

KFD-NeRF: Rethinking Dynamic NeRF with Kalman Filter

Yifan Zhan¹, Zhuoxiao Li¹, Muyao Niu¹, Zhihang Zhong³, Shohei Nobuhara², Ko Nishino², and Yinqiang Zheng^{*1}

¹ The University of Tokyo

² Kyoto University

³ Shanghai Artificial Intelligence Laboratory

Abstract. We introduce KFD-NeRF, a novel dynamic neural radiance field integrated with an efficient and high-quality motion reconstruction framework based on Kalman filtering. Our key idea is to model the dynamic radiance field as a dynamic system whose temporally varying states are estimated based on two sources of knowledge: observations and predictions. We introduce a novel plug-in Kalman filter guided deformation field that enables accurate deformation estimation from scene observations and predictions. We use a shallow Multi-Layer Perceptron (MLP) for observations and model the motion as locally linear to calculate predictions with motion equations. To further enhance the performance of the observation MLP, we introduce regularization in the canonical space to facilitate the network’s ability to learn warping for different frames. Additionally, we employ an efficient tri-plane representation for encoding the canonical space, which has been experimentally demonstrated to converge quickly with high quality. This enables us to use a shallower observation MLP, consisting of just two layers in our implementation. We conduct experiments on synthetic and real data and compare with past dynamic NeRF methods. Our KFD-NeRF demonstrates similar or even superior rendering performance within comparable computational time and achieves state-of-the-art view synthesis performance with thorough training. Github page: <https://github.com/Yifever20002/KFD-NeRF>.

Keywords: Dynamic NeRF · Deformable Network · Kalman Filter

1 Introduction

Neural Radiance Fields (NeRF) [30] have demonstrated outstanding success as a versatile and accurate 3D representation of real-world scenes, which has led to its wide adoption in daily and industrial applications in numerous domains. One of the remaining key desiderata for NeRFs is its extension to dynamic scenes.

Existing dynamic NeRF methods can be broadly categorized into two approaches. One is to learn deformation fields for motion warping (*e.g.*, D-NeRF [36]

^{*} denotes corresponding author.

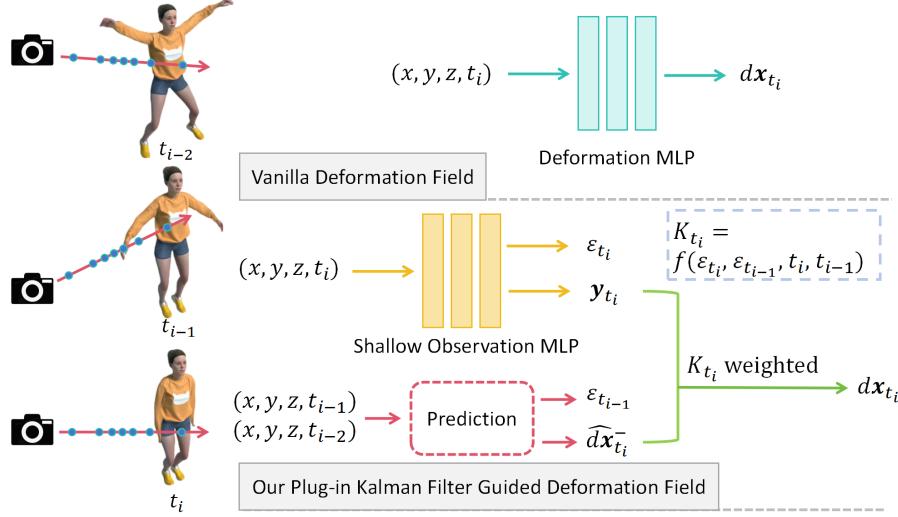


Fig. 1: In contrast to a vanilla deformation field, our plug-in Kalman filter guided deformation field consists of a prediction branch along with the direct observations from input data. From noise related terms ε_{t_i} and $\varepsilon_{t_{i-1}}$ we learn Kalman gain K_{t_i} , weighting observations y_{t_i} and predictions $\hat{dx}_{t_i}^-$ for more accurate deformation estimations.

and TiNeuVox [10]). Another is to disregard motion priors and directly interpolate time in the feature space (*e.g.*, DyNeRF [24] and KPlanes [12]). These approaches, however, often overlook the characteristics of a dynamic radiance field as a time-state sequence, missing the opportunity to fully leverage temporal contextual information.

In this work, we draw inspirations from control theory and model the 4D radiance field as a dynamic system with temporally-varying states. Our state estimates of this dynamic system come from two sources of knowledge: observations based on the input data and predictions based on the system physical dynamics. Optimal state estimates cannot be achieved with only one of these two sources of knowledge. On the one hand, observations, as commonly used in previous dynamic NeRF works, are inherently error-prone due to the discrete temporal sampling of the dynamic scenes. On the other hand, predictions are governed by the correctness of the assumed kinematic model and may struggle to maintain accuracy for real dynamic scenes.

To maximize combined potential of both observations and predictions, we introduce an efficient plug-and-play Kalman filter [43] module to optimize state estimations of our dynamic system. In Fig. 1 we illustrate our plug-in Kalman filter guided deformation field. We model the 4D radiance field as a single-state system, with the state denoted as dx_{t_i} for the current frame deformation. In contrast to a vanilla deformation field that only considers system observations in the current frame, our approach incorporates richer information from previous

frames by introducing a prediction branch based on a motion equation. Given the absence of prior trajectory regarding the scene’s motion, we employ local linear motion.

Both observations and predictions are weighted using a learnable Kalman gain to calculate precise deformation estimations. During the initial stages of training, predictions primarily influence the process, facilitating the convergence of frames with substantial motion. In the later stages of training, observations take precedence, allowing for the recovery of more precise and fine-grained motion details.

All points in the real space are warped to a time-independent canonical space according to the estimated deformations. To further improve the performance of our two-branch deformable model, we employ an efficient tri-plane spatial representation for encoding our canonical space. Experimental evidence shows that this representation permits a shallower observation MLP with only two layers in our implementation. At the same time, we improve the warping ability of the observation MLP by regularizing the learning of the radiance field in the canonical space.

In summary, our contributions are as follows:

- 1) The first method for modeling 4D radiance fields as dynamic systems by integrating a Kalman filter into the deformation field formulation, which results in a plug-in, efficient method for estimating deformations;
- 2) KFD-NeRF, a novel deformable NeRF with the Kalman filter plug-in and a tri-plane spatial representation, trained with a novel strategy of gradually releasing temporal information to facilitate learning dynamic systems;
- 3) and regularization in the canonical space for enhancing the learning capacity of a shallow observation MLP. We achieve state-of-the-art results on both synthetic and real data with all these designs, compared to dynamic NeRFs.

2 Related Works

Neural Radiance Fields Representation. NeRF [30] represents 3D scenes based on an implicit representation encoded by an MLP. Given multi-view images and corresponding camera poses as input, the MLP is trained by “analysis by synthesis,” achieving novel view synthesis and coarse geometry reconstruction. Vanilla NeRF leaves room for improvement that has attracted abundant research.

Many methods [13, 20, 26, 39, 48] use sparse voxel grids to represent the 3D scenes. While grid-based modeling achieves fast training and rendering, fine details require high-resolution grids leading to excessive memory consumption. The optimization process for grids is also unstable, failing to leverage the smoothness bias brought by MLP. Instant-NGP [31] proposes an efficient hash encoding for mapping spatial and feature domains. For sparsely-observed regions, however, this one-to-many mapping can easily introduce noise. The tri-plane representation [5, 6, 35] significantly reduces the memory footprint by leveraging its low-rank decomposition. We use this representation to build a fine-detailed canonical space.

Dynamic Neural Rendering. For dynamic scenarios, modeling the time dimension together with the spatial representation becomes the main challenge. One approach to dynamic NeRF [4, 12, 14, 15, 21, 24, 25, 27, 34, 38, 45] is to add timestamps as an extra dimension to the 3D space and formulate 4D interpolation. Though intuitive, this leads to temporal incoherence due to the lack of supervision and priors between frames. Another approach [7, 9, 10, 17, 18, 25, 32, 33, 36, 40, 49] is to model the motion with deformation fields that warp points in arbitrary frames to the canonical space conditioned on the timestamps. This design can take advantage of the MLP smoothness bias in accordance with the motion smoothness prior. Nevertheless, deformation fields fail to find correspondences in frames with significant motion. We emphasize the performance improvement resulting from the switch in backbone spatial representation from D-NeRF [36] (MLP-based) to TiNeuVox [10] (voxel grids based) and later analyze the impact of different spatial representations on the deformation field. We also compare with latest point-based 4D rendering works [29, 44, 46] inspired by 3D Gaussian Splatting [23].

Deep Kalman Filter. The combination of deep learning and Kalman filtering has been explored to address the challenges of incomplete observations in various scenarios. Some works [28, 52] use Convolutional Neural Network (CNN)-based Kalman filtering for video compression and camera localization. Recurrent Neural Network (RNN)-based Kalman filter has also been used in some studies [8, 37] to improve state estimation and to optimize pose regularization. The Transformer in [50] enables a more comprehensive exploitation of temporal contextual information. All of these works along with [3, 11, 19], however, model dynamics only in the learned latent space without taking into account physical priors, with the exception of [16] which fuses known physical priors and network outputs to construct a Kalman filter for video prediction. In this paper, for the first time, we employ a neural Kalman filter to assist in the dynamic NeRF task. We derive a two-stream method consisting of a shallow MLP and physical priors to achieve efficient motion estimation.

3 Preliminaries

3.1 NeRF and Volume Rendering Revisited

NeRF [30] consists of three parts: sampling, volume mapping, and rendering. In the sampling process, points $\mathbf{x} \in \mathbb{R}^3$ are sampled along rays calculated from the camera pose. Then, in the volume mapping process, each \mathbf{x} in the 3D world, as well as its viewing direction $\mathbf{v} \in \mathbb{R}^3$, is queried to output the volume density σ and radiance $\mathbf{c} = (r, g, b)$ of \mathbf{x} . Finally, in the rendering process, the color of each ray is computed with volume rendering [22]. The expected color of the ray $\mathbf{r}(s) = \mathbf{o} + s\mathbf{v}$ becomes [30]

$$C(\mathbf{r}) = \int_{s_n}^{s_f} T(s)\sigma(\mathbf{r}(s))\mathbf{c}(\mathbf{r}(s), \mathbf{v})ds, \quad (1)$$

where s_n and s_f are the near and far bounds, respectively, and

$$T(s) = \exp \left(- \int_{s_n}^s \sigma(\mathbf{r}(k)) dk \right). \quad (2)$$

The only supervision for training NeRF comes from the ground truth color $C_{gt}(\mathbf{r})$ of each ray

$$\mathcal{L}_{image} = \sum_{\mathbf{r} \in \mathcal{R}} \|C(\mathbf{r}) - C_{gt}(\mathbf{r})\|_2^2, \quad (3)$$

where \mathcal{R} is the ray batch.

3.2 Kalman Filter

Consider a dynamic system with input u_t , output y_t , process noise n_t and measurement noise m_t . We want to obtain its state x_t at each frame t , assuming the state equation follows

$$x_t = Ax_{t-1} + Bu_t + n_t, \quad (4)$$

and the output equation follows

$$y_t = Cx_t + m_t, \quad (5)$$

where A , B and C are the system matrix, control matrix and output matrix, respectively.

There are two methods for obtaining the system's state x_t at any given t : observation and prediction. The observation method tries to rely on y_t in Eq. (5) while the prediction method tries to mathematically model the system dynamics to predict the state. Both of these methods, however, come with non-negligible errors due to the existence of m_t and n_t and for inaccurate system modeling.

The Kalman filter algorithm posits that the state x_t is obtained by a weighted sum of the prediction based on the state x_{t-1} and the observation y_t . For a specific t , estimating the state x_t can be divided into two steps: prediction step and update step. Assuming that the process noise and measurement noise follow zero-mean Gaussian distributions with variances Q and R , respectively, at the prediction step, it first calculates a predicted state based on the estimated state at $t-1$

$$\hat{x}_t^- = A\hat{x}_{t-1} + Bu_t, \quad (6)$$

and its error covariance

$$P_t^- = AP_{t-1}A^T + Q. \quad (7)$$

In the update step, this predicted state is combined with observation y_t to obtain updated state estimate

$$\hat{x}_t = \hat{x}_t^- + K_t(y_t - C\hat{x}_t^-), \quad (8)$$

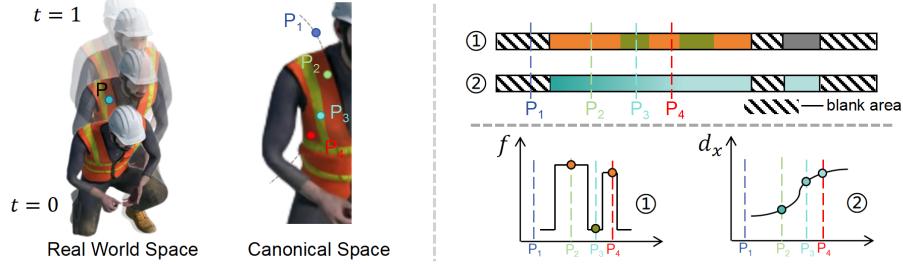


Fig. 2: Visualization of feature planes learned by the feature interpolation method ① and the deformation fields method ②. We show a point P in the real world space and its corresponding four points P_1, P_2, P_3 and P_4 in the canonical space at four different timestamps. The feature plane in ② exhibits better smoothness compared to ①, so we use ② to construct our backbone.

and its error covariance

$$P_t = (I - K_t C) P_t^-, \quad (9)$$

where

$$K_t = \frac{P_t^- C^T}{C P_t^- C^T + R} \quad (10)$$

is the Kalman gain, reflecting the weights assigned to the observation and prediction components.

4 Method

Figure 3 illustrates the three stages of the complete pipeline of KFD-NeRF. In this section, we will first analyze the advantages of using deformation fields as the motion representation over feature interpolation. We will then introduce KFD-NeRF based on the Kalman filter to achieve accurate deformation estimations. Finally, we will discuss spatial reconstruction details, the training strategy, and the incorporation of regularization.

4.1 Motion Representation: Deformation Fields *vs.* Feature Interpolation

4D NeRFs generally employ two approaches to temporal modeling: motion deformation fields or time-conditioned feature interpolation. A deformation field $D(\mathbf{x}, t) \rightarrow \Delta\mathbf{x}$ calculates the deformation $\Delta\mathbf{x}$ at each timestamp t , which can be used to warp a 3D point at t to its corresponding position in the canonical space. In contrast, time-conditioned feature interpolation $F(\mathbf{x}, t) \rightarrow \mathbf{f}$ directly learns a feature vector \mathbf{f} from a given 3D point at t , which is sent to a decoder for RGB and density calculation.

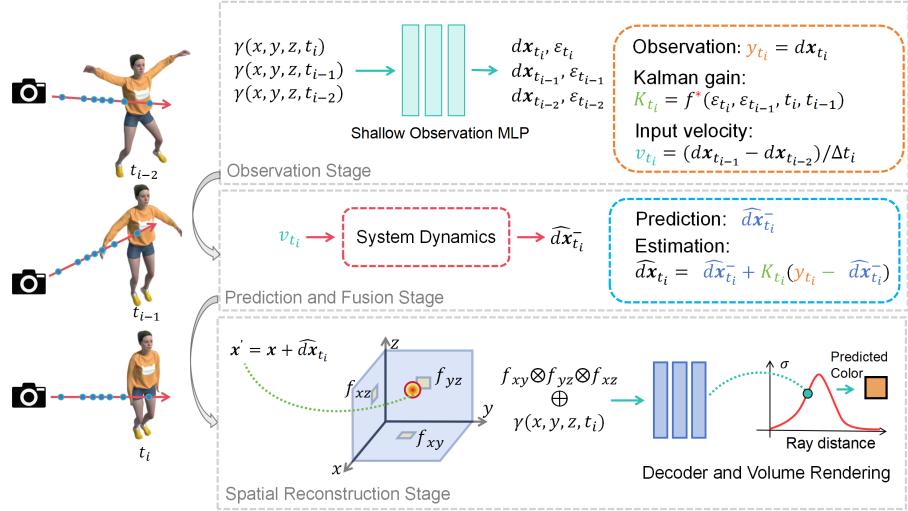


Fig. 3: The overall pipeline of our KFD-NeRF. Our designed deformation field calculates final deformations based on two sources of knowledge. At the observation stage, system observations are directly output by a shallow observation MLP. At the prediction and fusion stage, we calculate predictions based on system dynamics, and further fuse observations and predictions to obtain final deformation estimations. At the spatial reconstruction stage, tri-plane encoded canonical points are concatenated with raw positions and timestamps, which are further decoded to obtain predicted colors for loss calculation. (* f represents a linear layer, used to obtain the Kalman gain from noise-related terms ε_{t_i} and $\varepsilon_{t_{i-1}}$.)

In Fig. 2, we illustrate the difference between these two methods with a motion example from time $t = 0$ to time $t = 1$. Given a 3D point P in the real world space, we can identify its corresponding four points in the canonical space at four different timestamps, named P_1 , P_2 , P_3 , and P_4 . ① shows the feature to be learned at P which changes over time. Notice that when the radiance undergoes abrupt changes, the feature space exhibits many high-frequency signals, which are hard to fit. ② shows the backward deformation from the real world space to the canonical space to be learned at P which changes over time. These signals are smoother and easier to fit by leveraging the motion’s smoothness prior. MLP representations have inductive smoothness bias and are well-suited for learning such deformations. Hence, we employ MLP-based deformation fields to represent motion in this work.

4.2 Kalman Filter Guided Motion Prediction

Next, we derive the system equations for the dynamic radiance field following Eqs. (4) and (5). As we are trying to model non-rigid motions, each 3D point in the dynamic radiance field could be independently seen as a dynamic system, with its velocity v_{t_i} at each frame t_i as a single input u_t . Also note that

this dynamic system is a single-state system since we only focus on estimating deformation at each frame t_i , denoted as $d\mathbf{x}_{t_i}$.

In the next step, we determine the coefficients matrix A , B , and C in Eqs. (4) and (5), each degenerating to a value under a single-input and single-state system. As we directly observe $d\mathbf{x}_{t_i}$, the output matrix C becomes 1. Since the motion trajectory is unknown, we use a locally-linear motion model to describe this dynamic system: $d\mathbf{x}_{t_i} = d\mathbf{x}_{t_{i-1}} + \Delta t_i \cdot \mathbf{v}_{t_i}$ ($A = 1$ and $B = \Delta t_i$). Further considering noise, our state and output equations become

$$\begin{cases} d\mathbf{x}_{t_i} = d\mathbf{x}_{t_{i-1}} + \Delta t_i \cdot \mathbf{v}_{t_i} + n_{t_i}, \\ y_{t_i} = d\mathbf{x}_{t_i} + m_{t_i}. \end{cases} \quad (11)$$

In Fig. 3, our designed deformation field consists of two parts: an MLP-based observer and a system-dynamics-based predictor. As for the observer, by querying 3D points coordinates \mathbf{x} and timestamps t_i , a two-layer shallow MLP is used to output the mean observation y_{t_i} (assuming zero-mean Gaussian measurement noise m_{t_i}) and noise-related terms ε_{t_i} . For the predictor, we follow the locally-linear motion model and first calculate the input \mathbf{v}_{t_i} . Because the deformation of the current frame t_i is being estimated, we only use the information from frame t_{i-1} and frame t_{i-2} to approximate the velocity

$$\mathbf{v}_{t_i} = (d\mathbf{x}_{t_{i-1}} - d\mathbf{x}_{t_{i-2}}) / \Delta t_i. \quad (12)$$

Based on the estimated velocity, we compute the predicted deformation state $\hat{d}\mathbf{x}_{t_i}^-$. The final estimation $\hat{d}\mathbf{x}_{t_i}$ is the sum of $\hat{d}\mathbf{x}_{t_i}^-$ and y_{t_i} weighted by a Kalman gain K_{t_i} . The Kalman gain in Eq. (10) is calculated by current measurement noise and past process noise. In our implementation, we obtain K_{t_i} utilizing a linear layer that takes noise-related terms ε_{t_i} and $\varepsilon_{t_{i-1}}$ and timestamps t_i and t_{i-1} as inputs, incorporating historical information. Algorithm 1 summarizes this deformation estimation.

Volume is warped to the canonical space based on the estimated deformation, which is further decoded to obtain density σ and color. We compute the loss by comparing the rendered values with the ground truth and use it to update the network, following Eq. (3). In addition, we note that the estimated deformation serves as a good supervision for learning the observation MLP, in analogy with observation updating process in Kalman filter. Therefore, we try to minimize the error between the current observation and the estimated deformation with

$$\mathcal{L}_{kf} = \frac{1}{N} \sum_{\mathbf{x} \in \mathcal{N}} \|y_{t_i} - \hat{d}\mathbf{x}_{t_i}^-\|_2^2. \quad (13)$$

The weighted y_{t_i} and $\hat{d}\mathbf{x}_{t_i}^-$ stand for two sources of knowledge from the whole system. y_{t_i} represents the state directly observed by the observation MLP, which lacks the information from past frames and thus has a low confidence in the early stages of training. In contrast, $\hat{d}\mathbf{x}_{t_i}^-$ represents the current frame information predicted from history, which provides a good prior in the early stages of training.

Algorithm 1 Deformation Estimations

Prediction:	Update:
$d\mathbf{x}_{t_{i-1}}, \varepsilon_{t_{i-1}} = \text{observer}(\mathbf{x}, t_{i-1})$	$y_{t_i}, \varepsilon_{t_i} = \text{observer}(\mathbf{x}, t_i)$
$d\mathbf{x}_{t_{i-2}}, \varepsilon_{t_{i-2}} = \text{observer}(\mathbf{x}, t_{i-2})$	$K_{t_i} = f(\varepsilon_{t_i}, \varepsilon_{t_{i-1}}, t_i, t_{i-1})$
$\mathbf{v}_{t_i} = (d\mathbf{x}_{t_{i-1}} - d\mathbf{x}_{t_{i-2}}) / \Delta t_i$	$\hat{d}\mathbf{x}_{t_i} = \hat{d}\mathbf{x}_{t_i}^- + K_{t_i}(y_{t_i} - \hat{d}\mathbf{x}_{t_i}^-)$
$\hat{d}\mathbf{x}_{t_i}^- = d\mathbf{x}_{t_{i-1}} + \Delta t_i \cdot \mathbf{v}_{t_i} = 2d\mathbf{x}_{t_{i-1}} - d\mathbf{x}_{t_{i-2}}$	

In the later stages of training, however, the confidence of the predictor drops significantly due to the inherent errors of the modeled motion dynamics, which is gradually compensated by the well learned observation MLP. Our Kalman Filter guided model automatically strikes the balance of both leveraging $\hat{d}\mathbf{x}_{t_i}^-$ in the early stages to accelerate convergence and avoid local minimum and also using y_{t_i} in the later stages to learn fine details.

4.3 Spatial Representation

MLP-based spatial representations suffer from slow convergence and require a deeper deformation network (*e.g.*, eight layers MLP in [36]) to model 4D scenes. Otherwise, the deformation field can get stuck in a local optimum before learning the radiance field in the canonical space. Voxel grids based spatial representations converge very quickly (*e.g.*, TiNeuVox [10]) but demand high spatial resolution to store fine details, which requires significant memory footprint. Under memory constraints, achieving high resolution in the canonical space can be challenging, resulting in losing scene details and deteriorating the warping quality for each frame. We employ a tri-plane representation, which uses three sets of mutually orthogonal 2D planes to represent 3D space. This low-rank model allows for rapid convergence while significantly reducing memory consumption, making it possible to achieve fast and high-quality canonical space reconstruction.

What's more, there are some inevitable errors due to coordinate shifts when \mathbf{x}' are encoded by finite resolution tri-plane. To enhance the raw coordinate information, we follow [10] by concatenating encoded tri-plane features with raw coordinate inputs.

4.4 Training Strategy and Regularization

Our model takes into account temporal information so the learning of the current frame partially relies on the results of previous frames. To ensure that previous frames can offer ample priors, we employ a training strategy of gradually releasing the training images in chronological order.

We also notice that the lack of constraints or priors in the canonical space can affect the performance of the observation branch, which could be regularized by pre-setting the shape in the canonical space. D-NeRF [36] sets frame t_0 to be the canonical space and force the deformation output of frame t_0 to be masked to 0. Such a hard mask, however, does not allow learning of the deformation

at frame t_0 and may cause the canonical space to be too complex for motions warping.

We instead design a soft regularization term to normalize the difference between the canonical space and the radiance field in the real world at frame t_0 to improve the observation branch. The canonical observation loss of a points batch \mathcal{N} is

$$\mathcal{L}_{co} = \frac{1}{|\mathcal{N}|} \sum_{\mathbf{x} \in \mathcal{N}} \mathbf{1}(t) d\mathbf{x}, \quad (14)$$

where $\mathbf{1}(t) = 1$ when $t = 0$ and $\mathbf{1}(t) = 0$ when $t \neq 0$.

We use a proposal sampling strategy from Mip-NeRF 360 [2] by distilling the density field for occupancy estimation. This online distillation necessitates a loss function \mathcal{L}_{prop} to ensure consistency between the proposal network and our learned model. Please refer to Section 3 in [2] for detailed definition.

Total variation loss \mathcal{L}_{tv} is a common regularizer in inverse problems, which encourages sparse edges in space. We apply this loss to each of our tri-plane to get $\mathcal{L}_{tv}(\mathbf{x}')$, where \mathbf{x}' are warped 3D points in the canonical space. The total variation loss is

$$\mathcal{L}_{tv}(\mathbf{x}) = \frac{1}{|C|} \sum_{c,i,j} (\left\| \mathbf{x}_c^{i,j} - \mathbf{x}_c^{i-1,j} \right\|_2^2 + \left\| \mathbf{x}_c^{i,j} - \mathbf{x}_c^{i,j-1} \right\|_2^2), \quad (15)$$

where c is a certain plane from the tri-plane collection C and i, j are indices on the plane's resolution.

The final loss function becomes

$$\mathcal{L} = \mathcal{L}_{image} + \mathcal{L}_{kf} + \mathcal{L}_{co} + \mathcal{L}_{prop} + \lambda_{tv} \mathcal{L}_{tv}, \quad (16)$$

and we experimentally choose λ_{tv} to be 1×10^{-4} .

5 Experimental Results

5.1 Dataset

For synthetic data, we use the Dynamic NeRF Synthetic Dataset provided by D-NeRF [36], whose training and testing splits have already been well-organized. For each scene in the synthetic dataset, a photo from an arbitrary view with corresponding camera pose is provided at each timestamp. For real data, we use the Nvidia Real Dynamic Scenes Dataset [47] which consists of 8 dynamic scenes recorded by 12 synchronized cameras. We use 11 camera videos for training and the remaining one for testing.

5.2 Baselines and Metrics

Due to differences in the format of synthetic and real data, we carefully select cutting-edge baselines to thoroughly validate our method based on comparative experiments. For synthetic data, we test deformation based methods

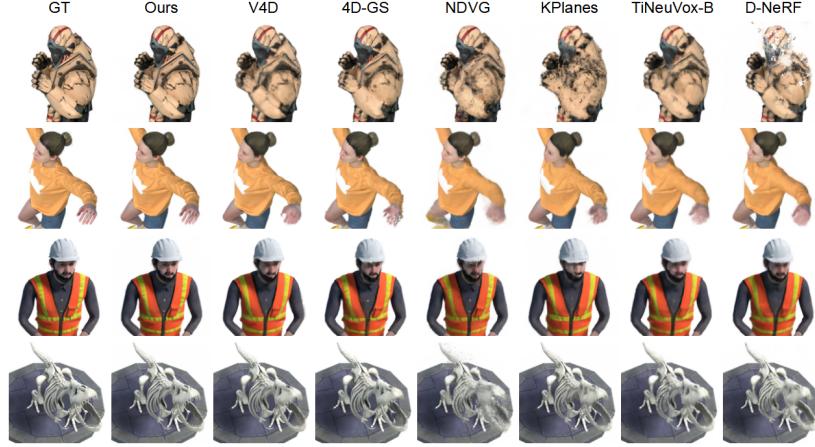


Fig. 4: Qualitative Comparison of our KFD-NeRF against other dynamic NeRF methods on synthetic data. Zoom in for better details.

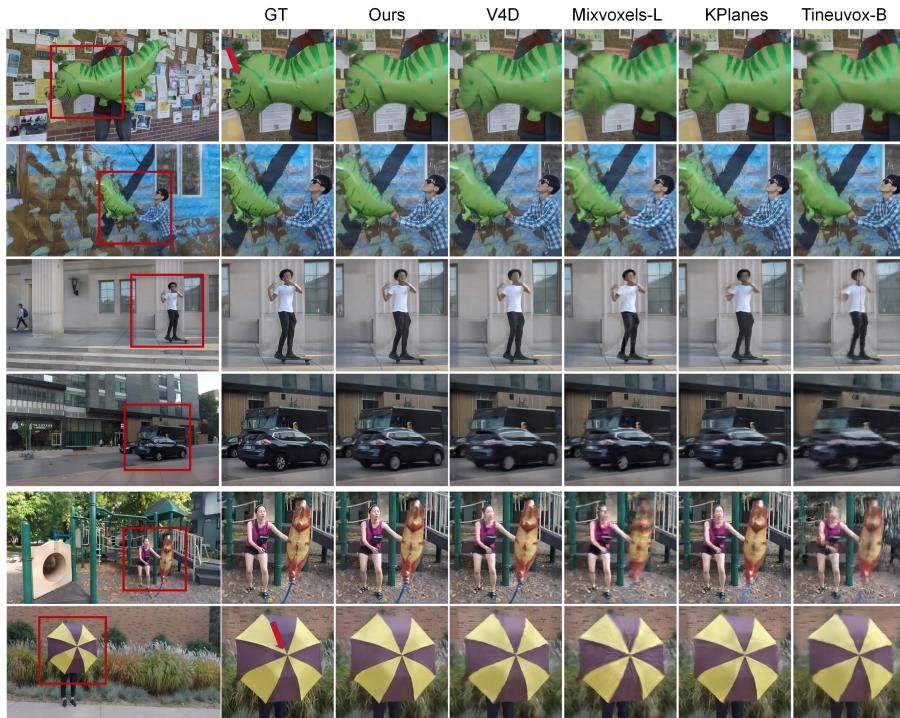


Fig. 5: Qualitative Comparison of our KFD-NeRF against other dynamic NeRF methods on real data. Zoom in for better details.

Model	Synthetic Data				Average↓
	PSNR(dB)↑	SSIM↑	LPIPS↓	Average↓	
D-NeRF [36]	30.25	0.954	0.076	0.0288	
TiNeuVox-B [10]	32.93	0.973	0.043	0.0173	
KPlanes [12]	30.47	0.963	0.072	0.0261	
NDVG [17]	30.51	0.964	0.056	0.0223	
V4D [14]	33.54	0.978	0.028	0.0141	
4D-GS [44]	33.31	0.979	0.026	0.0136	
KFD-NeRF	35.73	0.984	0.045	0.0131	

Model	Real Data				Average↓
	PSNR(dB)↑	SSIM↑	LPIPS↓	Average↓	
TiNeuVox-B [10]	24.39	0.734	0.303	0.0869	
KPlanes [12]	28.25	0.881	0.137	0.0433	
Mixvoxels-L [41]	26.95	0.828	0.222	0.0619	
V4D [14]	28.16	0.884	0.136	0.0426	
KFD-NeRF	28.75	0.891	0.115	0.0389	

Table 1: Quantitative comparison of our KFD-NeRF against other dynamic NeRF methods. We show results on both synthetic and real data. See Sec. 5.2 for more details.

	D-NeRF	TiNeuVox-B	KPlanes	NDVG	V4D	4D-GS	Mixvoxels-L	KFD-NeRF-S	KFD-NeRF-L
Training Time↓	48hrs	40mins	60mins	35mins	7hrs	25mins	1.5hrs	30mins	3hrs
Params(MB)↓	16	50	340	150	275	163	140	175	175

Table 2: Training time and parameter size for different methods.

D-NeRF [36] (MLP based spatial representation), TiNeuVox-B [10] (voxel grids based spatial representation), NDVG [17] (voxel grids based spatial representation) and 4D-GS [44] (Gaussian points based spatial representation), and feature interpolation based methods KPlanes [12] and V4D [14]. For real data, besides TiNeuVox-B and KPlanes, we further compare multi-view videos reconstruction methods MixVoxels [41]. We train all these methods on a single GeForce RTX3090. See Tab. 2 for detailed training time and parameters consumption of ours and other methods.

We provide an exhaustive qualitative and quantitative comparison of our KFD-NeRF with these baseline methods. Three main metrics are reported, namely the peak signal-to-noise ratio (PSNR), the structural similarity index measure (SSIM) [42], and the learned perceptual image patch similarity (LPIPS) [51]. To provide more intuitive results, we further calculate metric “average” [1], which is the geometric mean of $\text{MSE} = 10^{-\frac{\text{PSNR}}{10}}$, $\sqrt{1 - \text{SSIM}}$, and LPIPS. Please see Tab. 1 for quantitative comparison and Figs. 4 and 5 for qualitative comparison. Per-scene results can be found in the supplemental material. We strongly recommend readers to watch the supplemental videos for a more intuitive understanding of the results.

5.3 Ablation Studies

We conduct ablation studies on both synthetic and real data to validate our various proposed system components. We compare our full model with variants related to \mathcal{L}_{co} , \mathcal{L}_{kf} and prediction branch.

Canonical observation loss \mathcal{L}_{co} . This loss is designed to regularize a continuous and smooth volume shape in the canonical space for better observations. Specifically, we guide the shape in the canonical space to be close to the shape



Fig. 6: Two sources of knowledge at frame t_i from our pre-trained dynamic system with inputs up to and including frame t_{i-1} . Notice that during the early stages of training, our observation MLP has insufficient initialization, resulting in the loss of many details (green boxes on the right). Fortunately, our predictions provide additional priors (green boxes on the left) for the current frame based on information from previous frames, compensating for the loss in observations.

	\mathcal{L}_{kf}	\mathcal{L}_{co}	PSNR(dB)↑	SSIM↑	LPIPS↓
a)			30.15	0.914	0.110
b)	✓		30.60	0.920	0.106
c)		✓	31.78	0.931	0.086
d)	✓	✓	32.24	0.938	0.080

Table 3: Ablation Studies on the \mathcal{L}_{kf} and \mathcal{L}_{co} . We show all the ablation combinations and report the average results on synthetic and real data.

Model	Dynamic		Static	
	PSNR(dB)↑	SSIM↑	PSNR(dB)↑	SSIM↑
P.N., (\mathbf{x}_i, t_i) Input	25.61	0.963	29.58	0.812
P.N., $(\mathbf{x}_i, t_i, t_{i-1}, t_{i-2})$ Input	25.66	0.964	29.55	0.812
w/ Prediction Branch (full)	26.71	0.967	29.57	0.813

Table 4: The ablation study on our prediction branch for real data in average, where dynamic and static areas are counted separately. P.N. is short for Pure MLP Network.

at frame t_0 . In Tab. 3 line *b*), we remove \mathcal{L}_{co} and observe a decrease in model performance.

Estimation update loss \mathcal{L}_{kf} . This loss tries to minimize the difference between estimated deformation and current observation. This updating process ensures that with each iteration, the observation acquire progressively more accurate information to guide the learning process. In Tab. 3 line *c*), we remove \mathcal{L}_{kf} and witness a decrease in model performance.

5.4 Extra Studies on Prediction Branch

This study aims to demonstrate the effectiveness of our plug-in Kalman filter. The most crucial step in Kalman filter is to fuse the original network observations with the predictions from system dynamics. Therefore, we directly remove the ‘‘Prediction and Fusion Stage’’ in our pipeline and use pure shallow observation MLP to generate deformations. We further conduct an experiment where a pure deformable network takes the current frame and the previous two states as inputs to ablate the effectiveness of the Kalman motion modeling. In Tab. 4, we see a clear drop in performance by pure MLP network-based methods, indicating that simply inputting previous Kalman filter parameters without modeling motion prediction is insufficient.

The prediction branch we have designed can offer reliable priors in the early stages of training. We visualize these priors through an experiment to gain further insight. Specifically, we train our system with only inputs up to and including frame t_{i-1} and then let our system produce rendering results at frame t_i with two different branches: predictions and observations. This experiment



Fig. 7: An example in Nvidia Dataset [47] to illustrate scale and topological issues in choosing canonical space. From ① to ② there is a scale change due to moving closer to the camera. From ③ to ② there is a topological change caused by target disappearance. According to this, choosing canonical space ① will suffer from low resolution and choosing canonical space ③ will suffer from incomplete topology while ② is a better canonical space to choose.

effectively simulates the amount of knowledge contained in predictions and observations in the early stages of training when the system encounters new input data. In Fig. 6 we show two sources of knowledge from prediction branch and from direct output of observation MLP at frame t_i . We see an obvious deficiency in the observation MLP when initializing new input frames and are pleased to find that the prediction branch compensates for this loss based on information from previous frames.

6 Discussion and Conclusion

Limitations. Our method relies on a well-reconstructed radiance field in the canonical space, which in our pipeline is guided by \mathcal{L}_{co} . This design, however, will partially fail, if the chosen canonical space exhibits significant scale changes or even topological changes related to other frames. In Fig. 7 we use an example to illustrate the influences caused by choosing canonical space. This phenomenon, however, cannot be mitigated by precisely setting the canonical space as we lack a priori access to the input data. We note that some works [1,33] focus on addressing scale or topological issues in radiance fields reconstruction. Nevertheless, these issues are not the main focus of our paper and will be explored in future works to further refine our model.

Conclusion. In this work, we present KFD-NeRF, a Kalman filter guided NeRF, for 4D dynamic view synthesis. We model the dynamic radiance field as a dynamic system in control theory and use Kalman filter to estimate the deformation states based on both observations and predictions. We further enhance our observation by encoding canonical space with an efficient tri-plane and by regularizing the shape in the canonical space. Through our temporal training strategy and newly derived pipeline, KFD-NeRF achieves state-of-the-art view synthesis performance among a variety of dynamic NeRF methods. We hope the dynamic system modeling of 4D radiance fields will encourage researchers to explore motion contextual information. KFD-NeRF hopefully inspires the use of existing sequential methods mainly in control theory and visual state estimation to further improve 4D view synthesis and deformation estimations tasks.

Acknowledgements

This research was supported in part by JSPS KAKENHI Grant Numbers 24K22318, 22H00529, 20H05951, JST-Mirai Program JPMJMI23G1.

References

1. Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-NeRF: A Multiscale Representation for Anti-aliasing Neural Radiance Fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5855–5864 (2021)
2. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-NeRF 360: Unbounded Anti-aliased Neural Radiance Fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5470–5479 (2022)
3. Becker, P., Pandya, H., Gebhardt, G., Zhao, C., Taylor, C.J., Neumann, G.: Recurrent Kalman Networks: Factorized Inference in High-dimensional Deep Feature Spaces. In: International conference on machine learning. pp. 544–552. PMLR (2019)
4. Cao, A., Johnson, J.: Hexplane: A Fast Representation for Dynamic Scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 130–141 (2023)
5. Chan, E.R., Lin, C.Z., Chan, M.A., Nagano, K., Pan, B., De Mello, S., Gallo, O., Guibas, L.J., Tremblay, J., Khamis, S., et al.: Efficient Geometry-aware 3D Generative Adversarial Networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16123–16133 (2022)
6. Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H.: Tensorf: Tensorial Radiance Fields. In: European Conference on Computer Vision. pp. 333–350. Springer (2022)
7. Choe, J., Choy, C., Park, J., Kweon, I.S., Anandkumar, A.: Spacetime Surface Regularization for Neural Dynamic Scene Reconstruction. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 17871–17881 (2023)
8. Coskun, H., Achilles, F., DiPietro, R., Navab, N., Tombari, F.: Long Short-term Memory Kalman Filters: Recurrent Neural Estimators for Pose Regularization. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 5524–5532 (2017)
9. Du, Y., Zhang, Y., Yu, H.X., Tenenbaum, J.B., Wu, J.: Neural Radiance Flow for 4D View Synthesis and Video Processing. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 14304–14314. IEEE Computer Society (2021)
10. Fang, J., Yi, T., Wang, X., Xie, L., Zhang, X., Liu, W., Nießner, M., Tian, Q.: Fast Dynamic Radiance Fields with Time-aware Neural Voxels. In: SIGGRAPH Asia 2022 Conference Papers. pp. 1–9 (2022)
11. Fraccaro, M., Kamronn, S., Paquet, U., Winther, O.: A Disentangled Recognition and Nonlinear Dynamics Model for Unsupervised Learning. Advances in neural information processing systems **30** (2017)
12. Fridovich-Keil, S., Meanti, G., Warburg, F.R., Recht, B., Kanazawa, A.: K-planes: Explicit Radiance Fields in Space, Time, and Appearance. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12479–12488 (2023)

13. Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxtels: Radiance Fields without Neural Networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5501–5510 (2022)
14. Gan, W., Xu, H., Huang, Y., Chen, S., Yokoya, N.: V4D: Voxel for 4D Novel View Synthesis. IEEE Transactions on Visualization and Computer Graphics (2023)
15. Gao, C., Saraf, A., Kopf, J., Huang, J.B.: Dynamic View Synthesis from Dynamic Monocular Video. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5712–5721 (2021)
16. Guen, V.L., Thome, N.: Disentangling Physical Dynamics from Unknown Factors for Unsupervised Video Prediction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11474–11484 (2020)
17. Guo, X., Chen, G., Dai, Y., Ye, X., Sun, J., Tan, X., Ding, E.: Neural Deformable Voxel Grid for Fast Optimization of Dynamic View Synthesis. In: Proceedings of the Asian Conference on Computer Vision. pp. 3757–3775 (2022)
18. Guo, X., Sun, J., Dai, Y., Chen, G., Ye, X., Tan, X., Ding, E., Zhang, Y., Wang, J.: Forward Flow for Novel View Synthesis of Dynamic Scenes. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 16022–16033 (2023)
19. Haarnoja, T., Ajay, A., Levine, S., Abbeel, P.: Backprop Kf: Learning Discriminative Deterministic State Estimators. Advances in neural information processing systems **29** (2016)
20. Hu, T., Liu, S., Chen, Y., Shen, T., Jia, J.: EfficientNeRF: Efficient Neural Radiance Fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12902–12911 (2022)
21. Jang, H., Kim, D.: D-tensorf: Tensorial Radiance Fields for Dynamic Scenes. arXiv preprint arXiv:2212.02375 (2022)
22. Kajiya, J.T., Von Herzen, B.P.: Ray Tracing Volume Densities. ACM SIGGRAPH computer graphics **18**(3), 165–174 (1984)
23. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3D Gaussian Splatting for Real-time Radiance Field Rendering. ACM Transactions on Graphics (ToG) **42**(4), 1–14 (2023)
24. 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: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5521–5531 (2022)
25. Li, Z., Niklaus, S., Snavely, N., Wang, O.: Neural Scene Flow Fields for Space-time View Synthesis of Dynamic Scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6498–6508 (2021)
26. 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)
27. Liu, Y.L., Gao, C., Meuleman, A., Tseng, H.Y., Saraf, A., Kim, C., Chuang, Y.Y., Kopf, J., Huang, J.B.: Robust Dynamic Radiance Fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13–23 (2023)
28. Lu, G., Ouyang, W., Xu, D., Zhang, X., Gao, Z., Sun, M.T.: Deep Kalman Filtering Network for Video Compression Artifact Reduction. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 568–584 (2018)
29. Luiten, J., Kopanas, G., Leibe, B., Ramanan, D.: Dynamic 3d Gaussians: Tracking by Persistent Dynamic View Synthesis. arXiv preprint arXiv:2308.09713 (2023)

30. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. *Communications of the ACM* **65**(1), 99–106 (2021)
31. Müller, T., Evans, A., Schied, C., Keller, A.: Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Transactions on Graphics (ToG)* **41**(4), 1–15 (2022)
32. Park, K., Sinha, U., Barron, J.T., Bouaziz, S., Goldman, D.B., Seitz, S.M., Martin-Brualla, R.: Nerfies: Deformable Neural Radiance Fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5865–5874 (2021)
33. Park, K., Sinha, U., Hedman, P., Barron, J.T., Bouaziz, S., Goldman, D.B., Martin-Brualla, R., Seitz, S.M.: Hypernerf: A Higher-dimensional Representation for Topologically Varying Neural Radiance Fields. *arXiv preprint arXiv:2106.13228* (2021)
34. Park, S., Son, M., Jang, S., Ahn, Y.C., Kim, J.Y., Kang, N.: Temporal Interpolation Is All You Need for Dynamic Neural Radiance Fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4212–4221 (2023)
35. Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., Geiger, A.: Convolutional Occupancy Networks. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16. pp. 523–540. Springer (2020)
36. Pumarola, A., Corona, E., Pons-Moll, G., Moreno-Noguer, F.: D-neRF: Neural Radiance Fields for Dynamic Scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10318–10327 (2021)
37. Revach, G., Shlezinger, N., Ni, X., Escoriza, A.L., Van Sloun, R.J., Eldar, Y.C.: KalmanNet: Neural Network Aided Kalman Filtering for Partially Known Dynamics. *IEEE Transactions on Signal Processing* **70**, 1532–1547 (2022)
38. Shao, R., Zheng, Z., Tu, H., Liu, B., Zhang, H., Liu, Y.: Tensor4d: Efficient Neural 4D Decomposition for High-fidelity Dynamic Reconstruction and Rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16632–16642 (2023)
39. Sun, C., Sun, M., Chen, H.T.: Direct Voxel Grid Optimization: Super-fast CONvergence for Radiance Fields Reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5459–5469 (2022)
40. Treitschke, E., Tewari, A., Golyanik, V., Zollhöfer, M., Lassner, C., Theobalt, C.: Non-rigid Neural Radiance Fields: Reconstruction and Novel View Synthesis of a Dynamic Scene from Monocular Video. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 12959–12970 (2021)
41. Wang, F., Tan, S., Li, X., Tian, Z., Song, Y., Liu, H.: Mixed Neural Voxels for Fast Multi-view Video Synthesis. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 19706–19716 (2023)
42. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image Quality Assessment: from Error Visibility to Structural Similarity. *IEEE transactions on image processing* **13**(4), 600–612 (2004)
43. Welch, G., Bishop, G., et al.: An Introduction to the Kalman Filter (1995)
44. Wu, G., Yi, T., Fang, J., Xie, L., Zhang, X., Wei, W., Liu, W., Tian, Q., Wang, X.: 4D Gaussian Splatting for Real-Time Dynamic Scene Rendering. *arXiv preprint arXiv:2310.08528* (2023)
45. Xian, W., Huang, J.B., Kopf, J., Kim, C.: Space-time Neural Irradiance Fields for Free-viewpoint Video. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9421–9431 (2021)

46. Yang, Z., Gao, X., Zhou, W., Jiao, S., Zhang, Y., Jin, X.: Deformable 3D Gaussians for High-Fidelity Monocular Dynamic Scene Reconstruction. arXiv preprint arXiv:2309.13101 (2023)
47. 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: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5336–5345 (2020)
48. Yu, A., Li, R., Tancik, M., Li, H., Ng, R., Kanazawa, A.: Plenoctrees for Real-time Rendering of Neural Radiance Fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5752–5761 (2021)
49. Yuan, W., Lv, Z., Schmidt, T., Lovegrove, S.: Star: Self-supervised Tracking and Reconstruction of Rigid Objects in Motion with Neural Rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13144–13152 (2021)
50. Zhang, H., Yang, Z., Xiong, H., Zhu, T., Long, Z., Wu, W.: Transformer Aided Adaptive Extended Kalman Filter for Autonomous Vehicle Mass Estimation. Processes **11**(3), 887 (2023)
51. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 586–595 (2018)
52. Zhou, L., Luo, Z., Shen, T., Zhang, J., Zhen, M., Yao, Y., Fang, T., Quan, L.: Kfnet: Learning Temporal Camera Relocalization Using Kalman Filtering. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4919–4928 (2020)

KFD-NeRF: Rethinking Dynamic NeRF with Kalman Filter

Supplemental Material

Yifan Zhan¹, Zhuoxiao Li¹, Muyao Niu¹, Zhihang Zhong³, Shohei Nobuhara², Ko Nishino², and Yinqiang Zheng^{*1}

¹ The University of Tokyo

² Kyoto University

³ Shanghai Artificial Intelligence Laboratory

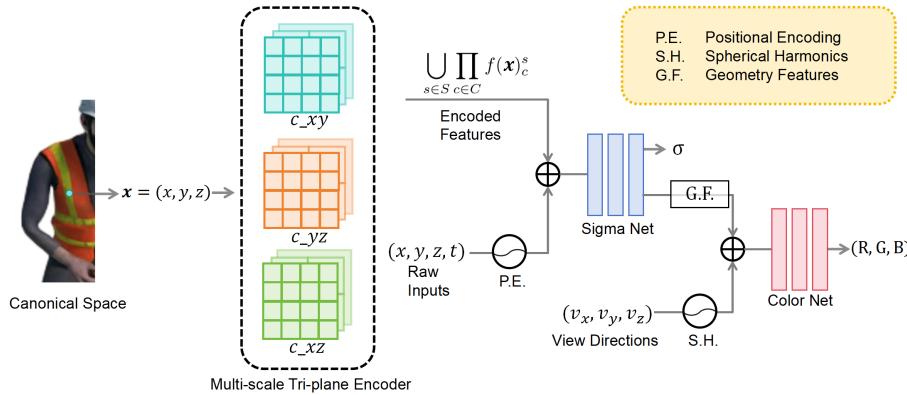


Fig. 1: Details of our spatial neural architectures.

1 Per-scene Results

We exhibit our per-scene results for synthetic data in Tab. 1 and for real data in Tab. 2. In Fig. 2 and Fig. 3 we additionally provide more visualization results on synthetic and real data.

2 Spatial Neural Architectures

We elaborate in detail on our spatial neural architectures in Fig. 1. Once obtaining the warped 3D points in the canonical space, we first use a multi-scale tri-plane to encode the spatial information. For each plane $c \in C$ and each scale $s \in S$, features are multiplied across planes and concatenated across scales to obtain tri-plane encoded features. However, coordinate shifts happen due to the limited resolution of the tri-plane grids and errors introduced by linear interpolation. We follow [?] by concatenating encoded tri-plane features with positional

Method	Bouncing Balls				Hell Warrior				Hook				Jumping Jacks			
	PSNR↑	SSIM↑	LPIPS↓	Average↓	PSNR↑	SSIM↑	LPIPS↓	Average↓	PSNR↑	SSIM↑	LPIPS↓	Average↓	PSNR↑	SSIM↑	LPIPS↓	Average↓
D-NeRF [?]	38.05	0.982	0.107	0.018	24.94	0.948	0.071	0.038	29.43	0.962	0.122	0.032	31.96	0.972	0.045	0.019
TiNeuVox-B [?]	40.53	0.990	0.036	0.007	28.21	0.965	0.072	0.027	32.31	0.975	0.044	0.016	34.70	0.983	0.033	0.012
KPlanes [?]	41.12	0.991	0.032	0.006	25.65	0.952	0.079	0.036	28.55	0.957	0.082	0.029	32.60	0.977	0.054	0.017
NDVG [?]	34.59	0.969	0.113	0.019	25.58	0.949	0.075	0.037	29.74	0.966	0.040	0.021	29.55	0.960	0.081	0.027
V4D [?]	42.00	0.992	0.029	0.005	27.06	0.960	0.054	0.028	30.95	0.972	0.037	0.018	35.29	0.984	0.022	0.010
4D-GS [?]	39.18	0.990	0.033	0.007	27.53	0.968	0.042	0.024	32.13	0.978	0.022	0.013	34.71	0.984	0.022	0.010
KFD-NeRF	42.16	0.991	0.038	0.006	30.55	0.978	0.051	0.019	35.65	0.990	0.034	0.010	35.64	0.988	0.041	0.011
Method	Logo				Mutant				Stand Up				T-Rex			
	PSNR↑	SSIM↑	LPIPS↓	Average↓	PSNR↑	SSIM↑	LPIPS↓	Average↓	PSNR↑	SSIM↑	LPIPS↓	Average↓	PSNR↑	SSIM↑	LPIPS↓	Average↓
D-NeRF [?]	21.70	0.842	0.169	0.077	31.19	0.973	0.029	0.016	33.41	0.980	0.023	0.012	31.30	0.972	0.043	0.018
TiNeuVox-B [?]	25.17	0.924	0.075	0.040	33.76	0.979	0.031	0.013	36.03	0.985	0.021	0.009	32.74	0.979	0.033	0.014
KPlanes [?]	25.52	0.948	0.059	0.034	24.71	0.917	0.178	0.057	33.89	0.983	0.051	0.014	31.71	0.981	0.038	0.016
NDVG [?]	25.21	0.933	0.052	0.034	35.44	0.988	0.015	0.008	34.01	0.982	0.023	0.011	30.00	0.967	0.048	0.021
V4D [?]	25.63	0.948	0.038	0.029	36.14	0.989	0.014	0.007	37.06	0.990	0.012	0.006	34.21	0.987	0.018	0.010
4D-GS [?]	25.37	0.940	0.044	0.031	37.80	0.993	0.009	0.005	36.82	0.990	0.011	0.007	32.97	0.984	0.022	0.012
KFD-NeRF	25.54	0.948	0.070	0.035	39.23	0.995	0.039	0.007	39.62	0.994	0.030	0.007	37.40	0.992	0.054	0.010

Table 1: Per-scene quantitative comparisons on synthetic dynamic scenes.

Method	Balloon1				Balloon2				dynamicFace				Jumping			
	PSNR↑	SSIM↑	LPIPS↓	Average↓												
TiNeuVox-B [?]	25.21	0.773	0.249	0.071	26.35	0.814	0.208	0.059	22.30	0.887	0.167	0.069	25.41	0.779	0.329	0.077
KPlanes [?]	28.20	0.887	0.100	0.037	26.85	0.863	0.157	0.044	25.44	0.923	0.112	0.045	27.09	0.857	0.206	0.058
Mixvoxels-L [?]	26.24	0.808	0.235	0.063	26.78	0.811	0.235	0.060	20.03	0.792	0.308	0.112	26.91	0.855	0.230	0.059
V4D [?]	27.11	0.888	0.101	0.040	24.55	0.847	0.148	0.059	27.20	0.951	0.083	0.033	27.78	0.883	0.175	0.049
KFD-NeRF	28.83	0.906	0.076	0.031	27.30	0.888	0.089	0.038	26.45	0.937	0.092	0.037	26.93	0.845	0.215	0.061
Method	Playground				Skating				Truck				Umbrella			
	PSNR↑	SSIM↑	LPIPS↓	Average↓												
TiNeuVox-B [?]	16.60	0.376	0.461	0.200	27.93	0.840	0.276	0.056	25.78	0.765	0.356	0.077	25.56	0.636	0.381	0.086
KPlanes [?]	24.59	0.836	0.152	0.060	34.06	0.956	0.092	0.020	32.93	0.925	0.118	0.025	26.84	0.801	0.159	0.057
Mixvoxels-L [?]	23.19	0.748	0.253	0.085	33.14	0.945	0.148	0.026	32.60	0.920	0.133	0.027	26.70	0.748	0.236	0.063
V4D [?]	25.69	0.875	0.100	0.046	33.79	0.956	0.115	0.022	32.95	0.930	0.126	0.026	26.24	0.742	0.237	0.066
KFD-NeRF	24.59	0.846	0.129	0.056	34.95	0.964	0.077	0.017	33.44	0.930	0.100	0.023	27.49	0.815	0.146	0.048

Table 2: Per-scene quantitative comparisons on real dynamic scenes.

Model	PSNR(dB)↑	SSIM↑	LPIPS↓	Training Time↓	Params(MB)↓
MLP Based	32.13	0.971	0.052	43hrs	0.5
Voxel Grid Based	36.21	0.984	0.029	60mins	2336
Tri-plane Based	38.92	0.990	0.037	150mins	31.5

Table 3: Ablation Study on Spatial Representations. We compare three different spatial models on rendering quality, training time and spatial-only size. The tri-plane model we choose significantly reduces the model size compared to the voxel grid based model and achieves the highest rendering quality in a relatively short training time.

encoded [?] raw inputs. Our Sigma Net (single-hidden-layer MLP) outputs volume density σ and 15-dimensional geometry features. The geometry features will be further concatenated with spherical harmonics encoded view directions for color calculation using Color Net (two-hidden-layer MLP).

3 Model Hyperparameters

The frequency number of positional encoding is set to 5 for both spatial and temporal inputs. Our shallow observation MLP consists of two hidden layers,

each has a channel dimension of 128. As for the tri-plane grids, we use multi-scale planes with 4 different resolutions at 64^2 , 128^2 , 256^2 and 512^2 . As can be seen in Fig. 1, the per-plane and per-scale features are multiplied across planes and concatenated across scales. We set each of these per-plane and per-scale features’ dimension to be 32 so the final encoded feature has dimension 128. Sigma Net has single hidden layer with channel dimension 64 and Color Net has two hidden layers with channel dimension 64.

For optimization, an Adam optimizer [?] is used and we set ray batch to be 4096 in each iteration. We train our KFD-NeRF with learning rate set to 1×10^{-3} .

4 Ablation Study on Spatial Representations

In Section 4.3, we analyze the characteristics of different spatial representations and their reconstruction capabilities in the canonical space. Based on our full model, we further conduct ablation studies by only changing spatial representations to show their impacts on reconstruction results.

Specifically, we compare three different spatial models, namely, pure MLP (8 hidden layers with dimension 256), multi-scale voxel grid (resolutions at 64^3 , 128^3 and 256^3) and multi-scale tri-plane (resolutions at 64^2 , 128^2 and 256^2). In Tab. 3, We compare these models based on three dimensions: model size, convergence time, and rendering quality.

Our full model only uses a shallow MLP with two hidden layers to calculate deformations, so the pure spatial MLP would suffer from convergence difficulties. The voxel grid based model converges faster while the model size is far from acceptable. Therefore we choose the tri-plane based model as a spatial model that converges relatively fast, has a moderate size, and provides high rendering quality.



Fig. 2: More qualitative results on synthetic data. Zoom in for better details.

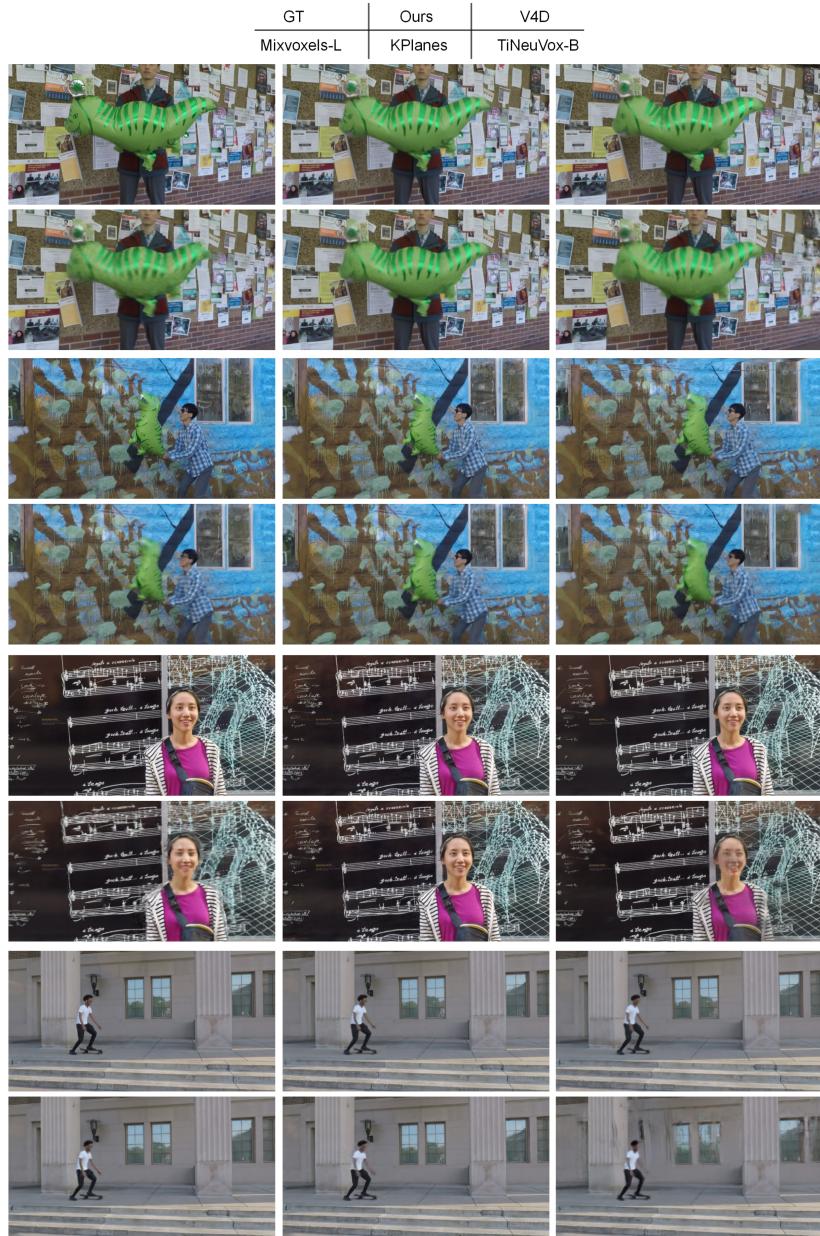


Fig. 3: More qualitative results on real data. Zoom in for better details.