PAPR in Motion: Seamless Point-level 3D Scene Interpolation

Shichong Peng, Yanshu Zhang, Ke Li
APEX Lab
School of Computing Science
Simon Fraser University
{shichong_peng,yanshu_zhang,keli}@sfu.ca

Abstract

We propose the problem of point-level 3D scene interpolation, which aims to simultaneously reconstruct a 3D scene in two states from multiple views, synthesize smooth point-level interpolations between them, and render the scene from novel viewpoints, all without any supervision between the states. The primary challenge is on achieving a smooth transition between states that may involve significant and non-rigid changes. To address these challenges, we introduce "PAPR in Motion", a novel approach that builds upon the recent Proximity Attention Point Rendering (PAPR) technique, which can deform a point cloud to match a significantly different shape and render a visually coherent scene even after non-rigid deformations. Our approach is specifically designed to maintain the temporal consistency of the geometric structure by introducing various regularization techniques for PAPR. The result is a method that can effectively bridge large scene changes and produce visually coherent and temporally smooth interpolations in both geometry and appearance. Evaluation across diverse motion types demonstrates that "PAPR in Motion" outperforms the leading neural renderer for dynamic scenes. For more results and code, please visit our project website at https://niopeng.github.io/PAPR-in-Motion/.

1. Introduction

In computer vision and graphics, it is commonly desired to visualize how a scene might evolve smoothly from a state to a very different state. If this can be done, it could open the way to many possibilities. For instance, in animation and filmmaking, the ability to create lifelike animations from just two observed states could greatly simplify production processes and reduce the time and effort required compared to traditional frame-by-frame animation.

To tackle this problem, we introduce a novel task, *point-level 3D scene interpolation*. Given observations of a 3D scene at only two states from multiple views, our objec-tive is to synthesize a smooth and plausible interpolation between scene points at the different states, so that the scene can be rendered from novel views at any intermediate state, as shown in Figure 1. No observation of any intermediate state is assumed. Our task differs from traditional static scene reconstruction [15, 16, 25, 41, 44, 49], which does not handle scene changes, and also from novel view synthesis of dynamic 3D scenes [23, 28–30], which assumes the observation of the scene at all times as it changes.

Our task carries unique challenges beyond those of novel view synthesis on static or dynamic scenes. Because intermediate states are not observed, the scene geometry can change substantially between the observed states. The geometry changes can be highly complex and involve different local deformations that may be non-rigid. Additionally, these geometry changes can lead to appearance changes, for example in the cast shadows and specular reflections.

To tackle this task, we need to be able to represent the large geometry changes between the states accurately. We can view such changes as motion. Different representations of motion are suitable for capturing different types of scene changes. Eulerian motion representations record changes at fixed scene locations, while Lagrangian motion representations track specific points through space and time. Unlike Eulerian representations, which estimate changes independently at different time steps, Lagrangian representations jointly estimate changes for the same point tracked across time. As a result, Lagrangian representations can model long-range dynamics more accurately. Since there may be substantial geometry changes between the observed states in our task, we opt for modelling scene changes with the Lagrangian approach. To enable this, we leverage recent advances in point-based neural renderers, which are well-suited to representing Lagrangian motion through the point-based scene representations they learn. At a high level, our method involves first training a model on the scene's initial state and then finetuning it to reconstruct the scene in its final state. The intermediate iterations of this finetuning process effectively create interpolations between the two states.
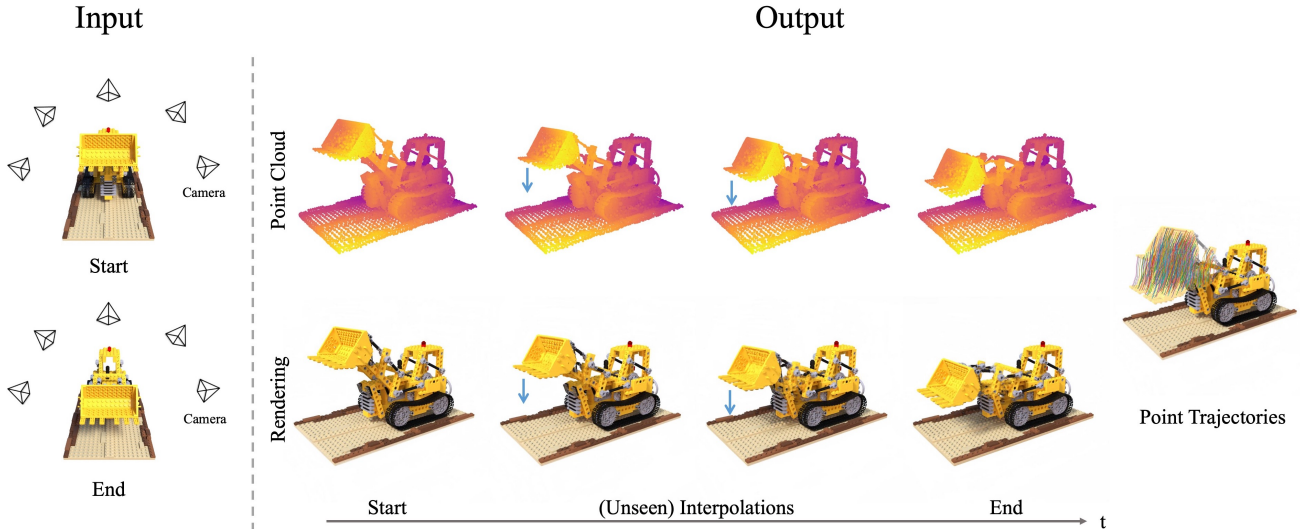
Input | **Output**

Figure 1. We introduce the novel task of point-level 3D scene interpolation: Given multi-view RGB images of a scene at two distinct states, the start state and the end state, our goal is to achieve a seamless point-level interpolation between these states without any intermediate supervision. Our method, PAPR in Motion, can effectively synthesize a plausible geometry for any intermediate state, represented by a point cloud (top row), and renders the corresponding scene appearance (bottom row). Additionally, we provide a visualization of the point interpolation trajectories (right), which illustrate the coherent motion synthesized within the scene.

To ensure seamless and realistic interpolations, the choice of point rendering technique is crucial. We leverage the Proximity Attention Point Rendering (PAPR) [48] technique, which offers two key advantages: (i) its ability to deform a point cloud to match a significantly different shape, crucial for adapting to substantial scene changes; and (ii) its ability to maintain shape contiguity and visual coherence after geometry modifications such as non-rigid deformations, ensuring realistic novel view synthesis of the scene at all interpolated states.

Our method, which we dub "PAPR in Motion", harnesses the strengths of PAPR and incorporates regularization techniques specifically designed for our problem. We leverage the observation that nearby points tend to move together and design regularizers for PAPR that encourage local rigidity and uniform point motion. We show that our method produces smooth and realistic interpolations in both geometry and appearance across diverse and complex scene changes, encompassing synthetic and real-world scenes. PAPR in Motion consistently outperforms the leading neural renderer for dynamic scenes, Dynamic Gaussian [24], on point-level 3D scene interpolation. These findings not only validate the effectiveness of our approach in tackling this challenging new task but also establish a new baseline. In summary, our contributions are as follows:

1. We introduce the novel task of point-level 3D scene interpolation between two distinct scene states.
2. We propose PAPR in Motion, a point-based approach capable of synthesizing plausible and smooth 3D scene interpolations without any supervision in between.
3. We evaluate our approach on a diverse set of complex scene changes, and show that it significantly outperforms the leading neural renderer for dynamic scenes, Dynamic Gaussian [24], on this task.

2. Related Work

Our work introduces the novel task of point-level 3D scene interpolation, a task for which specific methods have not yet been established. However, in a broader context, our work intersects with and draws insights from related areas such as dynamic scene novel view synthesis and motion analysis and synthesis of 3D objects.

Novel View Synthesis for Dynamic Scenes The introduction of neural radiance fields (NeRF) [25] marked a significant milestone in novel view synthesis, subsequently giving rise to extensive research focused on adapting NeRF for dynamic scenes. Some studies have proposed fitting a separate model for each frame [3, 40], while others have utilized Eulerian representations on a 4D space-time grid to model the scene, optimizing efficiency through techniques like planar decomposition or hash functions [4, 8, 34]. Alternatively, some methods define the representation in a canonical timestep and use deformation fields to map this reference to other frames [6, 19, 23, 28–30, 32, 35, 43]. Other research has also explored the use of pre-defined templates or skeleton tracking to guide the deformation pro-

cess [13, 17, 38]. While effective in certain scenarios, these approaches rely on prior knowledge on the reconstructed objects, limiting their applicability to general scenes.

Another line of work in the modelling of dynamic scenes has adopted point-based representations [2, 24, 46]. Recently, the Dynamic Gaussian method [24], which builds on the 3D Gaussian Splatting point-based representation [15], has showcased impressive capabilities in long-term point tracking and novel view synthesis. By treating points as particles, it effectively creates a Lagrangian representation that naturally follows points through space and time.

However, a common assumption in these methods, is the availability of uninterrupted observation of the scene at all time steps. In contrast, our method is designed to function without requiring constant observation, and can address scenarios where only discrete snapshots of the scene at different states are available.

3D Object Motion Analysis and Synthesis Unlike the works mentioned earlier and our work, these methods only estimate the change in object geometry, but they cannot produce the appearance of the object as it moves. The analysis and synthesis of 3D object motion represents a long-standing area of research with considerable advances over time [1, 9, 10, 21, 26]. The rise of data-driven methodologies has further expanded this domain, with numerous methods [7, 11, 14, 22, 36, 42, 45] emerging to learn object motions in a supervised framework. A common limitation among these methods, however, is their reliance on 3D supervision and sometimes also on articulation annotations, which are challenging and costly to obtain at scale. Another line of research has focused on category-specific modelling, where separate models are developed for distinct object categories [18, 27, 33, 37]. While effective within their defined scope, these methods often face difficulties in generalizing to arbitrary novel objects.

In a related context, recent work by Liu et al. [20] investigates using pairs of RGB observations to analyze the motion parameters of an articulated object. Their approach, however, only focuses on part-level rigid motion of a single movable part. In contrast, our approach models point-level scene changes, accommodating both rigid and non-rigid transformations, and handles scenarios where multiple parts of an object move simultaneously.

3. Problem Setup

We introduce the novel task of point-level 3D scene interpolation. In this task, we consider a scene undergoing changes within a 3D space, with our inputs limited to observations from only two distinct scene states: the initial state at time $t = 0$ and the final state at time $t = T$. Our objective is to generate a smooth and plausible interpolation at the point

level between these two states, without observing any intermediate states, as illustrated in Figure 1. This means for each point i in the scene, which moves from its initial location p_i^0 at the start state to its final location p_i^T at the end state, we aim to generate its trajectory $\{p_i^t\}_{t \in (0,T)}$ through time and space. Moreover, the synthesized images $\hat{\mathbf{I}}_t$ of the scene at any given time step $t \in [0, T]$ should also appear realistic, thereby yielding a visually coherent interpolation between the initial and final states.

We are given a set of RGB images of a 3D scene at two states from multiple views along with their associated camera poses. It is important to note the distinction between our problem setting and static 3D scene reconstruction [15, 16, 25, 41, 44, 49], which does not consider changes in the scene over time. It also differs from the classic problem of novel view synthesis of dynamic 3D scenes [23, 28–30], where all time steps between start and end states are typically observed. Our focus is on synthesizing plausible interpolations between scenes at two distinct states, rather than on reconstruction or tracking of observed motion across time. Our problem setting is inherently quite challenging due to the potential for significant scene changes between the two observed states.

4. Method

To achieve point-level 3D scene interpolation between two states, our approach starts with training a point-based neural rendering model on the initial state, followed by finetuning this model to match the final state. The intermediate iterations of this finetuning process generate the interpolations between the initial and final states. The rest of this section starts by discussing the rationale behind our choice of the point-based rendering technique we build upon, namely Proximity Attention Point Rendering (PAPR) [48] (Section 4.1). This is followed by a brief overview of PAPR (Section 4.2). Finally, we delineate our method, "PAPR in Motion", which leverages the capabilities of PAPR to generate scene interpolations (Section 4.3).

4.1. Choice of Point-based Rendering Technique

Our method leverages point-based representations to model long-range scene dynamics in an Lagrangian fashion. Given the challenges posed by large scene changes and non-rigid transformations, selecting an appropriate point-based renderer is critical.

Point-based neural renderers come in two flavours: splat-based renderers and attention-based renderers. Splat-based renderers [15, 31, 39] place primitives, e.g., ellipsoids or Gaussian kernels, at every point and check which primitives intersect with a given ray. On the other hand, attention-based renderers, like PAPR [48], directly predict the intersection point between the surface and a given ray by interpolating between nearby points using attention.

In our problem setting, since there may be large scene changes between the observed states, the initial and final point clouds should differ substantially. We opt for PAPR over splat-based renderers because the latter struggle with large scene changes. This limitation of splat-based renderers arises from the decreasing contribution of each point to a pixel as it moves away from the pixel, leading to vanishing gradients for distant points. Consequently, splat-based renderers can become stuck in point cloud configurations far from the ground truth, particularly when the initial point cloud differs substantially from it. In contrast, instead of solely considering the absolute distance of each point to the pixel, PAPR takes the relative distances of different points to the pixel into account. This is achieved through an attention mechanism, which normalizes the contributions of all points, ensuring their total contribution always sums to 1. As a result, there will always be points that have significant contributions to each pixel, even for pixels far from all points. This prevents the gradient from vanishing even when the point cloud is far from the ground truth geometry, thereby allowing PAPR to move points at ease regardless of where they were initially.

Furthermore, because there can be complex, non-rigid geometry changes between observed states, synthesizing realistic views after applying such deformations becomes crucial. Again, PAPR outperforms splat-based renderers in this aspect. Splat-based renderers often produce porous shapes after non-rigid transformations, as shown in [48]. This phenomenon stems from the need to redistribute splats to maintain surface coverage after non-rigid deformations like stretching. In contrast, PAPR, by implicitly representing surfaces through point interpolation, can adapt to changes in point density, preserving shape contiguity and visually coherent appearances under non-rigid deformations.

4.2. Overview of Proximity Attention Point Rendering (PAPR)

PAPR takes a set of RGB images of a static 3D scene from multiple views and their associated camera poses as input, and jointly learns a point-based scene representation, an attention mechanism, and a differentiable renderer based on U-Net architecture. Each point i in the scene representation consists of a point location $\mathbf{p}_i \in \mathbb{R}^3$, an influence score $\tau_i \in \mathbb{R}$ and a view-independent feature vector $\mathbf{u}_i \in \mathbb{R}^d$.

Given a camera pose, PAPR casts a ray from the camera's centre of projection through each pixel on the image plane. At each ray, PAPR first produces a ray-dependent embedding for each point. It then uses the attention mechanism to compute the relative contribution weights of these points, and combines the feature vectors of these points using the output weights. The aggregated feature map is then fed to the differentiable renderer to produce the output image $\hat{\mathbf{I}}$. The entire pipeline is trained end-to-end to minimize

reconstruction loss between the output $\hat{\mathbf{I}}$ and ground truth image $\mathbf{I}_{gt}$, as measured by the function $\mathcal{L}_{recon}$:

$$\mathcal{L}_{recon} = \text{MSE}(\hat{\mathbf{I}}, \mathbf{I}_{gt}) + \lambda \cdot \text{LPIPS}(\hat{\mathbf{I}}, \mathbf{I}_{gt}) \qquad (1)$$

Here, the reconstruction loss is composed of a weighted combination of mean squared error (MSE) and the LPIPS metric [47]. For further details and in-depth analysis of PAPR, we direct readers to the original paper [48].

4.3. PAPR in Motion

We will assume that the start and end states differ in geometry, but are similar in appearance. However, because the ground truth geometries are not known at any state, we must estimate them. To do so, we will build an scene template that can support different geometries and fit the same template to both states by varying the geometry parameters. To build the scene template, we will train a PAPR model of the scene at the start state. Because PAPR is adept at handling geometry deformations, we can use the PAPR model with all parameters fixed except for the point coordinates as an scene template. In this template, the point coordinates are the free parameters that represent geometry. On the other hand, the view-independent feature vectors associated with the points carry localized scene texture information, and represent the appearance. So each point can viewed as a geometry-agnostic marker of a part that can moved freely to match the observed state of the scene.

We can then fit the scene template to the end state to recover the change in geometry from the start to the end state. During the fitting process, the geometry will change gradually from the start to the end state, and so will form a natural interpolation trajectory. In more concrete terms, this entails finetuning the PAPR model trained earlier on the images from the end state, with all parameters except for the point coordinates fixed. As the spatial positions of the points are changed during the finetuning process, we will keep track of the trajectories they follow and adjust them as needed to ensure their physical plausibility. These trajectories form a Lagrangian representation of the scene's evolution.

To generate a point-level 3D scene interpolation, we use these trajectories to obtain the geometry of the scene at the intermediate states. Because the appearance between the observed states may change as a result of the geometry change, e.g., cast shadows or specular reflection, after the scene template's geometry has been fitted to the end state, we finetune the appearance parameters of our model as well. We then interpolate between the appearance parameters of the models at the start and end states to obtain the appearance of the scene at intermediate states. To synthesize a novel view at any state, we render the geometry and appearance parameters at that state with PAPR.

Below we delineate how we make interpolation trajectories physically plausible and handle appearance changes.
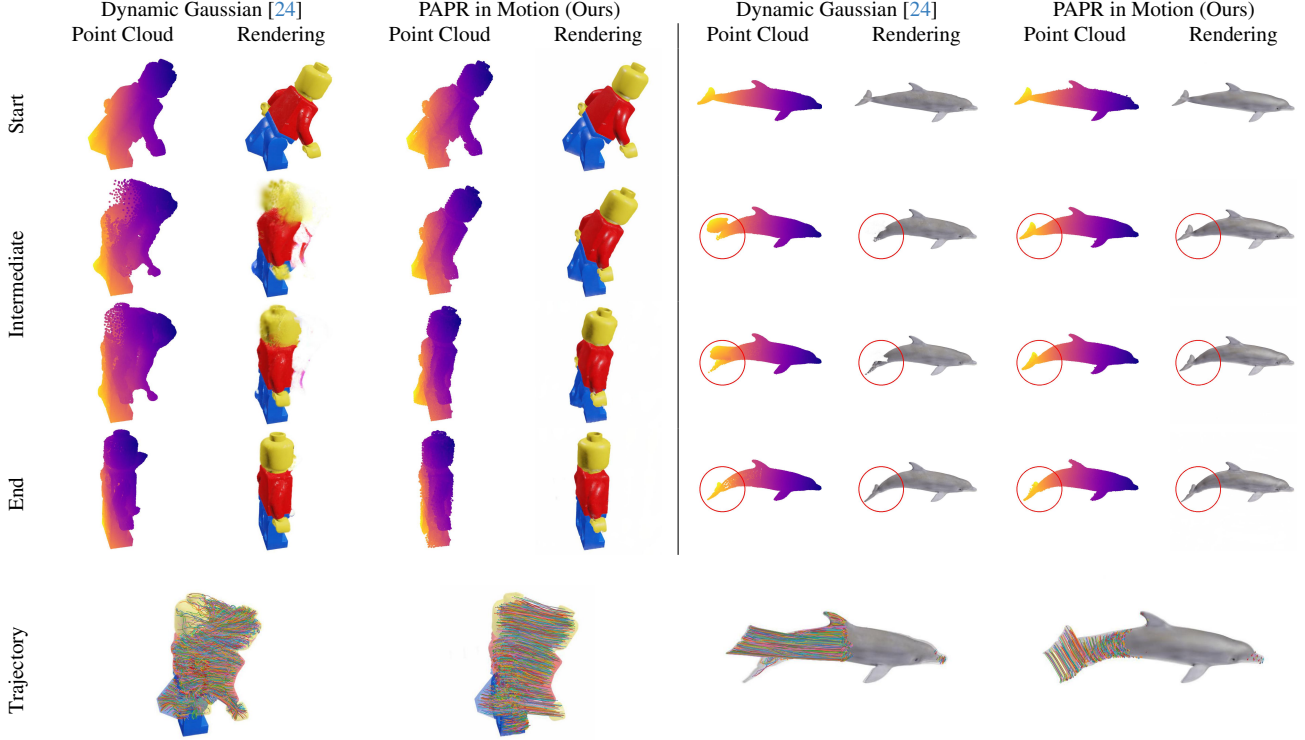
Figure 2. Qualitative comparison of 3D scene interpolation from start to end state using synthetic scenes. Both methods start by training a static model for the start state and subsequently finetune it towards the end state, all without any intermediate supervision. Dynamic Gaussian [24] struggles to maintain consistent scene geometry and appearance, resulting in unnatural motion, as shown by the point trajectories. On the other hand, PAPR in Motion creates smooth and plausible interpolations between states. It successfully handles a variety of transformations, from the rigid motion observed in the Lego man's standing pose to the non-rigid, fluid motion of the dolphin's tail.

Ensuring Plausibility of Interpolation Trajectories
Naïvely finetuning the positions of all points independently can lead to implausible deformations to the part geometry or some points becoming trapped in a local optimum, especially in cases with substantial scene changes or in regions with near-constant texture.

To avoid implausible deformations, we propose adding a regularization term, termed the **local distance preservation loss**, during finetuning, which encourages local rigidity by preserving distances between each point and its neighbours.

Specifically, we begin by finding the k-nearest neighbours $\text{NN}_k(i)$ for each point i based on their positions at the start state $\mathbf{p}_i^0$, and keeping track of its squared ℓ_2 distances $d_{i,j}^0$ to each neighbouring point j in $\text{NN}_k(i)$:

$$d_{i,j}^0 = ||\mathbf{p}_i^0 - \mathbf{p}_j^0||_2^2 \qquad (2)$$

At any given time $t \in (0, T]$, we calculate the regularization loss based on the current squared ℓ_2 distances $d_{i,j}^t$ from $\mathbf{p}_i^t$

to its neighbours from the start state:

$$d_{i,j}^t = ||\mathbf{p}_i^t - \mathbf{p}_j^t||_2^2 \qquad (3)$$

$$\mathcal{L}_{rigid} = \frac{1}{kN} \sum_i \sum_{j \in \text{NN}_k(i)} |d_{i,j}^0 - d_{i,j}^t| \qquad (4)$$

where N represents the total number of points in the scene. The updated loss function, incorporating this new regularization term, becomes the following:

$$\mathcal{L} = \mathcal{L}_{recon} + \lambda_{\text{rigid}} \cdot \mathcal{L}_{rigid} \qquad (5)$$

While the integration of a local distance preservation loss helps regulate the movements of points and avoid excessive drift, we find that it's not always sufficient in cases involving substantial scene changes. In such scenarios, it may not fully ensure that points within a moving part move cohesively as a unit. This issue stems from the loss's inability to account for the directions of point motion. To address this and further smooth out point motion, we introduce a **local displacement averaging step**. This step is designed to

encourage uniform motion of nearby points and its neighbours, which is crucial for maintaining the cohesiveness of moving parts and ensuring temporal visual coherence.

In the local displacement averaging step, we modify the position of each point i by first recalculating its displacement from its initial position at the start state $\mathbf{h}_i^t = \mathbf{p}_i^t - \mathbf{p}_i^0$. Then we adjust it based on the average displacement of its k-nearest neighbours at the start state $\mathrm{NN}_k(i)$ from their respective initial positions:

$$\hat{\mathbf{h}}_i^t = \frac{1}{k} \sum_{j \in \mathrm{NN}_k(i)} (\mathbf{p}_j^t - \mathbf{p}_j^0) \tag{6}$$

$$\hat{\mathbf{p}}_i^t = \mathbf{p}_i^0 + \hat{\mathbf{h}}_i^t \tag{7}$$

Here, $\hat{\mathbf{p}}_i^t$ denotes the adjusted position for point i. This averaging regularization is applied every m iterations to gently guide the points towards a more coherent trajectory.

Once the geometry has nearly converged, we disable local displacement averaging to allow the point cloud to more closely match the end state geometry.

The point positions generated throughout this optimization process serve as interpolations for the scene geometry. We then apply temporal smoothing to the point positions by taking the arithmetic mean over a sliding window of size 7.

Handling Appearance Changes In the final phase, with the scene geometry aligned to the geometry of the end state, we shift our focus to refining the appearance parameters of the template. At this stage, the positions of the points are fixed, and we finetune two key elements. First, the feature vectors associated with each point are optimized. This finetuning specifically targets adapting view-independent aspects of scene appearance, for example changes in colour due to variations in shadows cast by moving parts. Second, we refine the ray-dependent attention mechanism. This adjustment is crucial for adapting view-dependent effects, including reflections. The combination of adjusting both the feature vectors and the attention mechanism enables a more precise capture of the nuances in appearance between the start and end states.

To render the appearance for an intermediate state $t \in (0, T)$ with point positions $\{\mathbf{p}_i^t\}_{i=1}^N$, we interpolate both the feature vectors and the model weights associated with the attention mechanism between the original model and the model after end-state finetuning. This interpolation enables gradual and smooth changes in appearance, such as shadows or specular reflections, between the start and end states. The interpolation weight is given by its progress α_t in the interpolation process within the range of $[0, 1]$ as follows:

$$\alpha_t = \frac{\sum_{i=1}^N ||\mathbf{p}_i^t - \mathbf{p}_i^0||_2}{\sum_{i=1}^N ||\mathbf{p}_i^T - \mathbf{p}_i^0||_2} \tag{8}$$

Here, N denotes the total number of points in the point cloud, and $|| \cdot ||_2$ represents the ℓ_2 distance. Essentially, α_t measures the progress of interpolation using the total distance travelled by the points from their initial positions. The farther they have travelled relative to their final distances, the later the checkpoint is on the interpolation timeline.

5. Experiments

To evaluate the effectiveness of our method on point-level 3D scene interpolation, we experiment on both synthetic and real-world scenes. The synthetic scenes include six scenes from a combination of the NeRF Synthetic Dataset [25] and the Objaverse dataset [5]. In addition, we manually captured two real-world scenes. The scenes we use represent a wide range of complex scene changes, encompassing rigid and non-rigid transformations, large displacements, and scenarios with multiple parts in motion.

Given the novelty of this task and the absence of existing methods for the task, we compare our method to the leading point-based method for dynamic scenes, Dynamic Gaussian [24]. To pre-train Dynamic Gaussian for the initial state, we begin with a point cloud constructed by COLMAP, containing between 100,000 to 500,000 points for each scene. In contrast, the pre-training of PAPR for each scene uses a significantly sparser point cloud of just 30,000 points, which are initialized randomly. This approach is feasible due to PAPR's ability in learning point clouds from scratch and its capability to learn parsimonious yet effective representations. Regarding our method's parameters, we set $\lambda = 0.01$, $\lambda_{\mathrm{rigid}} = 5$, $m = 100$ and a k that ranges from 70 to 300, depending on the specific scene.

We evaluate each method qualitatively on both the geometry and the renderings of the scenes. This evaluation spans the entire duration of the interpolation, from the start state to the end state. Additionally, we visualize and compare the interpolation trajectories of points within the scene, providing more insights into the quality of the interpolated motion generated by each method. For the quantitative evaluation, due to the absence of an existing metric for evaluating scene interpolation quality, we introduce a novel metric termed the *scene interpolation metric*.

Scene Interpolation Metric This metric is designed to assess the overall quality of interpolation by evaluating the quality at different intermediate steps and aggregating these individual quality assessments to arrive at an overall score.

Specifically, the evaluation is performed sequentially at each model checkpoint throughout the interpolation process. For the tth checkpoint, we compute its progress α_t using Equation 8. In our evaluations, we select 24 evenly spaced checkpoints that span the interpolation process.

Given the lack of ground truth geometries and appearances for intermediate steps, at each step t, we assess the
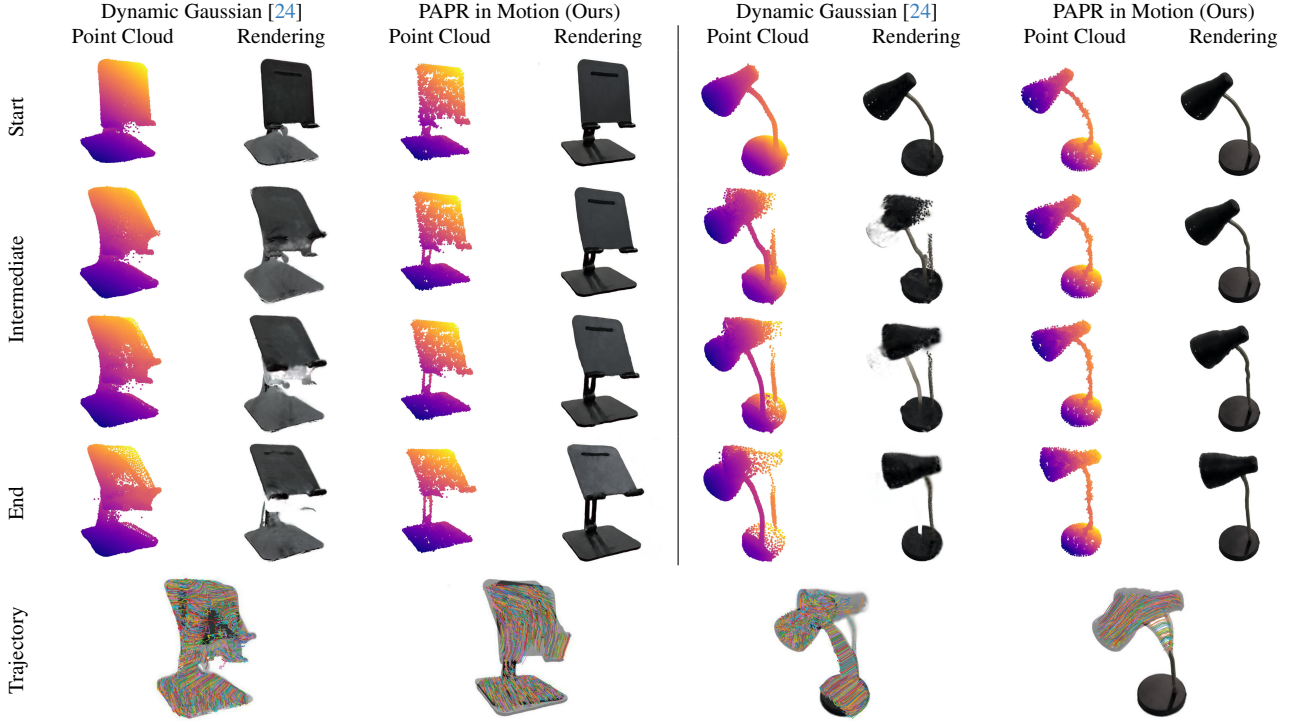
Figure 3. Qualitative comparison of 3D scene interpolation from start to end state using real-world scenes. Both methods start by training a static model for the start state and subsequently finetune it towards the end state, all without any intermediate supervision. Dynamic Gaussian [24] struggles with significant scene changes, as shown by both scenes where only a portion of the points move to the right place. In contrast, PAPR in Motion successfully handles these challenging scenarios, producing smooth and realistic interpolations between states.

quality f_t of interpolation P_t by comparing it separately to the ground truth at the start and end states, P_{start} and P_{end}:

$$f_t = (1 - \alpha_t)d(P_{start}, P_t) + \alpha_t d(P_{end}, P_t)$$

Here, P represents either the geometry (as a point cloud) or the appearance (as a set of scene renderings using camera poses $\mathcal{C}$). For the ground truth point clouds at the start and end states, we use those derived from training the corresponding static 3D scene reconstruction method, namely PAPR [48] for our method and Gaussian Splatting [15] for Dynamic Gaussian [24]. For appearance evaluation, we choose $\mathcal{C}$ to consist of 128 viewpoints encircling the scene in 3D. The distance function $d(\cdot, \cdot)$ denotes either the Chamfer distance (CD) or the Earth Mover's distance (EMD) for geometry evaluation and the Fréchet inception distance (FID) [12] for appearance evaluation.

The overall scene interpolation metric is derived by aggregating the measurements across all steps:

$$\sum_{t=1}^{T}(\alpha_t - \alpha_{t-1}) \cdot \frac{(f_t + f_{t-1})}{2}$$

We use Scene Interpolation CD and EMD (SI-CD, SI-EMD) for geometry evaluation and Scene Interpolation FID (SI-FID) for appearance evaluation.

5.1. Quantitative Results

We include the scene interpolation metric scores in Table 1. As shown, our method significantly outperforms the baseline across all metrics in both synthetic and real-world scenes. These results demonstrate our method's capability to interpolate scene appearance and geometry effectively between the start and end states.

5.2. Qualitative Results

Figures 2 and 3 present a qualitative comparison between our method, PAPR in Motion, and the baseline Dynamic Gaussian [24] on various synthetic and real-world scenes. PAPR in Motion demonstrates superior capabilities in generating plausible and seamless scene interpolations. It effectively handles a diverse range of motion types, including rigid transformations seen in the Lego man and non-rigid bending motions observed in the dolphin in the synthetic scenes (Fig.2). Additionally, our method proves effective in scenes with significant changes, such as raising the tablet stand and the lamp in the real-world scenes (Fig.3). In contrast, the baseline method fails to maintain consistent scene geometry and appearance, resulting in unnatural interpolations. For more qualitative comparisons, please refer to the supplementary materials.

Metric	Method	Synthetic Scenes							Real-world Scenes		
		Butterfly	Crab	Dolphin	Giraffe	Lego Bulldozer	Lego Man	Avg	Stand	Lamp	Avg
SI-FID ↓	Dynamic Gaussian [24]	352.43	106.82	134.01	199.76	173.07	201.74	194.64	293.60	266.87	280.24
	PAPR in Motion (Ours)	**99.06**	**76.95**	**120.27**	**173.89**	**110.78**	**140.84**	**120.30**	**182.74**	**216.74**	**199.74**
SI-CD ↓	Dynamic Gaussian [24]	45.63	1.66	0.66	1.06	6.31	2.74	9.68	20.27	53.99	37.13
	PAPR in Motion (Ours)	**17.98**	**1.32**	**0.25**	**0.27**	**1.99**	**1.82**	**3.94**	**8.91**	**9.19**	**9.05**
SI-EMD ↓	Dynamic Gaussian [24]	56.74	17.33	22.29	18.04	55.62	18.95	31.49	70.30	120.32	95.31
	PAPR in Motion (Ours)	**35.85**	**9.45**	**2.67**	**5.06**	**8.78**	**13.89**	**12.62**	**44.52**	**61.52**	**53.02**

Table 1. Comparison of scene interpolation appearance rendering quality, measured by Scene Interpoation FID (SI-FID), and geometry quality, measured by Scene Interpolation Chamfer distance (SI-CD) and Scene Interpolation Earth Mover's distance (SI-EMD). Lower scores for all three metrics indicate better quality. The SI-CD and SI-EMD results are reported on a scale of 10^{-3}. Our method consistently outperforms the baseline on both synthetic and real-world scenes.

5.3. Ablation Study

Figure 4 presents the results of our ablation study, where we incrementally remove the local displacement averaging step (LDAS) and the local distance preserving loss $\mathcal{L}_{rigid}$. The study reveals that upon the removal of the local averaging step, the points, while maintaining cohesiveness, still exhibit undesirable deformations in the part geometry, such as in the neck and rear leg of the giraffe. Further, the elimination of the loss $\mathcal{L}_{rigid}$ leads to excessive drift of the points, resulting in a failure to preserve the structural integrity of the object's shape. These results underscore the vital role of our regularization techniques in maintaining scene geometry and demonstrate their necessity for facilitating smooth and plausible scene interpolations.

6. Discussion and Conclusion

Limitations While our method showcases promising results, it does have certain limitations, which open avenues for future research. Specifically, since our model is pretrained on the start state, it might have challenges rendering areas not visible in the initial state but are present in the end state, which can happen when new objects enter the scene. Additionally, since we do not assume any prior knowledge on the scene object, such as its semantics, the generated motion may not precisely replicate real-world motion. These challenges, however, present exciting opportunities for further exploration and expansion of our method.

Conclusion In this paper, we introduce the novel task of point-level 3D scene interpolation between two distinct scene states. Addressing this novel and challenging task, we leverage the capabilities of the Proximity Attention Point Rendering (PAPR) technique to develop a method that can create plausible and smooth interpolations. Notably, our approach proves to be effective even in cases involving substantial non-rigid scene changes. Our work establishes a baseline on this challenging task and paves the way for further exploration and advancement.
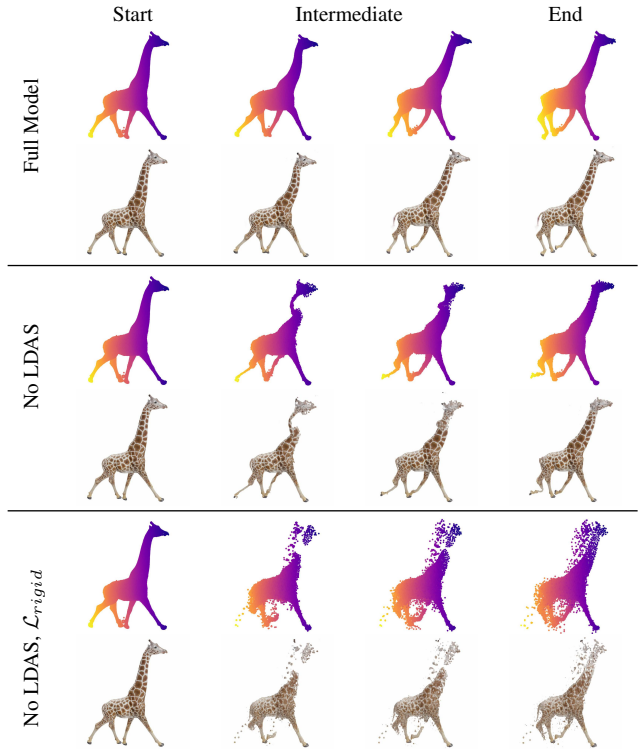

Figure 4. Ablation study on the impact of proposed regularization terms, where we incrementally remove the local displacement averaging step (LDAS) and the local distance preserving loss ($\mathcal{L}_{rigid}$). The results show that removing LDAS leads to unwanted deformations in the part geometry. When $\mathcal{L}_{rigid}$ is also removed, there is a further detrimental effect, with points drifting away from the object surface. These findings validates the critical role of both regularization techniques in maintaining the integrity and the quality of our model's interpolations.

References

[1] Ben Abbatematteo, Stefanie Tellex, and George Dimitri Konidaris. Learning to generalize kinematic models to novel objects. In *Conference on Robot Learning*, 2019. 3

[2] Jad Abou-Chakra, Feras Dayoub, and Niko Sunderhauf. Particlenerf: A particle-based encoding for online neural radiance fields in dynamic scenes. *ArXiv*, abs/2211.04041, 2022. 3

[3] Aayush Bansal and Michael Zollhoefer. Neural pixel composition for 3d-4d view synthesis from multi-views. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 290–299, 2023. 2

[4] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 130–141, 2023. 2

[5] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13142–13153, 2022. 6

[6] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B. Tenenbaum, and Jiajun Wu. Neural radiance flow for 4d view synthesis and video processing. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14304–14314, 2020. 2

[7] Hehe Fan, Yi Yang, and Mohan S. Kankanhalli. Point 4d transformer networks for spatio-temporal modeling in point cloud videos. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14199–14208, 2021. 3

[8] Sara Fridovich-Keil, Giacomo Meanti, Frederik Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12479–12488, 2023. 2

[9] Samir Yitzhak Gadre, Kiana Ehsani, and Shuran Song. Act the part: Learning interaction strategies for articulated object part discovery. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15732–15741, 2021. 3

[10] Sanjay Haresh, Xiaohao Sun, Hanxiao Jiang, Angel X. Chang, and Manolis Savva. Articulated 3d human-object interactions from rgb videos: An empirical analysis of approaches and challenges. *2022 International Conference on 3D Vision (3DV)*, pages 312–321, 2022. 3

[11] Nick Heppert, Muhammad Zubair Irshad, Sergey Zakharov, Katherine Liu, Rares Ambrus, Jeannette Bohg, Abhinav Valada, and Thomas Kollar. Carto: Category and joint agnostic reconstruction of articulated objects. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21201–21210, 2023. 3

[12] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Neural Information Processing Systems*, 2017. 7

[13] Mustafa Işık, Martin Rünz, Markos Georgopoulos, Taras Khakhulin, Jonathan Starck, Lourdes de Agapito, and Matthias Nießner. Humanrf: High-fidelity neural radiance fields for humans in motion. *ACM Transactions on Graphics (TOG)*, 42:1 – 12, 2023. 3

[14] Ajinkya Jain, Rudolf Lioutikov, and Scott Niekum. Screwnet: Category-independent articulation model estimation from depth images using screw theory. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13670–13677, 2020. 3

[15] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 1, 3, 7

[16] Christoph Lassner and Michael Zollhöfer. Pulsar: Efficient sphere-based neural rendering. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1440–1449, 2021. 1, 3

[17] Ruilong Li, Julian Tanke, Minh Vo, Michael Zollhofer, Jurgen Gall, Angjoo Kanazawa, and Christoph Lassner. Tava: Template-free animatable volumetric actors. *ArXiv*, abs/2206.08929, 2022. 3

[18] Xiaolong Li, He Wang, L. Yi, Leonidas J. Guibas, Amos Lynn Abbott, and Shuran Song. Category-level articulated object pose estimation. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3703–3712, 2019. 3

[19] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6494–6504, 2020. 2

[20] Jiayi Liu, Ali Mahdavi-Amiri, and Manolis Savva. Paris: Part-level reconstruction and motion analysis for articulated objects. *ArXiv*, abs/2308.07391, 2023. 3

[21] Xingyu Liu, C. Qi, and Leonidas J. Guibas. Flownet3d: Learning scene flow in 3d point clouds. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 529–537, 2018. 3

[22] Xingyu Liu, Mengyuan Yan, and Jeannette Bohg. Meteornet: Deep learning on dynamic 3d point cloud sequences. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9245–9254, 2019. 3

[23] Y. Liu, Chen Gao, Andreas Meuleman, Hung-Yu Tseng, Ayush Saraf, Changil Kim, Yung-Yu Chuang, Johannes Kopf, and Jia-Bin Huang. Robust dynamic radiance fields. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13–23, 2023. 1, 2, 3

[24] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. *ArXiv*, abs/2308.09713, 2023. 2, 3, 5, 6, 7, 8

[25] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *ArXiv*, abs/2003.08934, 2020. 1, 2, 3, 6

[26] Kaichun Mo, Leonidas J. Guibas, Mustafa Mukadam, Abhinav Kumar Gupta, and Shubham Tulsiani. Where2act: From

pixels to actions for articulated 3d objects. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6793–6803, 2021. 3

[27] Jiteng Mu, Weichao Qiu, Adam Kortylewski, Alan Loddon Yuille, Nuno Vasconcelos, and Xiaolong Wang. A-sdf: Learning disentangled signed distance functions for articulated shape representation. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12981–12991, 2021. 3

[28] Keunhong Park, U. Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B. Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5845–5854, 2020. 1, 2, 3

[29] Keunhong Park, U. Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B. Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. Hypernerf. *ACM Transactions on Graphics (TOG)*, 40:1 – 12, 2021.

[30] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10313–10322, 2020. 1, 2, 3

[31] Darius Rückert, Linus Franke, and Marc Stamminger. Adop: Approximate differentiable one-pixel point rendering. *ACM Trans. Graph.*, 41:99:1–99:14, 2021. 3

[32] Chong Chul. Song, Gengshan Yang, Kangle Deng, Junchen Zhu, and Deva Ramanan. Total-recon: Deformable scene reconstruction for embodied view synthesis. *ArXiv*, abs/2304.12317, 2023. 2

[33] Wei-Cheng Tseng, Hung-Ju Liao, Yen-Chen Lin, and Min Sun. Cla-nerf: Category-level articulated neural radiance field. *2022 International Conference on Robotics and Automation (ICRA)*, pages 8454–8460, 2022. 3

[34] Haithem Turki, Jason Y. Zhang, Francesco Ferroni, and Deva Ramanan. Suds: Scalable urban dynamic scenes. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12375–12385, 2023. 2

[35] Qianqian Wang, Yen-Yu Chang, Ruojin Cai, Zhengqi Li, Bharath Hariharan, Aleksander Holynski, and Noah Snavely. Tracking everything everywhere all at once. *ArXiv*, abs/2306.05422, 2023. 2

[36] Xiaogang Wang, Bin Zhou, Yahao Shi, Xiaowu Chen, Qinping Zhao, and Kai Xu. Shape2motion: Joint analysis of motion parts and attributes from 3d shapes. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8868–8876, 2019. 3

[37] Fangyin Wei, Rohan Chabra, Lingni Ma, Christoph Lassner, Michael Zollhofer, Szymon Rusinkiewicz, Chris Sweeney, Richard A. Newcombe, and Miroslava Slavcheva. Self-supervised neural articulated shape and appearance models. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15795–15805, 2022. 3

[38] Chung-Yi Weng, Brian Curless, Pratul P. Srinivasan, Jonathan T. Barron, and Ira Kemelmacher-Shlizerman. Humannerf: Free-viewpoint rendering of moving people from monocular video. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16189–16199, 2022. 3

[39] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7465–7475, 2019. 3

[40] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9416–9426, 2020. 2

[41] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5428–5438, 2022. 1, 3

[42] Zihao Yan, Ruizhen Hu, Xingguang Yan, Luanmin Chen, Oliver Matias van Kaick, Hao Zhang, and Hui Huang. Rpm-net: Recurrent prediction of motion and parts from point cloud. *ArXiv*, abs/2006.14865, 2020. 3

[43] Gengshan Yang, Minh Vo, Natalia Neverova, Deva Ramanan, Andrea Vedaldi, and Hanbyul Joo. Banmo: Building animatable 3d neural models from many casual videos. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2853–2863, 2021. 2

[44] Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5491–5500, 2021. 1, 3

[45] Y. Zeng et al. Idea-net: Dynamic 3d point cloud interpolation via deep embedding alignment. *CVPR*, 2022. 3

[46] Qian Zhang, Seung-Hwan Baek, Szymon Rusinkiewicz, and Felix Heide. Differentiable point-based radiance fields for efficient view synthesis. *SIGGRAPH Asia 2022 Conference Papers*, 2022. 3

[47] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018. 4

[48] Yanshu Zhang*, Shichong Peng*, Alireza Moazeni, and Ke Li. Papr: Proximity attention point rendering. *Advances in Neural Information Processing Systems*, 2023. 2, 3, 4, 7

[49] Yiming Zuo and Jia Deng. View synthesis with sculpted neural points. *ArXiv*, abs/2205.05869, 2022. 1, 3