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

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

1. Animated Blender Dataset

We show a more detailed view of our results on the animated Blender dataset in Suppl. Table 1. Visuals can found in Suppl. Figure 1.

2. Ablation Studies

Physics vs Adam In Suppl. Figure 3, we update the particle positions on the wheel dataset using Adam instead of using the PBD physics system. Our results show that the physics system produces higher quality reconstructions in dynamic scenes when compared to Adam.

Selecting the Gradient Scale Suppl. Figure 4 shows how the gradient scale α – used to integrate the positions gradients into the particle velocities – was chosen for the wheel dataset. A small α creates a noticeable drop in quality when the wheel begins to rotate at frame 100 because the particle velocities take longer to adjust to the required speed. A large gradient scale creates particle instability.

Long Running Scene In Suppl. Figure 5, the wheel experiment is run for around 2000 frames at $2.4°$/second. We observe a degradation in quality over time as a result of particles slowly losing neighbours during the movement. When particles lose all their neighbours, a degenerate case occurs which inhibits other particles from reassembling within each other's search radius. This can be addressed in future work by creating a particle growing strategy that detects this case and does not allow lone particles to exist. Note, however, that even at the lowest reconstruction quality, the wheel maintains its shape.

Full Scene We also visually demonstrate in Suppl. Figure 6 that ParticleNeRF can be used to reconstruct a full scene – albeit at a reduced reconstruction quality when compared to InstantNGP.

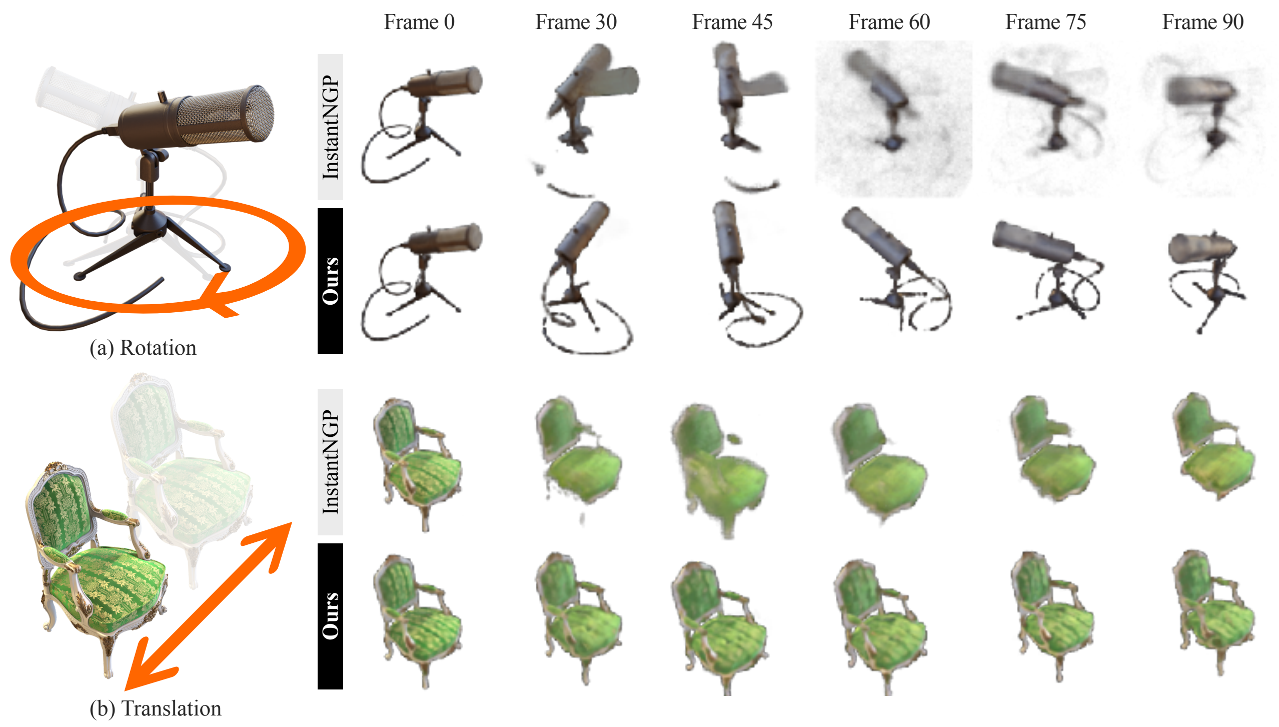
3. Implementation Details

InstantNGP uses an occupancy grid as an acceleration structure to skip evaluating points in free space. In the original implementation, this structure was updated every 16 training steps by evaluating the density at each vertex in the grid. This worked well for static scenes because it was expected that the densities of the points would converge to their values and not change thereafter. In our dynamic scenes, we update the occupancy grid at every training step for both InstantNGP and ParticleNeRF. This allows the acceleration structure to quickly change with the scene and therefore does not inhibit training in areas which were once unoccupied but have since become occupied.

Towards Realtime In our dynamic dataset, out of the 100,000 particles initialized in the scene, only around 7000 are actually contributing to the geometry of the object. If we prune away particles with a density less than a certain threshold, our system can run a single training step in under 6 ms. A pruning strategy however must be paired with a growing strategy so that particles can appear again when they are needed. A simple implementation of a growing strategy would be to replicate particles with less than a certain number of neighbours. The implementation of a pruning and growing strategy will be released with ParticleNeRF's codebase and a video of it in operation can be seen in the project website https://sites.google.com/view/particlenerf. However, making that strategy sufficiently robust for online usecases is left for future work.

The computational cost of updating the acceleration structure at each training step is high, particularly for a ParticleNeRF with a large number of particles, as shown in Suppl. Figure 7. To address this issue, an alternative acceleration structure that exploits the point-based nature of the encoding can be utilized. One such system is Nvidia's Optix framework which utilizes hardware raytracing cores to perform quick traversal of a scene.

Finally, we observe that the backward pass takes 5 times longer than the forward pass despite having a similar number of computations. The difference is due to the memory contention caused by multiple query points trying to up-



Supplementary Figure 1. We test our encoding on an animated version of the Blender dataset. **(a)** shows an object rotating around its up axis. **(b)** shows an object translating from side to side. InstantNGP cannot learn features fast enough to maintain a high quality reconstruction. ParticleNeRF is able to move its features in space and maintain the structure of the object.

Supplementary Table 1. Performance of InstantNGP and ParticleNerf on the Animated Blender Dataset reported through the mean and standard deviation of the photometric PSNR over 100 frames.

	Step per Frame	Model Encoding	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship
Static		Hash	**27.43**	**21.82**	**26.40**	**30.13**	**27.00**	**25.25**	**29.61**	**22.24**
		Particle	24.60	19.93	22.92	27.35	23.23	23.33	28.06	23.09
Rotation	1°	Hash	21.30 ± 2.3	18.72 ± 1.4	22.44 ± 0.9	25.71 ± 1.1	21.07 ± 1.4	21.28 ± 1.5	26.42 ± 1.0	22.71 ± 1.1
		Particle	**24.02 ± 1.4**	**19.70 ± 1.2**	**22.85 ± 0.7**	**27.02 ± 0.8**	**22.77 ± 1.1**	**23.23 ± 1.2**	**27.77 ± 0.5**	**23.04 ± 1.0**
	2°	Hash	18.20 ± 1.9	17.76 ± 1.3	21.80 ± 0.9	24.13 ± 1.2	19.25 ± 1.4	18.80 ± 1.5	24.07 ± 1.0	22.15 ± 1.2
		Particle	**23.02 ± 1.4**	**19.26 ± 1.2**	**22.23 ± 0.7**	**26.83 ± 0.8**	**22.01 ± 1.1**	**22.65 ± 1.2**	**26.97 ± 0.5**	**22.93 ± 1.0**
	3°	Hash	16.91 ± 1.7	16.92 ± 1.2	21.19 ± 0.9	23.22 ± 1.3	18.29 ± 1.5	17.20 ± 1.5	22.76 ± 1.4	21.64 ± 1.4
		Particle	**21.94 ± 1.5**	**18.72 ± 1.3**	**21.77 ± 0.8**	**26.15 ± 0.8**	**21.32 ± 1.1**	**22.11 ± 1.2**	**26.32 ± 0.6**	**22.64 ± 1.0**
	4°	Hash	16.56 ± 1.8	16.38 ± 1.0	20.66 ± 1.0	22.77 ± 1.4	17.65 ± 1.7	16.72 ± 1.5	22.99 ± 1.1	21.53 ± 1.3
		Particle	**21.16 ± 1.5**	**18.26 ± 1.2**	**21.13 ± 0.8**	**25.45 ± 0.9**	**20.73 ± 1.2**	**21.38 ± 1.3**	**25.28 ± 0.7**	**22.21 ± 1.1**
Translation	1 cm	Hash	22.60 ± 2.1	19.06 ± 1.5	22.13 ± 0.8	25.18 ± 1.7	21.40 ± 1.6	22.87 ± 1.6	27.06 ± 1.1	21.89 ± 1.4
		Particle	**24.20 ± 1.4**	**19.68 ± 1.3**	**22.43 ± 0.7**	**26.11 ± 1.5**	**22.85 ± 1.1**	**23.34 ± 1.3**	**28.03 ± 0.6**	**22.69 ± 1.2**
	2 cm	Hash	19.24 ± 2.1	17.68 ± 1.4	21.38 ± 0.9	22.97 ± 1.7	20.06 ± 1.6	21.07 ± 1.9	24.43 ± 1.3	21.41 ± 1.3
		Particle	**23.15 ± 1.3**	**19.31 ± 1.3**	**21.60 ± 0.7**	**25.38 ± 1.2**	**22.07 ± 1.1**	**23.06 ± 1.2**	**27.12 ± 0.7**	**22.71 ± 1.1**
	3 cm	Hash	17.90 ± 2.0	17.12 ± 1.3	**21.11 ± 0.9**	21.45 ± 1.8	19.37 ± 1.8	19.18 ± 2.3	23.47 ± 1.4	21.07 ± 1.4
		Particle	**22.15 ± 1.4**	**18.80 ± 1.2**	20.94 ± 0.7	**24.28 ± 1.3**	**21.29 ± 1.2**	**22.69 ± 1.2**	**26.16 ± 0.7**	**22.15 ± 1.2**

date the same set of neighbouring particles. This too can be alleviated in future work. This issue can be addressed by changing the ray sampling procedure. Instead of sampling rays from every camera in a training iteration, rays can be sampled from only one camera at a time. This minimizes the chance memory contention on the atomics. In this work, we chose to keep the training procedue as similar to InstantNGP's as possible.

4. SE(3) Invariance and Interpretability

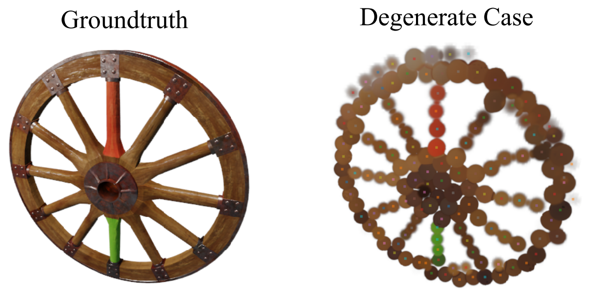
Our encoding is invariant under 3D rigid transformations. For a transformation $T \in \mathrm{SE}(3)$:

$$\mathbf{f}_j = F(\boldsymbol{x}_j, \mathcal{P}) = F(T\boldsymbol{x}_j, T \circ \mathcal{P}) \quad (1)$$

where

$$T \circ \mathcal{P} = \{(T\boldsymbol{x}_i, T\boldsymbol{v}_i, \mathbf{f}_i) : i = 1, 2, ..., M\} \quad (2)$$

Our particle encoding approach generates local features that are directly associated with local geometry patches. As a result of its SE(3) invariance, when these particles are moved or deformed, the underlying geometry will be similarly affected. For instance, if a subset of particles representing an object within a scene is moved or deformed, the corresponding geometry will change in predictable ways. In contrast, non-parametric NeRF methods encode geometry within the weights of a neural network. There is no natural transformation that can be applied to the weights that would change the scene in predictable ways. Additionally, voxel-based features, particularly those using a multi-level approach, cannot be easily transformed to modify the scene. The lack of controllability and interpretability over these features creates two challenges: (i) It is challenging to apply motion priors in dynamic scenes - for example in the case where a certain object's velocity is known. This limitation was also identified by the authors of TiNeuVox. (ii) The usefulness of NeRFs as a representation of geometry is limited when the underlying features lack the invariance property. Therefore, NeRFs without the invariance property are primarily suitable for view synthesis and not as a new method of geometry representation in downstream tasks.



Supplementary Figure 2. A degenerate case occurs when none of the particles have any neighbours within their search radius. The resulting reconstruction manifests as a set of spheres due to the inability of one particle alone to express more complicated geometries.

5. Degenerate Case

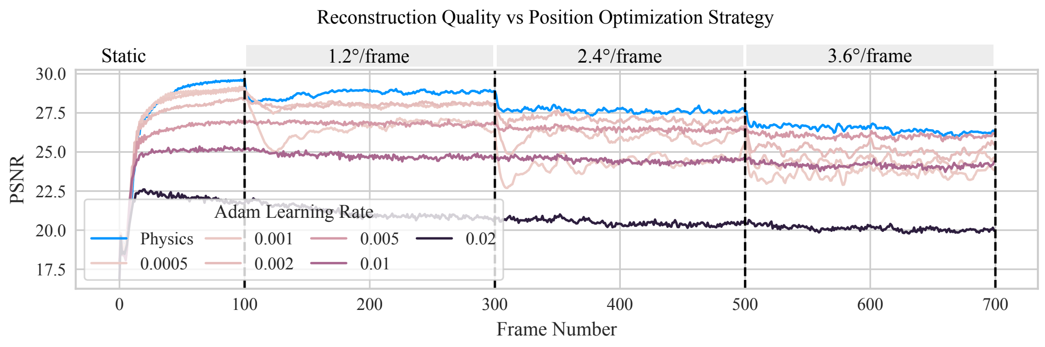
The interpolated features around a **lone** particle will be the same along the surface of a sphere centered on that particle. The effect is shown visually in Suppl. Figure 2 where we reduce the number of particles in the scene to 200 and ensure that every particle has no neighbours within its search radius. In our experiments, we used a sufficient number of particles when initializing the scene to circumvent the issue. Future work can develop a particle growing algorithm that detects lone particles and creates neighbours in their vicinity.

6. Timing Information

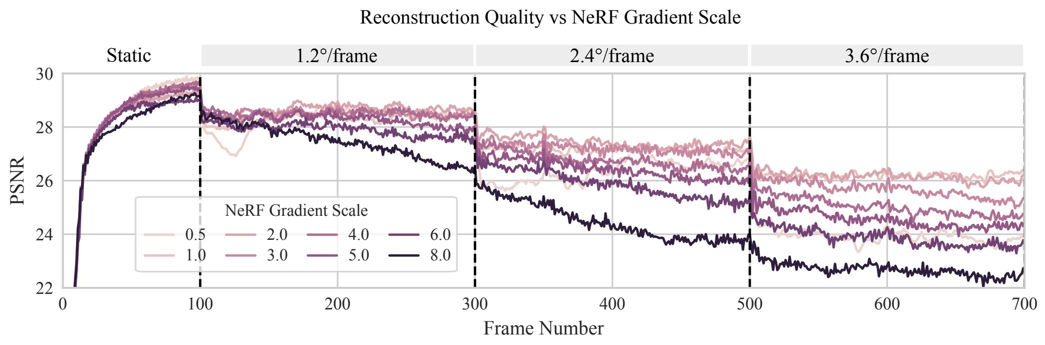
The training times per step are `[ParticleNeRF 40ms]`, `[InstantNGP 8ms]`, `[TiNeuVox-S 50ms]`, `[TiNeuVox 130ms]`. The 200 ms for ParticleNerf we mention in the abstract is the time taken for 5 steps which we found sufficient to adapt to changes in the scene. Further optimizations discussed in "Towards Realtime" (Section 3) achieve a training time of below 6 ms per step.

7. Typical Recovery

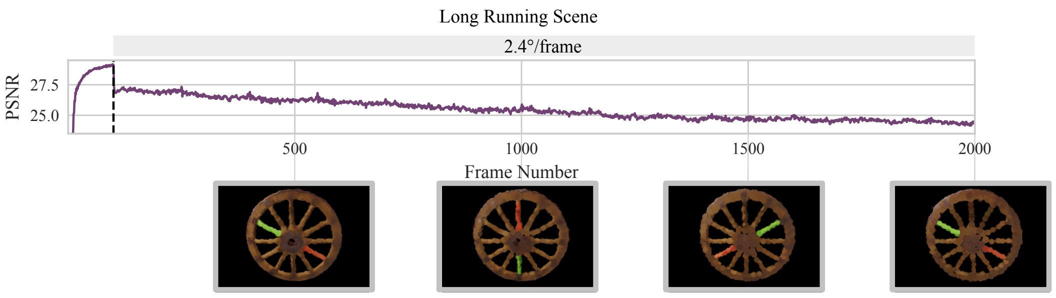
The training steps the baselines typically need to represent changes in the scene is dependant on the speed of the motion. We use the wheel scene from Fig. 6 which turns at successively faster rates to explore the average number of steps needed for each method to fully recover. ParticleNeRF only needs 5 training steps per frame for all speeds of the wheel. At low speeds, InstantNGP matches ParticleNeRF's reconstruction quality with 15 steps. At medium and high speeds, InstantNGP requires 20 and 40 training steps respectively. TiNeuVox requires 40 steps at all speeds.



Supplementary Figure 3. A comparison between using Adam and the physics system to update the particle positions after calculating their gradients relative to the NeRF reconstruction loss. Integrating the gradients with the physics system produces higher quality reconstructions on the wheel dataset whilst providing a well formulated means of adding constraints. Adam is configured with $\beta_1 = 0.9$, $\beta_2 = 0.99$ and a learning rate which is indicated in the legend.

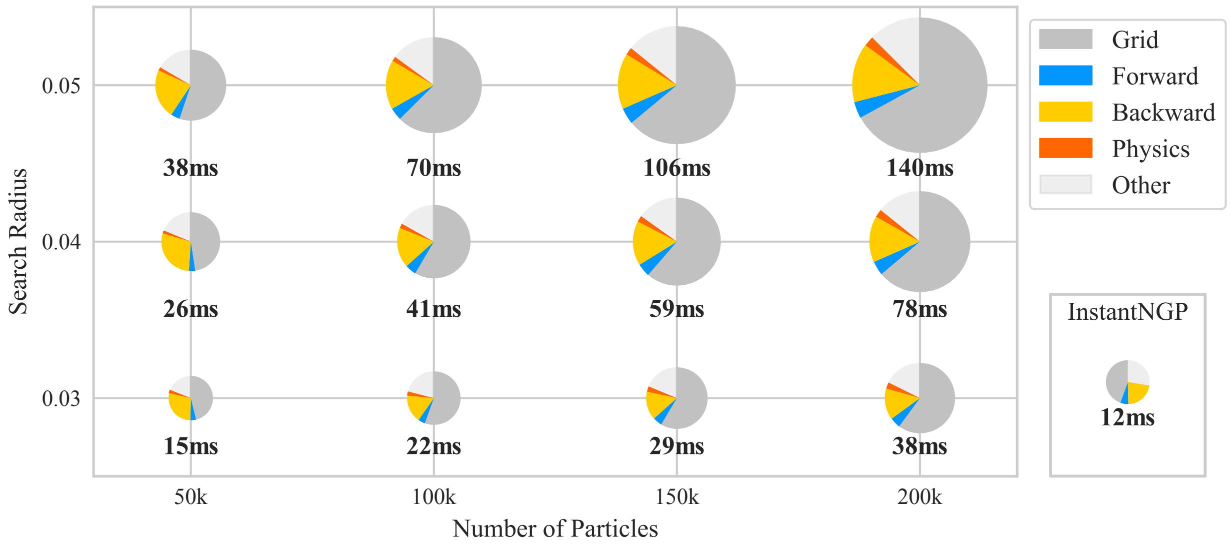


Supplementary Figure 4. An ablation performed on the wheel dataset with 100,000 particles and a search radius of 0.04 showing the impact of the gradient scale α used to combine the NeRF gradients with the particle velocities. Values around 2.0 have the best results.



Supplementary Figure 5. Figure showing degradation that occurs over a longer sequence of the Wheel dataset. The initial loss in quality is because the transition from static to 2.4 degrees is more sudden than the static to 1.2 degrees used in earlier experiments. Degradation occurs as particles begin to lose neighbours and are unable to acquire new ones.

Supplementary Figure 6. ParticleNeRF trained on a full scene after 3000 training steps with 200,000 particles at a search radius of 0.02.



Supplementary Figure 7. We profile ParticleNeRF and InstantNGP and show where the majority of the training time is spent per training step. The time taken by the acceleration structure (Grid), the forward pass, the backward pass, and the physics system are recorded for a range of configurations. InstantNGP is also shown on the right.