\mathbf{K}_0^{\mathrm{c}}$ models the low-frequency energy of the video, with possible blur over the dynamic area. Such an explicit modelling scheme for the low-frequency component proves to be important [35] and let the subsequent dynamic modelling to better focus on the temporal variation.

Temporal Basis Learning for High-Frequency Contents Compared with low-frequency components, the high-frequency contents in $\mathcal{M}$ constitute the high-frequency energy along the time dimension. Being high-dimensional and with dynamic variations, the high-frequency contents still constitute a highly regularized manifold, considering the fact that (i) the video length is limited (we model video with 24 frames in length, although these frames could be extracted from longer video sequences), and (ii) time-variant pixels within a scene usually show consistent motion in clusters. This motivates us to compactly represent the high-frequency components based on a few learned time-variant temporal basis.

We denote the temporal basis as $\mathcal{B} \in \mathbb{R}^{4\times T\times N_{\text{basis}}}$ which span the temporal variation space for $\mathcal{M}$. Here N_{basis} denotes the total number of basis. The first dimension of $\mathcal{B}$ is set to 4 which is reserved for modelling both the MPI color component (with 3 channels): $\mathcal{B}^c = \{\mathbf{b}_n^c\}_{n=1}^{N_{\text{basis}}}$, and the alpha component $\mathcal{B}^\alpha = \{\mathbf{b}_n^\alpha\}_{n=1}^{N_{\text{basis}}}$ (1 channel). Therefore $\mathcal{B} = [\mathcal{B}^c, \mathcal{B}^\alpha]$.

In our proposed framework, the temporal basis will be estimated by two time-dependent functions which are Multi-Layer Perceptron (MLP) networks $\mathcal{V}^c$ and $\mathcal{V}^\alpha$:

$$\{\mathbf{b}_n^c(t)\}_{n=1}^{N_{\text{basis}}} = \mathcal{V}^c(\mathcal{E}(t)) :\ \mathbb{R} \mapsto \mathbb{R}^{3\times N_{\text{basis}}}, \tag{4}$$

$$\{\mathbf{b}_n^\alpha(t)\}_{n=1}^{N_{\text{basis}}} = \mathcal{V}^\alpha(\mathcal{E}(t)) :\ \mathbb{R} \mapsto \mathbb{R}^{1\times N_{\text{basis}}}. \tag{5}$$

Here $\mathcal{E}(\cdot)$ is a time-encoding function which encodes time-sequential information into a high dimensional latent vector [10]. The temporal basis $\mathcal{B}$ learns a parsimonious frame that efficiently spans the temporal variation manifold. With a *pixel-specific* coding coefficient (to be elaborated in the next section), $\mathcal{B}$ can efficiently model the MPI pixel's temporal variation throughout the entire video.

3.3 Temporal Coding for Novel-view Synthesis

For an arbitrary frame index $t \in [1, T]$, a time-instance MPI $\mathbf{M}_t = [\mathbf{A}_t, \mathbf{C}_t]$ can be constructed based on the temporal basis $\mathcal{B}$ according to:

$$\mathbf{C}_t(\mathbf{x}) = \mathbf{K}_0^c(\mathbf{x}) + \sum_{n=1}^{N_{\text{basis}}} \mathbf{K}_n^c(\mathbf{x}) \times \mathbf{b}_n^c(t), \tag{6}$$

$$\mathbf{A}_t(\mathbf{x}) = \sum_{n=1}^{N_{\text{basis}}} \mathbf{K}_n^\alpha(\mathbf{x}) \times \mathbf{b}_n^\alpha(t). \tag{7}$$

Here $\mathbf{K}_n^\alpha(\mathbf{x})$ and $\mathbf{K}_n^c(\mathbf{x})$ are the coding coefficients for the respective temporal basis at a given MPI spatial location $\mathbf{x} \in \mathbb{R}^3$ (the 3 dimensions of $\mathbf{x}$ include its 2D coordinates and the depth plane index in $\mathcal{M}_t$). These coding coefficients are estimated by another set of MLPs $\mathcal{K}^c$ and $\mathcal{K}^\alpha$:

$$\{\mathbf{K}_n^c(\mathbf{x})\}_{n=1}^{N_{\text{basis}}} = \mathcal{K}^c(\mathcal{R}(\mathbf{x})) : \mathbb{R}^3 \mapsto \mathbb{R}^{3 \times N_{\text{basis}}}, \tag{8}$$

$$\{\mathbf{K}_n^\alpha(\mathbf{x})\}_{n=1}^{N_{\text{basis}}} = \mathcal{K}^\alpha(\mathcal{R}(\mathbf{x})) : \mathbb{R}^3 \mapsto \mathbb{R}^{1 \times N_{\text{basis}}}. \tag{9}$$

Similarly, here $\mathcal{R}(\cdot)$ is a position-encoding function which encodes the spatial information $\mathbf{x}$ into high-dimensional representations [18].

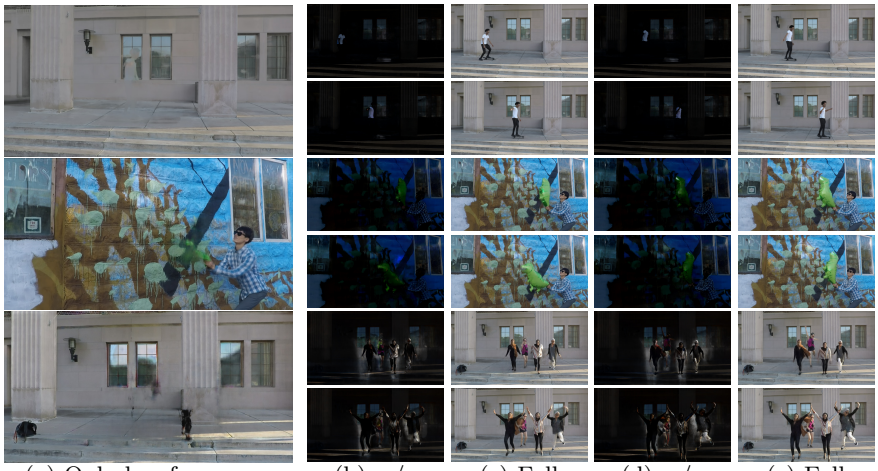
Based on Equation (6) and (7), the time-instance MPI $\mathbf{M}_t$ can be warped and composited to any arbitrary viewing angles according to Equation (2) and (1). In addition, by querying all elements $t = 1, \cdots, T$ along the temporal basis, we can construct the time-instance MPI for each video frame.

Remarks. (i) our proposed temporal MPI representation composes of an explicitly learned low-frequency multi-plane color component $\mathbf{K}_0^c \in \mathbb{R}^{H \times W \times D/8 \times 3}$, and a dynamically coded time-variant component via simultaneous basis and coefficient learning. We have achieved compression along the temporal dimension via the temporal basis, which compactly encodes time-variant color and geometry variation information throughout the entire video.

(ii) To maintain rendering efficiency and save storage-space, spatial-temporal information is efficiently encoded and propagated among different components in the Temporal-MPI. First, the low-frequency component $\mathbf{K}_0^c$ is temporally shared among all time frames, this ensures overall reconstruction quality and enables the high-frequency components to focus on time-dependent variations only; and second, the high-frequency coefficients, i.e., $\{\mathbf{K}_n^c(\mathbf{x})\}_{n=1}^{N_{\text{basis}}}$ and $\{\mathbf{K}_n^\alpha(\mathbf{x})\}_{n=1}^{N_{\text{basis}}}$, are point-wisely coded/learned, however, over a common set of temporal basis. This helps to remove the redundancy in modelling dynamic variation, and also helps to remove motion ambiguities for some pixels.

3.4 Training Loss Function

To let the Temporal-MPI focus on reconstruction quality, we ignore the sparsity of coding coefficients for this task. Coefficients and the temporal basis are jointly



(a) Only low-frequency (b) w/o (c) Full (d) w/o (e) Full

Fig. 3. Low-frequency scene representation ablation study. The low-frequency components are rendered in (a); color output from the high-frequency components are rendered in (b) and (d); full rendering are visualised in (c) and (e).

learned and optimized. The whole system is optimized via the following loss function $\mathcal{L}$:

$$\mathcal{L} = ||\hat{\mathbf{I}}_t^k - \mathbf{I}_t^k||_2 + \lambda_1||\nabla\hat{\mathbf{I}}_t^k - \nabla\mathbf{I}_t^k||_1 + \lambda_2 \mathrm{TVC}(\mathbf{K}_0^c), \tag{10}$$

where $\hat{\mathbf{I}}_t^k$ is the rendered image at time t for the camera k, $\mathbf{I}_t^k$ is the ground truth image from the same view. The first term in $\mathcal{L}$ calculates the L_2 reconstruction loss. The second term penalises edge inconsistencies, with ∇ denoting the gradient operator. In the third term, TVC denotes total variation loss [3]. λ_1 and λ_2 are balancing weights for different loss terms.

4 Experiments

4.1 Implementation Details

Our model is implemented in PyTorch 1.10, using Adam as optimiser. The initial learning rate is set as 0.001, and decay by 0.1 every 2000 steps. The model takes 16 hours to be trained on one Nvidia Geforce RTX 2070 Super GPU with a batch of 1500 rays, using 5.3 GB memory. The output resolution is 576 $\times$ 300. The position-encoding method in [18] is formulated as $\mathcal{R}(p) = [sin(2^0\frac{\pi}{2}p), sin(2^0\frac{\pi}{2}p), \ldots, sin(2^l\frac{\pi}{2}p), cos(2^l\frac{\pi}{2}p)]$ where the input location of scene point is normalised to [-1, 1] and l is the index of encoding level set as 3. The index of time is embedded into a latent vector in size of 32 using dictionary learning as in [10]. For networks that parameterise $\mathcal{K}^c$ and $\mathcal{K}^\alpha$, we use MLP networks with 8 layers and 384 hidden nodes. Networks for $\mathcal{V}^c$ and $\mathcal{V}^\alpha$ are using MLP with 4 layers

and 64 hidden nodes. The shape of high-frequency coefficients $\{\mathbf{K}_n^c(\mathbf{x})\}_{n=1}^{N_{\text{basis}}}$ and $\{\mathbf{K}_n^\alpha(\mathbf{x})\}_{n=1}^{N_{\text{basis}}}$ in Temporal-MPI is $320 \times 596 \times 32 \times 4 \times 5$ where 32 is the number of planes D, 596 and 320 are width W and height H including marginal offsets set as 10, 4 includes 3 channels of colors and 1 channel of alpha, and 5 is the number of basis N_{basis}. The shape of temporal basis $\mathcal{B}$ is $4 \times 5 \times 24$ where 5 is the number of basis N_{basis}, 24 is the total number of timestamps and 4 includes 3 channels for color and 1 channel for alpha. Low-frequency component $\mathbf{K}_0^c$ is in the shape of $320 \times 596 \times 4 \times 3$ before the repetition along depth dimension.

4.2 Dataset

Our model is trained and evaluated on the Nvidia Dynamic Scenes Dataset [37] that contains 8 scenes with motions recorded by 12 synchronized cameras. Nvidia Dynamic Scenes Dataset captures a dynamic scene with static background via stationary cameras which suit our goal of separately learning low- and high-frequency components well. We extract camera parameters for every camera using COLMAP [26]. We extracted 24 frames from the video sequence, and used multi-view images in selected frames for training. We select camera views 1-11 for training, and camera 12 for testing. So the total number of training images is 264. The camera location arrangement is shown in Fig. 4.

6	5	4	3	2	1
7	8	9	10	11	12

Fig. 4. Camera indexes in camera array.

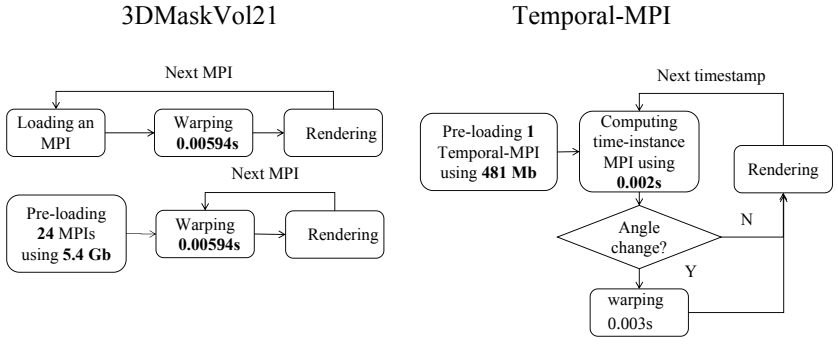


Fig. 5. Rendering pipeline comparison.

Table 1. Ablation study on PSNR vs. total number of timestamps. With D as 32 Planes.

Scene/PSNR	Number of timestamps						
	8	16	24	32	40	48	60
Skating-2	30.323	28.813	28.575	28.612	27.324	25.431	25.012
Truck-2	28.441	28.174	28.056	27.951	27.591	25.332	24.963
Jumping	25.850	25.661	25.486	25.301	25.001	24.759	23.854
Balloon2-2	25.775	25.381	25.171	24.893	24.557	24.474	23.945

4.3 Ablation Study

In this section, we investigate the effectiveness of our main contributions in Temporal-MPI: high-frequency coefficients, temporal basis and low-frequency component.

Video Length We evaluate the performance of our model trained with different length of videos. As shown in Table 1, We found performance degradation when the total number of timestamps T increased. This is due to reaching the representation threshold of temporal basis and high-frequency coefficients.

Low-frequency Colors and High-frequency Coefficients To validate the contributions of low-frequency component $\mathbf{K}_0^c$, high-frequency components and time-serial basis ($\{\mathbf{K}_n^c(\mathbf{x})\}_{n=1}^{N_{\text{basis}}}$, $\{\mathbf{K}_n^\alpha(\mathbf{x})\}_{n=1}^{N_{\text{basis}}}$, $\mathcal{B}$), we performed experiments without these modules. As shown in Table 2, without low-frequency component or high-frequency components will lead to a worse result than the full model. We separately render the low-frequency and high-frequency parts of the Temporal-MPI to prove their individual contributions. The visualization of separate rendering settings are shown in Fig. 3. The low-frequency component in Fig. 3 is calculated by directly summing $\mathbf{K}_0^c$ across depth planes. We wish to highlight that it is designed to facilitate the MLP to focus on modelling the high-frequency residual by explicitly modelling the low frequency content. It can be seen that the low-frequency components successfully capture the low-frequency energy of the video, while the high-frequency components complement the low-frequency ones to produce high quality rendering for dynamic scenes.

Inference Speed In this section, we investigate the relationships between inference speed and the size of Temporal-MPI. From Table 3, we can find that the computations of linear combinations of basis are efficient, and a big volume size of Temporal-MPI will not affect its real-time performance. Inference speed experiments are conducted on one Nvidia Tesla V100 GPU.

Table 2. Ablation study of low-frequency and high-frequency components.

Methods	No. Planes	Metrics		
		SSIM (↑)	PSNR (↑)	LPIPS (↓)
w/o low-frequency	32	0.192	10.3	0.726
w/o high-frequency	32	0.611	22.6	0.213
Full	32	0.859	24.87	0.196

Table 3. Inference time vs. shape of Temporal-MPI.

Resolution	No.Basis	No.Planes	Inference Time (seconds, ↓)
596×320	5	32	0.002
596×320	13	32	0.003
1038×1940	5	32	0.025
1038×1940	13	32	0.029
1038×1940	5	192	0.030

4.4 Evaluation and Comparison

To prove the efficiency and compactness of our method, we first compare our method with state-of-the-art algorithms, *3DMaskVol21* [12] and *NeuralFlow21* [11], in terms of storage space and inference speed. Then, we evaluate the view synthesis quality with other methods.

Evaluation on Compactness of Representation One of the main objectives of our approach is to learn a compact representation of a dynamic scene. So we evaluate the compactness of Temporal-MPI by comparing the number of network parameters and storage space with these two methods. As shown in Table 5, modeling a dynamic scene with 24 timestamps, Temporal-MPI only occupies 481 Mb for storage, which is 11 times smaller than *3DMaskVol21* on storage space. So our approach is extremely fast and compact for real-time rendering.

Evaluation on Efficiency From Table 5, we can find that i) rendering NeuralFlow21 requires querying MLP exhaustively, so it is the slowest on rendering. ii) our rendering time is much faster than 3DMaskVol21: as shown in Fig. 5, 3DMaskVol21 requires per-frame warping, which is not a mandatory step of ours; it also requires per-frame MPI loading, but we only need to load once for the entire sequence, therefore longer sequence will bring more advantages to our efficiency. Considering both loading and rendering time, ours are much faster on rendering than both NeuralFlow21 and 3DMaskVol21 (on a T=24 frame video). But 3DMaskVol21 is a generic method that generalizes to novel scenes, so it saves the cost of per-scene training. NeuralFlow21 has the highest rendering quality due to its advantages on dense sampling in depth dimensions.

Table 4. Quantitative evaluation of novel view synthesis on the Dynamic Scenes dataset. MV denotes whether the approach uses multi-view information or not, Ind. Src denotes the index of source views used to train the model. Ours (D=32) denotes using 32 planes in MPI.

Methods	Ind. Src	MV	Metrics		
			SSIM (↑)	PSNR (↑)	LPIPS (↓)
SynSin20 [34]	3	No	0.488	16.21	0.295
MPIs20 [33]	3	No	0.629	19.46	0.367
3D Ken Burn19 [22]	3	No	0.630	19.25	0.185
3D Photo20 [28]	3	No	0.614	19.29	0.215
NeRF20 [18]	$1-11$	Yes	0.893	24.90	0.098
ConsisVideoDepth20 [15]	3	Yes	0.746	21.37	0.141
DynSyn20 [37]	$1-11$	Yes	0.761	21.78	0.127
NeuralFlow21 [11]	3	Yes	**0.928**	**28.19**	**0.045**
D-NeRF21 [24]	$1-11$	Yes	0.334	17.05	0.545
3DMaskVol21 [12]	$3, 9$	Yes	0.603	20.10	0.285
Ours (D=32)	$1-11$	Yes	0.859	24.87	0.196

Evaluation on View-Synthesis Quality We evaluate the effectiveness of our approach by comparing it to baseline methods quantitatively and qualitatively. We compare our approach with state-of-the-art single-view or multi-view novel view synthesis methods. For monocular methods, we compare with *SynSin20* [34] and *MPIs20* [33] trained on RealEstate 10K dataset [40]. *3D Photo20* [28] and *3D Ken Burns19* [22] were trained by wild images. For multi-view methods, we compare with *NeRF20* [18], *ConsisVideoDepth20* [15], *DynSyn20* [37], *NeuralFlow21* [11], *3DMaskVol21* [12] and *D-NeRF21* [24]. Results are referenced from recent publications [11,12]. We document the rendering quality in three error metrics: structural similarity index measure (SSIM), peak signal-to-noise ratio (PSNR), and perceptual similarity through LPIPS [39]. From Table 4, our algorithm has competitive average score across three metrics. Per-scene breakdown results are shown in Table 6.

Qualitative comparisons can be seen in Fig. 6, which show that our method achieves competitive rendering quality in both low- and high-frequency parts. The visual results of *3D Photo20* in Fig. 6 (a), *NeRF20* in Fig. 6 (b) and *DynSyn20* in Fig. 6 (c) are referenced from [11]. Observed from above images, *D-NeRF21* in Fig. 6 (e) produces blurry results, *DynSyn20* has great artifacts on thin structures, *3D Photo20* generates distortions, *3DMaskVol21* produces ghosting effects around the object's boundary given scenes with forward moving motions, such as Jumping and Umbrella.

4.5 Baseline for Brute-force Scenario

To compare with brute-force scenario where an MPI is calculated for each time frame. We have tested LLFF19 [17] that includes all views 1-11 in a local fusion

Table 5. Comparison on real-time rendering/inference speed, output resolution and storage space. We assume the length of the modeled dynamic video is 24 frames. *3DMaskVol21* will require pre-loading 24 MPIs for real-time rendering of the whole sequence. *NeuralFlow21* is impossible for real-time tasks.

Time/Methods	*NeuralFlow21*	*3DMaskVol21*	Ours
MPI Generation Time (sec,↓)	-	2	**0.002**
MPI Loading Time (sec,↓)	-	0.043	**0.083**/T
Warping Time (sec,↓)	-	0.00594	**0.003**
Rendering Time (sec/frame,↓)	6	0.049	**0.008**(T=24)
Output Resolution(pixel,↑)	512 × 288	**640** × **360**	576× 300
Network Parameters (million, ↓)	5.26	**1.17**	6.00
Storage Space (Mb,↓)	–	225×T (24)=5400	**481** × 1(D=32)
Training Time (hour, ↓)	48†	120	16†

† denotes scene specific training.

Table 6. Per-scene breakdown results from *DynSyn20*'s Dynamic Scenes dataset.

	Skating-2	Balloon1-2	Jumping	Playground	Balloon2-2	Truck-2	Average
PSNR(↑)	28.575	21.309	25.486	20.594	25.171	28.056	24.865
SSIM(↑)	0.925	0.802	0.886	0.7211	0.885	0.937	0.859
LPIPS(↓)	0.163	0.239	0.202	0.253	0.171	0.150	0.196

manner, and view 12 for testing. It takes 39.5649 seconds to infer MPIs for a single frame, with average PSNR and SSIM 35.41 and 0.95, compared to 31.94 and 0.917 of the Temporal-MPI. Note that the baseline calculates and fuses several static MPIs for each frame, while we only calculate one neural MPI for the entire sequence.

5 Concluding Remarks

5.1 Limitations

Modeling dynamic scenes is challenging due to complex motions of dynamic objects over time, and specular surface and occlusions on angular domain. Our method makes the first attempt to use a compact temporal representation to reproduce dynamic scenes in time-sequences. Similar to *NeRF20*, our method requires optimization for each scene. Additionally, the output resolution is limited due to limited GPU memory. Furthermore, the rendering quality degrades when the length of sequence increases given default model parameters. Our approach is also only applicable to dynamic scenes without large camera motions that cause the change of background.

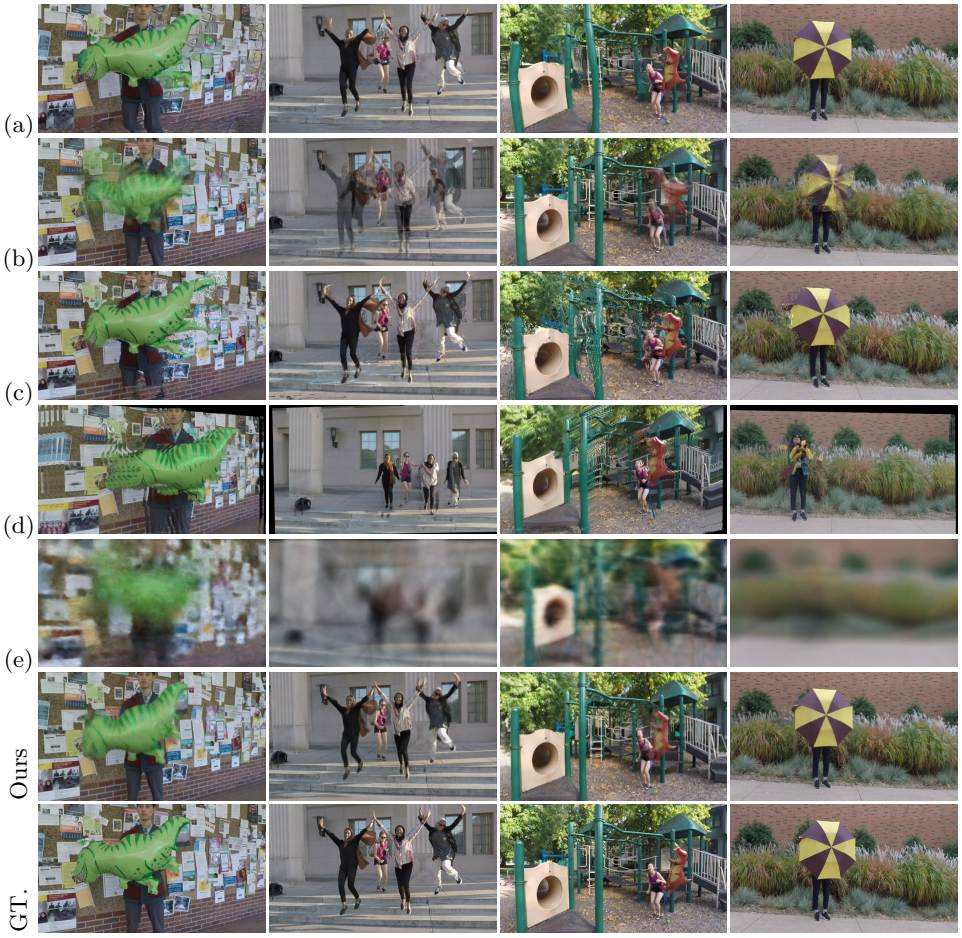


Fig. 6. Qualitative comparisons on the Dynamic Scenes dataset.

5.2 Conclusion

We have proposed a novel dynamic scene representation on top of Multi-plane Image (MPI) with basis learning. Our representation is efficient in computing, thus allowing real-time rendering of dynamics. Extensive studies on public dataset demonstrate the competitive rendering quality and efficiency of our approach. We believe using basis learning for temporal recovery and compression can be applied to the general problem of modeling dynamic contents and not limited to MPI. Using hierarchical encoding method to improve the learning power of MLP on modeling long-time-serial data could be a future extension of our work.

Acknowledgments The research was supported by the Theme-based Research Scheme, Research Grants Council of Hong Kong (T45-205/21-N).

References

1. Agudo, A., Moreno-Noguer, F.: Simultaneous pose and non-rigid shape with particle dynamics. In: CVPR. pp. 2179–2187 (2015)
2. Bartoli, A., Gérard, Y., Chadebecq, F., Collins, T., Pizarro, D.: Shape-from-template. IEEE transactions on pattern analysis and machine intelligence **37**(10), 2099–2118 (2015)
3. Chambolle, A., Lions, P.L.: Image recovery via total variation minimization and related problems. Numerische Mathematik **76**(2), 167–188 (1997)
4. Chen, A., Xu, Z., Zhao, F., Zhang, X., Xiang, F., Yu, J., Su, H.: Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In: CVPR. pp. 14124–14133 (2021)
5. Hu, R., Ravi, N., Berg, A.C., Pathak, D.: Worldsheet: Wrapping the world in a 3d sheet for view synthesis from a single image. In: ICCV. pp. 12528–12537 (2021)
6. Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: Flownet 2.0: Evolution of optical flow estimation with deep networks. In: CVPR. pp. 2462–2470 (2017)
7. Jeon, H.G., Park, J., Choe, G., Park, J., Bok, Y., Tai, Y.W., Kweon, I.S.: Accurate depth map estimation from a lenslet light field camera. In: CVPR. pp. 1547–1555 (2015)
8. Jin, J., Hou, J., Chen, J., Zeng, H., Kwong, S., Yu, J.: Deep coarse-to-fine dense light field reconstruction with flexible sampling and geometry-aware fusion. IEEE Transactions on Pattern Analysis and Machine Intelligence pp. 1–1 (2020)
9. Kalantari, N.K., Wang, T.C., Ramamoorthi, R.: Learning-based view synthesis for light field cameras. ACM Transactions on Graphics **35**(6), 1–10 (2016)
10. Li, T., Slavcheva, M., Zollhoefer, M., Green, S., Lassner, C., Kim, C., Schmidt, T., Lovegrove, S., Goesele, M., Newcombe, R., et al.: Neural 3d video synthesis from multi-view video. In: CVPR. pp. 5521–5531 (2022)
11. Li, Z., Niklaus, S., Snavely, N., Wang, O.: Neural scene flow fields for space-time view synthesis of dynamic scenes. In: CVPR. pp. 6498–6508 (2021)
12. Lin, K.E., Xiao, L., Liu, F., Yang, G., Ramamoorthi, R.: Deep 3d mask volume for view synthesis of dynamic scenes. In: ICCV. pp. 1749–1758 (2021)
13. Liu, G., Lin, Z., Yan, S., Sun, J., Yu, Y., Ma, Y.: Robust recovery of subspace structures by low-rank representation. IEEE transactions on pattern analysis and machine intelligence **35**(1), 171–184 (2012)
14. Liu, L., Gu, J., Zaw Lin, K., Chua, T.S., Theobalt, C.: Neural sparse voxel fields. Advances in Neural Information Processing Systems **33**, 15651–15663 (2020)
15. Luo, X., Huang, J.B., Szeliski, R., Matzen, K., Kopf, J.: Consistent video depth estimation. ACM Transactions on Graphics **39**(4) (2020)
16. McMillan, L., Bishop, G.: Plenoptic modeling: An image-based rendering system. In: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques. pp. 39–46 (1995)
17. Mildenhall, B., Srinivasan, P.P., Ortiz-Cayon, R., Kalantari, N.K., Ramamoorthi, R., Ng, R., Kar, A.: Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. ACM Transactions on Graphics **38**(4), 1–14 (2019)
18. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: NERF: Representing scenes as neural radiance fields for view synthesis. In: ECCV. pp. 405–421 (2020)
19. Moreno-Noguer, F., Fua, P.: Stochastic exploration of ambiguities for nonrigid shape recovery. IEEE transactions on pattern analysis and machine intelligence **35**(2), 463–475 (2012)

20. Navarro, J., Buades, A.: Robust and dense depth estimation for light field images. IEEE Transactions on Image Processing **26**(4), 1873–1886 (2017)
21. Ng, R., Levoy, M., Brédif, M., Duval, G., Horowitz, M., Hanrahan, P.: Light field photography with a hand-held plenoptic camera. Ph.D. thesis, Stanford University (2005)
22. Niklaus, S., Mai, L., Yang, J., Liu, F.: 3d ken burns effect from a single image. ACM Transactions on Graphics. **38**(6) (2019)
23. Porter, T., Duff, T.: Compositing digital images. SIGGRAPH Comput. Graph. **18**(3), 253–259 (1984)
24. Pumarola, A., Corona, E., Pons-Moll, G., Moreno-Noguer, F.: D-NERF: Neural radiance fields for dynamic scenes. In: CVPR. pp. 10318–10327 (2021)
25. Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., Koltun, V.: Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. IEEE Transactions on Pattern Analysis and Machine Intelligence p. 1 (2020)
26. Schönberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: CVPR. pp. 4104–4113 (2016)
27. Shade, J., Gortler, S., He, L.w., Szeliski, R.: Layered depth images. In: ACM SIGGRAPH. pp. 231–242 (1998)
28. Shih, M.L., Su, S.Y., Kopf, J., Huang, J.B.: 3D photography using context-aware layered depth inpainting. In: CVPR. pp. 8025–8035 (2020)
29. Sitzmann, V., Thies, J., Heide, F., Nießner, M., Wetzstein, G., Zollhofer, M.: Deepvoxels: Learning persistent 3d feature embeddings. In: ICCV. pp. 2437–2446 (2019)
30. Srinivasan, P.P., Wang, T., Sreelal, A., Ramamoorthi, R., Ng, R.: Learning to synthesize a 4D RGBD light field from a single image. In: ICCV. pp. 2243–2251 (2017)
31. Tang, C., Yuan, L., Tan, P.: Lsm: Learning subspace minimization for low-level vision. In: CVPR. pp. 6235–6246 (2020)
32. Tomasi, C., Kanade, T.: Shape and motion from image streams under orthography: a factorization method. International journal of computer vision **9**(2), 137–154 (1992)
33. Tucker, R., Snavely, N.: Single-view view synthesis with multiplane images. In: CVPR. pp. 551–560 (June 2020)
34. Wiles, O., Gkioxari, G., Szeliski, R., Johnson, J.: SynSin: End-to-end view synthesis from a single image. In: CVPR. pp. 7465–7475 (2020)
35. Wizadwongsa, S., Phongthawee, P., Yenphraphai, J., Suwajanakorn, S.: Nex: Real-time view synthesis with neural basis expansion. In: CVPR. pp. 8534–8543 (2021)
36. Wulff, J., Black, M.J.: Efficient sparse-to-dense optical flow estimation using a learned basis and layers. In: CVPR. pp. 120–130 (2015)
37. Yoon, J.S., Kim, K., Gallo, O., Park, H.S., Kautz, J.: Novel view synthesis of dynamic scenes with globally coherent depths from a monocular camera. In: CVPR. pp. 5336–5345 (2020)
38. Yu, A., Li, R., Tancik, M., Li, H., Ng, R., Kanazawa, A.: Plenoctrees for real-time rendering of neural radiance fields. In: ICCV. pp. 5752–5761 (2021)
39. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: CVPR. pp. 586–595 (2018)
40. Zhou, T., Tucker, R., Flynn, J., Fyffe, G., Snavely, N.: Stereo magnification: learning view synthesis using multiplane images. ACM Transactions on Graphics **37**(4), 1–12 (2018)