ParticleNeRF: A Particle-Based Encoding for Online Neural Radiance Fields

Jad Abou-Chakra[1] Feras Dayoub[2] Niko Sünderhauf[1]
[1]Queensland University of Technology [2]University of Adelaide

Abstract

*While existing Neural Radiance Fields (NeRFs) for dynamic scenes are offline methods with an emphasis on visual fidelity, our paper addresses the **online** use case that prioritises real-time adaptability. We present ParticleNeRF, a new approach that dynamically adapts to changes in the scene geometry by learning an up-to-date representation online, every 200 ms. ParticleNeRF achieves this using a novel particle-based parametric encoding. We couple features to particles in space and backpropagate the photometric reconstruction loss into the particles' position gradients, which are then interpreted as velocity vectors. Governed by a lightweight physics system to handle collisions, this lets the features move freely with the changing scene geometry. We demonstrate ParticleNeRF on various dynamic scenes containing translating, rotating, articulated, and deformable objects. ParticleNeRF is the first online dynamic NeRF and achieves fast adaptability with better visual fidelity than brute-force online InstantNGP and other baseline approaches on dynamic scenes with online constraints. Videos of our system can be found at the anonymous project website https://sites.google.com/view/particlenerf.*

1. Introduction

Neural Radiance Fields [19] are 3D scene representations trained from images using a differentiable rendering process that allows them to render scenes from novel viewpoints with high visual fidelity.

Dynamic NeRFs [6, 8, 23–26] learn to represent *dynamic* scenes. However, these methods are currently *offline*, which means they require access to the *entire* image sequence during training and can take several minutes or even hours to train for sequences that last only seconds in real-time.

Our paper addresses the challenge of ***online*** dynamic Neural Radiance Fields that can continuously learn an up-to-date implicit 3D representation of the dynamic scene in real-time, without the need for access to future frames during training.

We introduce ParticleNeRF, a novel dynamic NeRF approach that uses a particle-based encoding to represent the dynamic scene. Our key insight and novelty – illustrated in Fig. 1 – is that backpropagating the reconstruction loss into both the positions and the features of the particles enables the embedding features to move in space as the scene

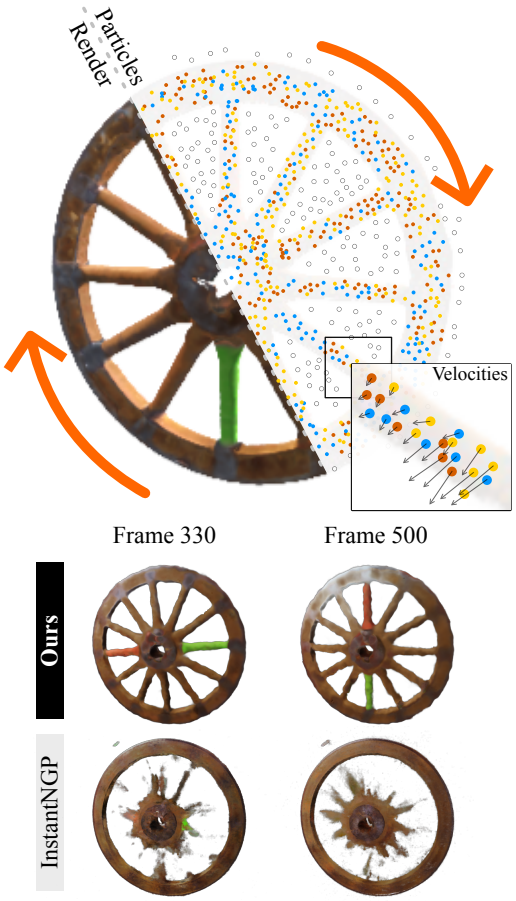

Figure 1. Our particle encoding significantly improves the ability to adapt to changing scenes when compared to online InstantNGP [20], the top-performing parametric encoding. The particle encoding fills the space with particles at random locations, associating each particle with a feature. Both the positions of the particles and their features are optimized during training. As the wheel spins, the particles move to maintain a low reconstruction error.

changes. As we illustrate in Fig. 2, this lets ParticleNeRF elegantly adapt to moving, articulated or deforming objects by adjusting the position of the feature-carrying particles.

ParticleNeRF uses a simple but effective physics system to govern the particles' positions and velocities, interpreting the individual position gradients as velocity vectors. The physics system implements collision constraints that prevent the particles from accumulating too close to each other. This creates a soft upper limit on the number of nearest neighbours to be considered for interpolation when calculating the feature for a query point, thereby supporting ParticleNeRF's efficient training process.

In contrast to the time-conditioned deformation networks used in [6, 8, 23–26], which are designed for *offline* training, ParticleNeRF incrementally adapts to changes in the environment every 200 ms – making it suitable for *online* usage.

ParticleNeRF is the first method to address the challenge of *online* learning of dynamic scenes (including rigid body transforms, articulated objects, and deforming objects). Our experiments show that ParticleNeRF can represent dynamic scenes with higher fidelity than (i) brute-force online Instant-NGP that simply continuously trains the network as new frames become available, (ii) and other baselines (Sec. 5). Our ablation in Sec. 5.2 shows that this superior performance on dynamic scenes is indeed due to the ability to backpropagate into the positions and move the feature-carrying particles during the training process. We furthermore show that ParticleNeRF's performance degrades gracefully with a decrease in the number of particles and the nearest neighbour search radius.

In summary, our contributions are:

1. A novel online NeRF formulation that can adapt to the changes in dynamic scenes every 200 ms without using time-conditioned deformation networks;
2. A new particle-based encoding that associates features with moving particles and allows the gradients from the photometric loss to propagate to and change the positions of the particles;
3. The incorporation of a physics system into the NeRF formulation to update the motion of the particles while preventing collisions.

Our implementation is open-sourced. Videos of ParticleNeRF rendering dynamic scenes can be found in the attached supplementary.

2. Related Work

We review NeRFs in the context of (i) how they have been used to tackle dynamic environments, (ii) how certain encodings allow close to realtime training and rendering, and lastly (iii) how our particle encoding is different to three other point-based encodings suggested to date.

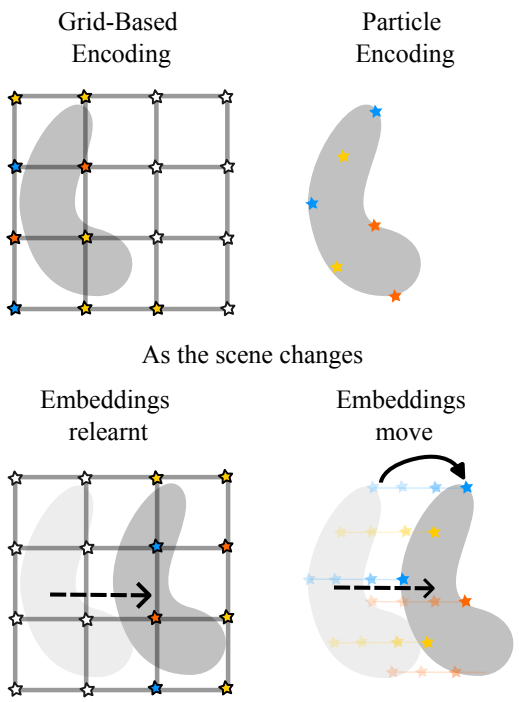

Figure 2. In a rigid grid-based encoding (e.g. InstantNGP [20]), features are associated with fixed positions in space and cannot move. If the scene geometry changes, features have to be unlearned and relearned in different positions. In contrast, our particle encoding associates features with particles that can move freely in 3D space. As the scene geometry changes, the particles can move to accommodate the scene change, resulting in faster adaptation and higher quality representation of dynamic scenes.

Dynamic NeRFs Several papers extended the NeRF model to encompass dynamic scenes [6, 8, 9, 15, 23–26, 32]. The distinguishing factor between the static and dynamic NeRF formulations is that the first represents a single static scene while the latter encodes the evolution of a 3D scene over time. During training, these dynamic NeRFs can access images taken at any time step. They are therefore designed to be trained **after** a sequence of images has already been taken. To date, the majority of these works are not close to being realtime capable taking minutes and even hours to train. Since these dynamic NeRF approaches use the complete set of images during training, we refer to them as "offline" methods. Alternatively, we seek to create a NeRF at every timestep and only capture the latest state of the scene. In this context, our method is "online". For many applications, representing a time-varying scene within the NeRF itself is unnecessary. A robot, for example, is usually only interested in the latest state of the world.

Learning an online NeRF is therefore the challenge of learning a NeRF at a timestep t given a NeRF at a timestep $t-1$ and a set of images taken at time t. Our particle

encoding is formulated to tackle this problem.

Embeddings In general, a NeRF is composed of two main parts: the encoding, and a multi-layered perceptron (MLP). An encoding transforms the spatial input of the NeRF to an alternative feature space where learning can be made easier [29]. The MLP transforms the resultant feature into values that represent color and geometry. Mildenhall et al. [19] introduced the first version of NeRFs with a Fourier encoding which is a simple trigonometric map. The Fourier encoding does not have any parameters that are learnt as part of the training and is not associated with a discrete datastructure. It is, therefore, referred to as non-parametric, or functional. This version of NeRF takes hours to train because it requires a large MLP to represent the scene. Later works [4, 20, 27] show that by storing a large set of features in a discrete datastructure, the spatial input could be used to index into and interpolate between features in that set. The computed features become the embeddings that are consumed by the NeRF's neural network. Since the features in the set are stored as parameters and are updated during training, these are referred to as "parametric" encodings by [20]. By storing features in memory, the computational burden on the MLP is reduced and training speed is significantly increased. The strategy used to index and interpolate into the feature set distinguishes the various types of parametric encodings. Almost all the parametric encodings conceptually rely on a fixed grid datastructure [4, 20, 27] with the exception of [33] which uses a point-based encoding. The grid encompasses the workspace of the NeRF and each node within is associated with a feature. Therefore, while the features associated with the nodes can be learnt, they cannot be moved.

In this paper, we show that if the features are allowed to be both learnt and moved, then NeRFs can be made to incrementally adapt to dynamic scenes. We introduce a particle encoding, where features are associated with moving points in space. We show that in a changing scene, the feature-holding particles move in accordance with the scene – Fig. 2.

Point-based NeRFs There are three works that use points as part of their formulation. Point-NeRF and SPIDR [16, 33] utilize points as a data structure to store features, but their embedding differs significantly from ours. Notably, they do not address the dynamic scene use case and they do not allow their points to move during optimization. In addition, they require depth data, their training takes in the order of minutes to hours, and their encoding is not invariant to $\mathrm{SE}(3)$ transformations. NeuroFluid [11] uses particles and NeRFs to specifically model the behaviour of fluids. Their encoding is non-parametric and their points represent particles of a physical medium rather than descriptors of an embedding space.

3. NeRF Preliminaries

A NeRF is a continuous representation of a 3D scene that maps a point $\boldsymbol{x} \in \mathbb{R}^3$ and a unit-norm view direction $\boldsymbol{d} \in \mathbb{R}^3$ to a color $\boldsymbol{c} \in \mathbb{R}^3$ and a density value $\sigma \in \mathbb{R}$. During training, rays are cast from camera centers through their image plane. Each ray $\boldsymbol{r}$ is associated with a direction vector $\boldsymbol{d}$ and a set of points $\boldsymbol{x}_i$ progressively sampled from the ray's length [19]. The points $\boldsymbol{x}_i$ and the direction vector $\boldsymbol{d}$ are mapped by the NeRF to $\boldsymbol{c}_i$ and σ_i. Defining $\delta_i = \|\boldsymbol{x}_{i+1} - \boldsymbol{x}_i\|$, the expected color of the ray $\hat{C}(\boldsymbol{r})$ is given by:

$$\hat{C}(\boldsymbol{r}) = \sum_{i=1}^{N} w_i \boldsymbol{c}_i \tag{1}$$

$$w_i = T_i(1 - \exp(-\sigma_i \delta_i)) \quad \text{and} \quad T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right)$$

A photometric loss is defined as

$$L_{\text{rgb}} = \sum_{\boldsymbol{r} \in \mathcal{R}} \|\hat{C}(\boldsymbol{r}) - \boldsymbol{c}_{\text{gt}}(\boldsymbol{r})\|_2 \tag{2}$$

The ground-truth color $\boldsymbol{c}_{gt}(\boldsymbol{r})$ is the color of the pixel in the training image that the ray $\boldsymbol{r}$ intersects. $\mathcal{R}$ is the set of all rays that pass through the training images. Refer to [18, 28] for an in-depth derivation.

Encodings Given a point $\boldsymbol{x}_i$ and a direction vector $\boldsymbol{d}_i$, a NeRF first maps each to an embedding space $E_x(\boldsymbol{x}_i)$ and $E_d(\boldsymbol{d}_i)$. The embeddings are subsequently consumed by a multilayered perceptron to output color and density. Not including an embedding layer critically deteriorates ability of the NeRF to reconstruct scenes [29]. The choice of the embedding layer can significantly reduce the size of the MLP required [4, 20, 33] and in some cases removes the need for one entirely [7]. Close to realtime performance is possible due to a particular class of encodings that the authors of [20] refer to as "parametric". These encodings are characterized by a discrete datastructure that can be queried by the incoming coordinate $\boldsymbol{x}_i$ to produce a set of features $\mathbf{f}_j$. The features $\mathbf{f}_j$ are then combined to produce an embedding $\mathbf{f}_i$. By storing the features in memory, a large MLP is no longer required and significant savings are made in both rendering and training times when compared to other formulations that use functional encodings [2, 3, 5, 19, 31].

4. ParticleNeRF

ParticleNeRF introduces a new particle-based encoding that associates features with moving particles and allows the gradients from the photometric loss to propagate through and change the positions of the particles. This is a novel method of dealing with dynamic scenes in an online manner without using time-conditioned deformation networks.

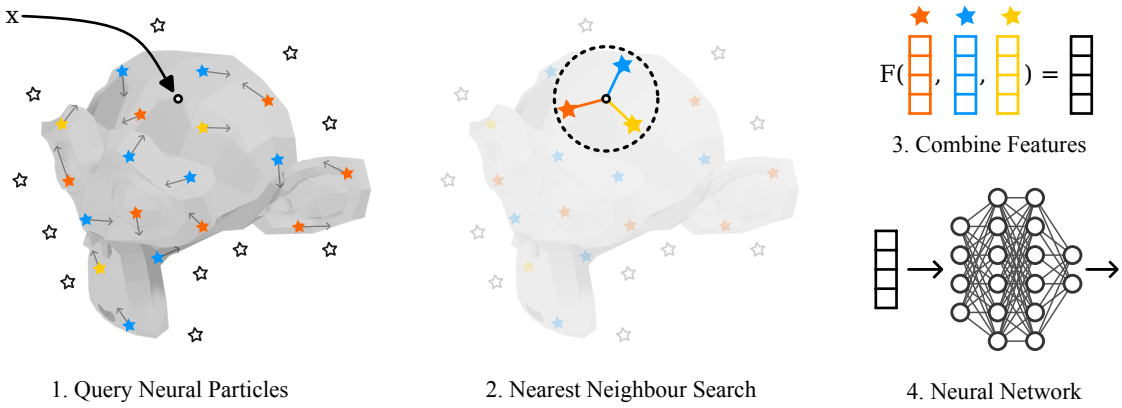

Figure 3. Illustration of our particle encoding. **(1)** A query point $\boldsymbol{x}$ is sampled in space. **(2)** The features and positions of the particles within a search radius are retrieved. **(3)** The features and distances from the query point are used to interpolate the feature at the query point $\boldsymbol{x}$. **(4)** The resulting feature is evaluated by the neural network to give color and density. To train the encoding, the loss gradients are backpropagated through the network **(4)**, the query feature **(3)**, and finally into the positions and features of the particles in **(2)**.

Our particle encoding is formulated as a neural pointcloud $\mathcal{P} = \{(\boldsymbol{x}_i, \boldsymbol{v}_i, \mathbf{f}_i) : i = 1, 2, ..., M\}$, where $\boldsymbol{x}_i \in \mathbb{R}^3$ is the position of a latent particle in the system, $\boldsymbol{v}_i \in \mathbb{R}^3$ is the velocity of the particle, $\mathbf{f}_i \in \mathbb{R}^m$ is its associated feature, and M is the total number of particles. The pointcloud $\mathcal{P}$ is an irregular grid of latent features that defines the map F from a coordinate $\boldsymbol{x}_j \in \mathbb{R}^3$ to $\mathbf{f}_j$.

$$\mathbf{f}_j = F(\boldsymbol{x}_j, \mathcal{P}) = \sum_{(\boldsymbol{x}_i, \mathbf{f}_i) \in \mathcal{P}} w(\|\boldsymbol{x}_j - \boldsymbol{x}_i\|_2)\mathbf{f}_i \tag{3}$$

where w is the bump function – a compactly supported radial basis function (RBF) – given by:

$$w(r) = \begin{cases} \exp\left(-\frac{s^2}{s^2 - r^2}\right), & \text{for } r \in (-s^2, s^2) \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

The search radius s controls the influence of a particle on a region of space. The compact RBF decreases the influence of a given particle on the interpolated feature to 0 as its distance approaches the search radius. This allows for the efficient calculation of $\mathbf{f}_j$ with a fixed-radius nearest neighbour search algorithm [30] where the radius is set to s. We find that using unnormalized features encourages the particles to spread across the geometry.

ParticleNeRF uses the same architecture as InstantNGP (a 3 layer MLP with 64 neurons each), but replaces its hash encoding with our particle encoding. Note that if a query point finds no neighbouring particles within the search radius s, the feature is set to $\mathbf{0}$. Fig. 3 illustrates our method.

We optimize the MLP parameters Φ, the particle features $\mathbf{f}_i$ and the particle positions $\boldsymbol{x}_i$ relative to the NeRF loss L_{rgb} defined in Eq. (2):

$$\Phi^*, \{\mathbf{f}_i\}^*, \{\boldsymbol{x}_i\}^* = \underset{\Phi, \{\mathbf{f}_i\}, \{\boldsymbol{x}_i\}}{\arg\min} L_{\text{rgb}} \tag{5}$$

A core novelty of our approach is that the gradients of the NeRF loss are used to update *both* the position and the associated features of the particles. This allows the particles to move with the geometry in the scene as it changes, leading to a higher fidelity representation of dynamic scenes. As our experiments in Sec. 5 will show, ParticleNeRF can adapt to changes in the scene *online*, i.e. every 200 ms, which is orders of magnitude faster than previous NeRFs in dynamic scenes [9, 15, 23–25, 32], and enables a much higher reconstruction quality compared to online InstantNGP [20] and other baselines.

4.1. Position Based Dynamics

We observe that propagating the NeRF loss into the particle positions can sometimes lead to multiple particles accumulating in a small region of space. Although this does not adversely affect the reconstruction quality, it is not desirable because it needlessly increases the number of neighbours that each query point has to process during training and evaluation. To address this problem in a structured way, we build a position-based physics system [17, 21] into InstantNGP that resolves the dynamics of our particles.

Position-based dynamics (PBD) is a simple, robust, and fast physics system that can evolve the motion of our particles while resolving the constraints we put on them. We interpret the gradients $\frac{dL_{\text{rgb}}}{d\boldsymbol{x}_i}$ as scaled velocity vectors that are added to the particles' current velocity – Algorithm 1. Using PBD not only provides a means of limiting the number of neighbours a particle has, but it also creates future opportunities to add other constraints into the system. For example, if an object is known to be a rigid object, or part of

Algorithm 1 PBD Physics Step

for all particles i **do**
  $\boldsymbol{v}_i \leftarrow \gamma \boldsymbol{v}_i - \alpha \frac{dL_{\text{rgb}}}{d\boldsymbol{x}_i}$ ▷ Update velocities
  $\mathbf{p}_i \leftarrow \boldsymbol{x}_i$ ▷ Store previous positions
  $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + \Delta t \boldsymbol{v}_i$ ▷ Integrate positions
for all collision pairs i, j **do**
  $l \leftarrow ||\boldsymbol{x}_j - \boldsymbol{x}_i||$
  $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + 0.5(l - \delta)\frac{\boldsymbol{x}_j - \boldsymbol{x}_i}{l}$ ▷ Resolve collisions
for all particles i **do**
  $\boldsymbol{v}_i \leftarrow (\boldsymbol{x}_i - \mathbf{p}_i)/\Delta t$ ▷ Update velocities

an articulated body, the particles can be suitably constrained.

In all our experiments, we choose a damping factor $\gamma = 0.96$, a timestep $\Delta t = 0.01$, a minimum distance of $\delta = 0.01$. The base InstantNGP implementation transforms a given scene so that it fits within a unit cube. Our minimum distance δ in the physics system and the search radius s that is used to query neighbouring particles are given in the units of the unit cube.

4.2. Implementation

We build our particle encoding as an extension of the InstantNGP implementation released by Müller *et al*. [20]. We clip the particle position gradients so that their norms do not exceed the search radius. We implement the sorting-based fixed-radius nearest search algorithm described by [10] and the position-based dynamics physics system described by [21] as additional CUDA kernels [22] within InstantNGP. Two different Adam optimizers [14] are used. The first is for the parameters of the MLP Φ and the second is for the particle features $\mathbf{f}_i$. Both are parameterized by $\beta_1 = 0.9$, $\beta_2 = 0.99$, $\epsilon = 10^{-10}$ with the learning rate set at 0.01. The particle positions are updated through the PBD system where collision constraints can be resolved.

5. Experiments

We experimentally evaluate ParticleNeRF and show its abilities to represent a dynamic environment online.

Datasets: We construct a dynamic dataset consisting of 6 scenes, inspired by D-NeRF [25], comprising of scenes with deformable and articulated moving objects that loop. In addition, we created an animated version of the original NeRF Blender dataset [19], by applying translations and rotations to the objects.

We note that datasets such as [23, 24] that are commonly used to evaluate offline dynamic NeRFs are not compatible with ParticleNeRF's (and InstantNGP's) requirement for multiple cameras per frame. We discuss this limitation further in Sec. 6.

Baseline: We compare ParticleNeRF against online InstantNGP [20] and online TiNeuVox [6].

InstantNGP is the fastest NeRF implementation currently available, and it can be utilized for online applications by continuously learning on new frames without resetting the weights. We are not the first to propose this, as EvoNeRF [12] uses a continuous data stream to train InstantNGP for a robotic grasping task. InstantNGP's fast training speed allows us to evaluate whether a brute force approach of continuous learning is a feasible strategy for online dynamic scenes.

As most of the dynamic NeRF approaches [9, 15, 23–25, 32] published to date are offline methods that are not trained at the required speeds for online applications, we do not compare our approach against them. However, a recent offline method called TiNeuVox [6] has been released, which trains within minutes by utilizing a time-conditioned deformation network and voxel-based parametric encodings. We compare our approach against the two configurations of TiNeuVox released by the authors: TiNeuVox and TiNeuVox-S. We convert them to an online use-case where only images from the current timestep can be sampled.

Hyperparameters: For all the experiments involving the particle encoding, we initialize the unit cube containing the scene with a grid of uniformly spaced particles (We obtain comparable outcomes when initializing the particles randomly). Each particle is associated with a 4 dimensional feature and is initialized randomly between $\pm 10^{-2}$. Increasing the dimension of the feature has minimal impact.

5.1. Dynamic Dataset

We evaluate our method using our dynamic dataset (Fig. 4). Each scene in the dataset has one or multiple objects being looked upon by 20 cameras distributed on the top half of the surface of a sphere. The objects are either articulated (robot), deformable (spring, cloth), or rigid (wheel). The remaining scenes are comprised of multiple moving objects (pendulums, robot system).

During each experiment, the scene remains fixed for 500 training steps, after which the animation starts. ParticleNeRF, online InstantNGP, and online TiNeuVox have only **five** training iterations before loading the next animation step.

At the end of every frame, we evaluate the photometric reconstruction quality (SSIM, PSNR and LPIPS) from 10 unseen views. The SSIM values are plotted against each frame in Fig. 4. The average PSNR, SSIM, and LPIPS values of all the scenes for each method are shown in Tab. 1. Qualitative visualizations are shown in Fig. 5 and the full videos can be found in the attached supplementary.

Our findings reveal that ParticleNeRF effectively adapts to motion within a small number of training steps, while the performance of online InstantNGP noticeably declines under motion. InstantNGP is able to recover when the speed

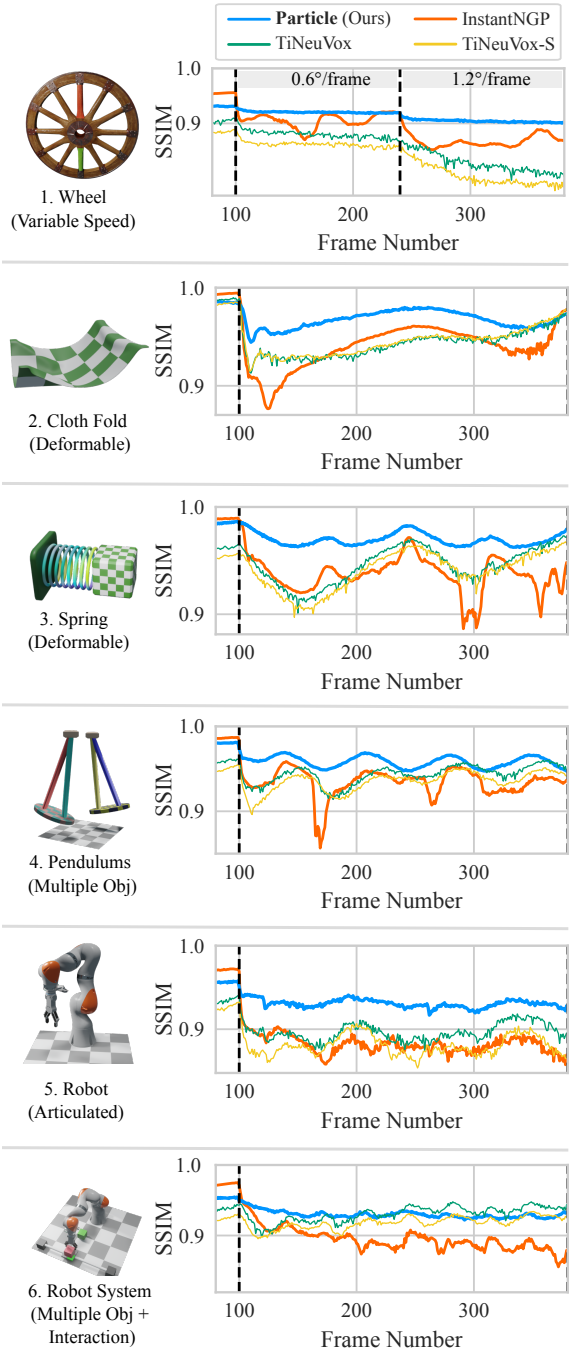

Figure 4. ParticleNeRF evaluated on 6 scenes with deformable, articulated, and multiple object movements. Scenes are held static for 100 frames before the animation begins. ParticleNeRF provides a stable representation during movements. InstantNGP can recover when motions slow down. Online TiNeuVox fails when movements get faster.

of movements slow down. Nonetheless, ParticleNeRF exhibits reduced performance when objects in the scene have

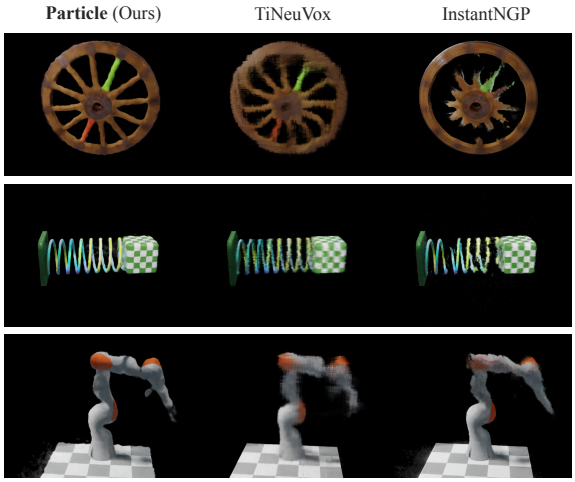

Figure 5. Image captures from a novel viewpoint at the 300th frame for ParticleNeRF, TiNeuVox, and InstantNGP.

Table 1. Average quantitative performance of methods on all dynamic scenes

Method	PSNR↑	SSIM↑	LPIPS↓
Particle	**27.47**	**0.94**	**0.08**
InstantNGP	24.69	0.91	0.12
TiNeuVox	27.28	0.91	0.13
TiNeuVox-S	26.64	0.92	0.14

varying sizes, which is due to its reliance on a fixed search radius, as shown by the decreased performance in the robot system scene. Additionally, online TiNeuVox exhibits artifacts, particularly visible in the wheel scene, caused by the deformation network's inability to adapt within 5 training iterations.

5.2. Animated Blender Dataset

We further evaluate our method using a novel animated version of the standard "Blender" dataset [19]. Each of the 8 scenes is made to rotate or translate at a certain speed for 100 frames to test ParticleNeRF's ability to deal with elementary motions. At the end of every frame, we evaluate the photometric reconstruction quality (PSNR) from 7 unseen views. In total, 56 animations are tested and their results are summarized in Tab. 2. Videos and a more detailed view of the results can also be found in the attached supplementary. The results reinforce our findings that allowing particles to move provides a method of adapting to moving geometries.

5.3. Ablation

To thoroughly ablate and understand the dynamic representational ability of ParticleNeRF, we use the wheel scene from the dynamic dataset. To investigate ParticleNeRF's

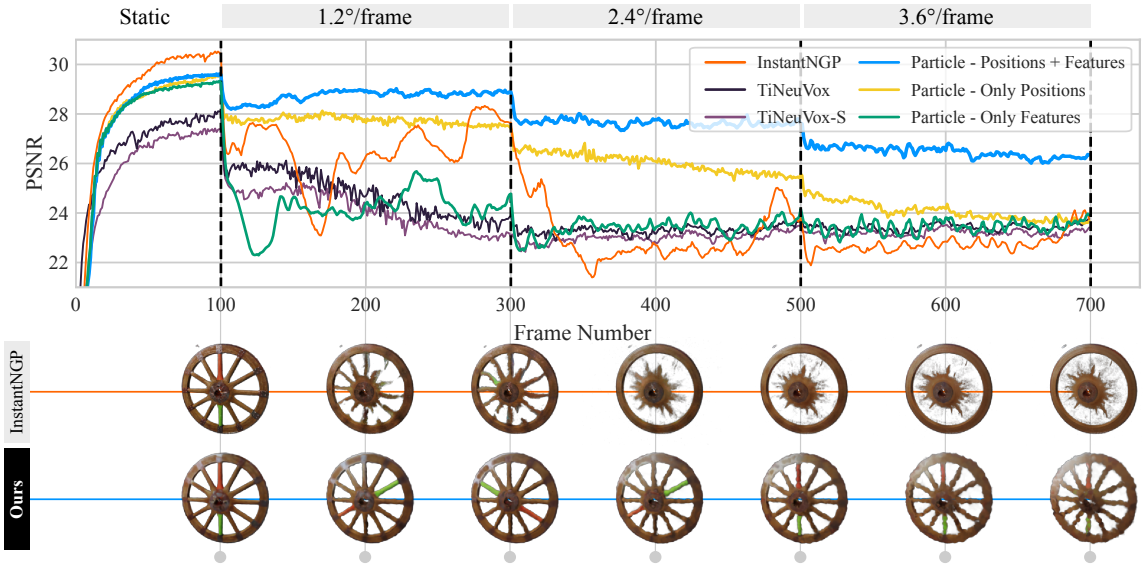

Figure 6. On the Wheel dataset, ParticleNeRF maintains the structure of the wheel as it rotates. We show how our particle encoding performs in three settings: **(i)** When both the features and the positions of the particles are optimized (blue). **(ii)** When only the positions of the particles are optimized (yellow). **(iii)** When only the particle features are optimized (green). Lastly we show how InstantNGP [20] performs (orange). We illustrate the results of the reconstruction from an unseen viewpoint at the frames indicated for both InstantNGP and the complete ParticleNeRF, corresponding to setup **(i)**. Even at the lowest PSNR at frame 700, our method preserves the wheel's structure.

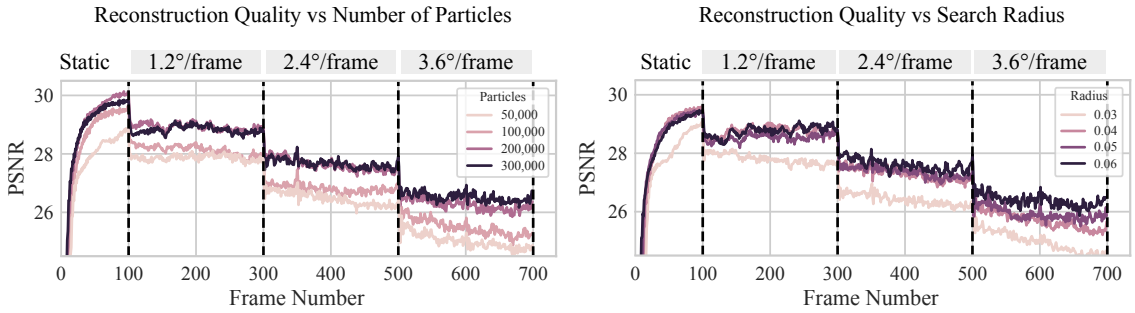

Figure 7. An ablation showing the effect of the number of particles (left) and the particle search radius (right) on the photometric reconstruction quality. When ablating the number of particles, search radius is set at 0.04. When ablating the search radius, number of particles is set at 100,000. In both cases, there is a diminishing effect at higher speeds.

and the baselines' performance in static scenes and dynamic scenes with varying degrees of motion, we change the rotation speed of the wheel from static to successively faster speeds. Most of the following results are illustrated in Fig. 6. Models are given 5 training steps per frame and are configured with the same parameters from Sec. 5.1.

Optimising Particle Positions We first investigate the contribution of ParticleNeRF's ability to backpropagate the rendering loss into the particle positions to the overall performance. In Fig. 6 we show PSNR performance when allowing the loss to propagate into only the features (green); only the particle positions, keeping the features at their random initialisations (yellow), and allowing to learn both the positions and the features (blue). We make the following observations: First, when allowing to learn only the features (green), the performance is identical to InstantNGP (orange), especially at the higher motion rates. This is not surprising, since in that setup ParticleNeRF operates like InstantNGP with randomly positioned feature embeddings, instead of a fixed grid structure. Second, keeping the features fixed, but allowing the particles to move, leads to increased PSNR performance that already outperforms the baselines in all but a few dynamic frames. Third, the combination of learning positions and features achieves best performance.

Table 2. Average PSNR on the animated blender scenes from [19].

Model	Enc.	Rot. (°/frm) 1	2	3	4	Trans. (cm/frm) 1	2	3
Chair	ngp	21	18	17	17	23	19	18
	ours	**24**	**23**	**22**	**21**	**24**	**23**	**22**
Drums	ngp	19	18	17	16	19	18	17
	ours	**20**	**19**	**19**	**18**	**20**	**19**	**19**
Ficus	ngp	22	22	21	21	22	21	**21**
	ours	**23**	**22**	**22**	**21**	**22**	**22**	21
Hotdog	ngp	26	24	23	23	25	23	21
	ours	**27**	**27**	**26**	**25**	**26**	**25**	**24**
Lego	ngp	21	19	18	18	21	20	19
	ours	**23**	**22**	**21**	**21**	**23**	**22**	**21**
Mic	ngp	26	24	23	23	27	24	23
	ours	**28**	**27**	**26**	**25**	**28**	**27**	**26**
Ship	ngp	23	22	22	21	22	21	21
	ours	**23**	**23**	**23**	**22**	**23**	**23**	**22**
Matrl.	ngp	21	19	17	17	23	21	19
	ours	**23**	**23**	**22**	**21**	**23**	**23**	**23**

(a) Rotation (b) Translation

Increasing Motion Magnitude From Fig. 6 we can see that brute-force InstantNGP achieves a higher PSNR on static scenes, but drops significantly in dynamic scenes as it is not capable of relearning features fast enough to adapt. While ParticleNeRF has a slightly worse performance on static scenes, it maintains its PSNR with only slight degradation as the scene becomes more dynamic. Comparing the renderings at the bottom of Fig. 6, we can see the significant motion artifacts exhibited by brute-force InstantNGP. We furthermore observe a ghosting effect where InstantNGP quickly recovers structure upon the wheel's return to its original configuration. This is manifested as oscillations in the PSNR metric in Fig. 6.

Influence of Search Radius The right side of Fig. 7 illustrates how increasing search radii have a diminishing effect on reconstruction quality. We further observe that at the smaller search radius of 0.03 and a higher speed of $3.6°$/frame, reconstruction quality begins to degrade over time. This affect is due to a divergence that occurs when the number of neighbours around a particle drops below the necessary level to accurately represent the underlying geometry. We currently do not have a growing strategy that would detect such an event and multiply the particles in that region. This edge case and future work for implementing a growing strategy is further discussed in the supplementary.

Influence of Number of Particles In Fig. 7 (left), we observed that increasing the number of particles used to initialize the scene asymptotically improves reconstruction quality, with $300,000$ particles producing the same quality as $200,000$. However, we also found that only approximately 7,000 particles contribute to regions with high density, with the remaining particles providing minimal contribution. The additional particles improve reconstruction quality in two ways: Firstly, they enable more particles to begin near the geometry, thereby reducing the initial convergence time. Secondly, as movement occurs, particles may be ejected from the geometry, and having free particles in empty regions enables the geometry to reabsorb these particles, thus maintaining reconstruction quality. These observations suggest the need for future work on developing a pruning and growing strategy for the particles. We will release an implementation of one such strategy with our codebase. An example of it operating can be found on our project page.

6. Conclusions, Limitations and Future Work

Limitations Our particle embedding and online formulation require at least 10 views per frame to constrain the training, while offline dynamic methods that use time-conditioned deformation networks can use frames from any point during the trajectory to meet this requirement. We can overcome this disadvantage by incorporating depth supervision [1] and additional regularizers into the NeRF loss [13].

ParticleNeRF currently has a cubic memory requirement – albeit with a low constant – which can be addressed by implementing a pruning strategy that removes particles with a density below a certain threshold.

ParticleNeRF has lower visual fidelity on static scenes when compared to other methods. While we prioritized creating an encoding that is invariant to rigid transformations and quick to adapt to motions, we sacrificed the visual fidelity that comes with either multilevel encodings or positional Fourier transforms. These limitations are topics for future research.

Conclusion We show that moving embeddings in space and directly controlling them is a valid strategy for online scenes. By leveraging the position of our particles, we have shown an alternative approach to using time-conditioned deformation networks for dynamic scenes. Our approach is quicker to adapt to movement in the scenes providing a consistent representation that is robust to movement. We hope to inspire future work to build on this paradigm and address robustness, efficiency, and applicability.

Acknowledgements: The authors acknowledge continued support from the Queensland University of Technology (QUT) through the Centre for Robotics. This work was partially supported by the Australian Government through the Australian Research Council's Discovery Projects funding scheme (Project DP220102398).

References

[1] Jad Abou-Chakra, Feras Dayoub, and Niko Sunderhauf. Implicit object mapping with noisy data. *ArXiv*, abs/2204.10516, 2022. 8

[2] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021. 3

[3] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022. 3

[4] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision (ECCV)*, 2022. 3

[5] Shin-Fang Chng, Sameera Ramasinghe, Jamie Sherrah, and Simon Lucey. Garf: Gaussian activated radiance fields for high fidelity reconstruction and pose estimation. *arXiv e-prints*, pages arXiv–2204, 2022. 3

[6] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. *arxiv:2205.15285*, 2022. 1, 2, 5

[7] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5501–5510, June 2022. 3

[8] Wanshui Gan, Hongbin Xu, Yi Huang, Shifeng Chen, and Naoto Yokoya. V4d: Voxel for 4d novel view synthesis. *arXiv preprint arXiv:2205.14332*, 2022. 1, 2

[9] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5712–5721, October 2021. 2, 4, 5

[10] Simon Green. Particle simulation using cuda. *NVIDIA whitepaper*, 2010. 5

[11] Shanyan Guan, Huayu Deng, Yunbo Wang, and Xiaokang Yang. Neurofluid: Fluid dynamics grounding with particle-driven neural radiance fields. In *ICML*, 2022. 3

[12] Justin Kerr, Letian Fu, Huang Huang, Jeffrey Ichnowski, Matthew Tancik, Yahav Avigal, Angjoo Kanazawa, and Ken Goldberg. Evo-neRF: Evolving neRF for sequential robot grasping. In *6th Annual Conference on Robot Learning*, 2022. 5

[13] Mijeong Kim, Seonguk Seo, and Bohyung Han. Infonerf: Ray entropy minimization for few-shot neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12912–12921, 2022. 8

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[15] Tianye Li, Mira Slavcheva, Michael Zollhöfer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, and Zhaoyang Lv. Neural 3d video synthesis from multi-view video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5521–5531, June 2022. 2, 4, 5

[16] Ruofan Liang, Jiahao Zhang, Haoda Li, Chen Yang, and Nandita Vijaykumar. Spidr: Sdf-based neural point fields for illumination and deformation. *arXiv preprint arXiv:2210.08398*, 2022. 3

[17] Miles Macklin, Matthias Müller, and Nuttapong Chentanez. Xpbd: Position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games*, MIG '16, page 49–54, New York, NY, USA, 2016. Association for Computing Machinery. 4

[18] N. Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995. 3

[19] 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. In *ECCV*, 2020. 1, 3, 5, 6, 8

[20] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. 1, 2, 3, 4, 5, 7

[21] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118, 2007. 4, 5

[22] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. Cuda, release: 10.2.89, 2020. 5

[23] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. *ICCV*, 2021. 1, 2, 4, 5

[24] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *ACM Trans. Graph.*, 40(6), dec 2021. 1, 2, 4, 5

[25] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. *arXiv preprint arXiv:2011.13961*, 2020. 1, 2, 4, 5

[26] Liangchen Song, Anpei Chen, Zhong Li, Zhang Chen, Lele Chen, Junsong Yuan, Yi Xu, and Andreas Geiger. Nerfplayer: A streamable dynamic scene representation with decomposed neural radiance fields. *IEEE Transactions on Visualization and Computer Graphics*, 2023. 1, 2

[27] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5459–5469, 2022. 3

[28] Andrea Tagliasacchi and Ben Mildenhall. Volume rendering digest (for nerf). *arXiv preprint arXiv:2209.02417*, 2022. 3

[29] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020. 3

[30] Matthias Teschner, Bruno Heidelberger, Matthias Müller, Danat Pomeranets, and Markus Gross. Optimized spatial hashing for collision detection of deformable objects. *VMV'03: Proceedings of the Vision, Modeling, Visualization*, 3, 12 2003. 4

[31] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-NeRF: Structured view-dependent appearance for neural radiance fields. *CVPR*, 2022. 3

[32] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9421–9431, June 2021. 2, 4, 5

[33] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5438–5448, 2022. 3