Faster Neural Scene Graphs with Temporal Information

Yeji Song Seoyoung Lee Yongil Kim Seunggeon Lee
Seoul National University
{ldynx, seoyoung1215, miles94, chong2}@snu.ac.kr

Abstract

Recent advances in neural rendering methods have demonstrated that it is possible to synthesize photo-realistic views for static and dynamic scenes with more than one transforming object. Based on the framework of NeRF (Mildenhall et al., 2020), current methods predict novel views from arbitrary directions and locations based on directionally emitted RGB values and their corresponding volume density, supervised by a set of 2D RGB images. However, learning such accurate views with these representations usually takes significant amounts of computation time as a procedure of optical ray marching yields each pixel value. The issue of time-consuming inference process intensifies for dynamically moving scenes with multiple objects because a considerable amount of costly ray marching is needed for every frame. In this work, we introduce a simple but novel method to incorporate temporal information shared across frames to generate neural scene graphs for dynamic scenes with multiple objects. Our method explores the trade-off between image quality and inference time and provides flexible adjustments between the two elements.

1. Introduction

Novel viewing synthesis is to predict and synthesize images from a new viewpoint using images of various viewpoints. As in *building Rome in a day*[1], 3D-reconstruction through Colosseum photos taken from multiple viewpoints is an example of representative view synthesis. Novel view synthesis is a long-standing issue in computer vision. Point clouds, discretized voxel grids, and textured mesh[4, 25, 19] are traditional methods to approach the task, but the considerable amount of voxels and meshes required to compose high-resolution images cause costly problems. Multiplane Image(MPI)[27] comes up again as it is more effective than previous methods to alleviate the cost problem and to represent reflective and transparent objects.

Also a growing body of research struggles to express illuminance and appearance following the viewing direction. Neural Radiance Fields(NeRF)[14] represents static scene representation, including illuminance effect by evaluating a density indicating transparency and a color that varies depending on the direction. The object's detail can also be expressed at a low cost via a 5D implicit function implemented with an MLP. NeRF in the wild (NeRF-W)[12] presents a novel scene synthesis of photographs taken at different times and various illuminance through three MLPs. NeRF-W makes a robust representation of the object regardless of datasets and increases versatility.

Furthermore, Neural Scene Graphs for Dynamic Scenes[16] enables novel view synthesis of multiple dynamic objects using a neural scene graph. They decompose a scene into dynamic and static components and train models to learn these representations respectively. Additionally, the transformation of dynamic objects can be represented by an affine matrix and readily combined with a static scene. However, Neural Scene Graph (NSG) suffers from the long inference time because it computes all sampled queries on sampled ray for each frame. The longer the video, the more time it takes at inference time, which hinders practical application.

In this study, we propose a method named Temporal Neural Scene Graphs (TNSG) that adjusts the trade-off between image quality and the inference time, therefore, allows a flexible application to the various environments. Our research aims to decrease unnecessary computational redundancy to render outputs with similar information. In general, the movement of objects does not differ significantly between contiguous frames. Accordingly, our method reuses the rendered information from previous frames to estimate the values of the next frames by binning and reusing values which share similar characteristics across the frames. Specifically, our method measures the probability of reusability by scoring each bin. By selectively using bins with high reusable potential, the deterioration of image quality is prevented.

Our method explores the trade-off between image quality and inference time through the bin reusing process and offers flexible adjustments between the two elements. Our method also finds the optimal trade-off, that is, greatly improving the inference time with minimal degradation on the

image quality.

As a consequence, our contributions can be summarized as follows:

- We reduce computation time for inference with minimal degradation on image quality. Our method also offers flexible adjustments between image quality and inference time along this optimal trade-off line.
- Utilizing the temporal information our method estimates the color and density of queries that are likely to have redundant information.
- By scoring the reusability of the sampled queries we can take a learnable approach of reusing temporal information.

2. Related Work

Recently, the advance in research on implicit or neural representations has enabled researchers to achieve photo-realistic views for both static and dynamic scenes, including single and multiple objects. However, training and rendering processes based on neural representations often require time-consuming ray marching at inference time. Though reduced inference time brings cost benefits, computation time should not be considered solely over high-quality scene representations. Maintaining high fidelity results should be a principal constituent that is ensured along with reducing computation time, especially for computation-heavy dynamic scene representations. Here, we review related work in the fields of static and dynamic scene representations, as well as recent attempts to reduce inference time or maintain high fidelity rendered results for neural representations. We want to emphasize that in our knowledge, there has been no research considering *both* inference time cost and high fidelity in multi-object dynamic scene representation.

2.1. Static Scene Representation

Representing scenes have been a long-standing task in traditional computer graphics and modern computer vision. **Discrete representations** Traditionally, discrete representations were used to predict the geometry and appearance from sparse view samples in 2D images. Some of the popular approaches include polygon meshes[21, 24, 2], point clouds, and voxel grids[7, 18]. A polygon mesh is a collection of vertices, edges, and faces that define the shape of a polyhedral object, and using faces consisting of triangles gives a triangle mesh. Simple convex polygons (n-gons) have an advantage that they simplify the rendering process. Although mesh and point clouds have flexibility in representing various geometries, their irregular geometry types hinder further applications. Volumetric representations, such as voxel grids, the 3D counterpart of pixel in 2D, can realistically represent complex shapes and materials. While discrete representations are differentiable and optimizable, they often require 3D supervision which may not be available in real-world applications[8] and are difficult to scale up to high resolution due to discrete sampling. **Implicit representations** In response to the limitations of discrete representations, recent directions in computer vision utilize implicit or neural representations for scenes and objects by encoding them in the weights of a multi-layer perceptron (MLP) that directly maps information of a 3D spatial location to a property of the scene or object. A neural representation uses level sets, a set of real-valued solutions of a function, to represent continuous 3D shapes. Various implicit methods, such as signed distance functions[6], 3D occupancy fields[13] and additional numerical methods[15], have been utilized thus far to represent complicated and high-resolution geometry but showed limitations of over smoothed renderings which limits the application to simple shapes.

In order to suggest a better implicit representation method, Mildenhall *et al.*[14] introduced neural radiance field (NeRF). NeRF implicitly represents a single object static scene using 5D radiance fields to render photo-realistic novel views for even complex objects with a MLP. Given a set of locations on a camera ray crossing a point on the 3D object and its corresponding viewing direction, NeRF returns the directional emitted color and volume density of the color. However, as a 2D view requires a ray computed for each pixel in the view, models based on NeRF can be computationally costly and time consuming. It is especially notable that NeRF requires a significant amount of inference time due to optical ray marching. In efforts to reduce inference time, some research has introduced means to express more information via another intermediary function. Wizadwongsa *et al.*[23] introduces NeX, which involve basis expansion on the pixel representation for each view, while Liu *et al.*[10] proposed Neural Sparse Voxel Fields (NSVF), which uses a set of sparse voxel-bounded implicit fields to achieve efficient rendering at inference time.

2.2. Dynamic Scene Representation

Research on learning efficient representations of scenes has not been limited to static and simple scenes. As previous research on static scene representations yielded low fidelity results with unexpected artifacts when applied to dynamic objects and scenes, additional means needed to be considered to account for transformations and dynamics of objects and scenes. Recently, efforts have been extended to dynamic scenes consisting of multiple objects. Ost *et al.*[16] specifically proposes a neural rendering method that decomposes dynamic, multi-object scenes into a learned scene graph with decoupled components accounting for dy-

Method	NSG[16]	matching	binning
FPS	0.296	0.010	0.340

Table 1: The comparison of inference speed(FPS) between NSG[16] and heuristic methods. We report inference time based on *0006 example* of KITTI MOT dataset.

namic scene representation and object transformation of each dynamic object.

As representing dynamic scenes requires more learnable components than static scenes, inference time used to compose neural representations of the scene background and each individual object total up to a greater burden. However, reducing inference time can be more valuable when high quality and accuracy in scene representations are simultaneously achieved.

One way to concurrently render high-fidelity representations of dynamic scenes is to use temporal information shared across frames. D-NeRF (Neural Radiance Fields for Dynamic Scenes)[17] by Pumarola *et al.* utilizes time as an additional input to NeRF to model the dynamics of a single object scene by simultaneously learning to encode the scene into a canonical space and then mapping the canonical representation into a deformed scene at a particular time point. Li *et al.*[9] uses Neural Scene Flow Fields to model dynamic scenes based on a single monocular video with a 3 component time-variant continuous function that considers appearance, geometry and 3D scene motion, enabling the synthesis of novel views for arbitrary view point and time.

With regards to preserving high quality results in a dynamically changing scene, we can consider consistency in time and view points. Mallya *et al.*[11] provides methods to produce RGB videos based on high-level semantic inputs, such as segmentation maps and depth maps. Its contribution is in generating videos with temporal consistency based on temporal memory while maintaining consistency in view points via neural rendering of simultaneous multi-views.

3. Method

3.1. Neural Scene Graphs

[16] proposes the perspective of viewing a dynamic scene as a neural scene graph with leaf nodes (dynamic objects) and edges (transformation of dynamic objects). They use a particular KITTI[5] vehicle tracking data and represent each leaf node as a neural radiance field. They also group leaf nodes hierarchically, combining similar appearances in the same class and at the same time distinguish individual objects via learned latent encoding vector. The outputs of each neural radiance field are the color and density of each object.

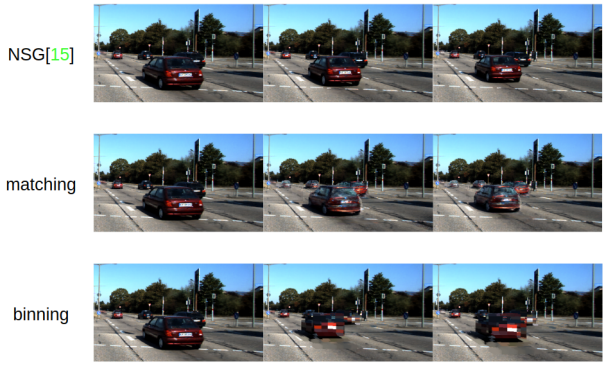

Figure 1: Visualized outputs of vanilla neural scene graphs[16] and heuristic methods. Naive matching harms the quality of output because color and density values are estimated from the previous frame.

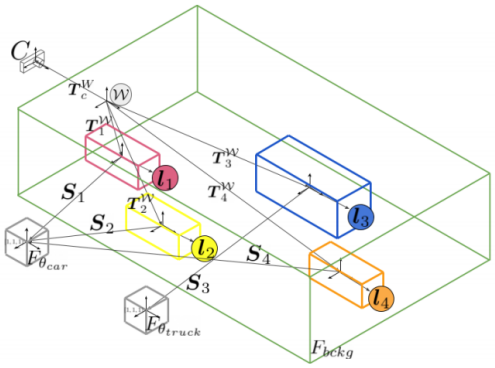

Figure 2: Visualized Neural Scene Graphs from the original paper. Figure is from [16].

$$[c(x_o), \sigma(x_o)] = F_{\theta_c}(l_o, p_o, x_o, d_o) \tag{1}$$

θ_c are weights shared between all objects in class c, and latent vector l_o is learned for individual representations. p_o refers to object location in the global frame to model the interaction between object and global illumination of a scene. Each x_o and d_o respectively refer to a 3D location in object coordinate frame and its viewing direction which are the same input given as the input of original NeRF.

Each directed edge connecting the root node ω as shown in Fig. 2 and each object represents the transformation between the global world space and local object as following:

$$x_o = S_o T_o^w x, \quad x_o \in [1, -1] \tag{2}$$

S_o and T_o^w refer to scaling and translation of the bounding box respectively. The inputs to the implicit function F_{θ_c} are changed according to the time and movement of the ob-

jects in class c. Neural scene graphs naturally decomposes scenes into dynamic and static scene components while the edges and nodes can be manipulated readily to create various novel dynamic scenes.

However, since a forward path should be computed for every frame, time-consuming ray-marching operation is also needed for every frame, thus hindering real-time applications of vanilla neural scene graphs.

3.2. Heuristic Faster Inference Methods

In order to improve the inference time of neural scene graphs, we implement naive ways to obtain some output values from the previous frame. Since the range of movement of each object in every frame is limited, it is highly probable that the color and density values of the current location or *queries* x_t and x_{t-1} are similar. From this insight, we reduce the inference burden by assuming some information are common between contiguous frames, and exploiting these common outputs from the previous frame to render the current frame.

Naive matching This approach follows that if the query x_t of the current frame exists within a predefined distance ball around the query x_{t-1} of the previous frame, we take the color and density value of x_{t-1} as those of x_t. If x_t is included in multiple balls, one of them is taken at random. However, although this method greatly reduced the number of queries fed into the model, significant overhead occurred in the process of calculating and matching the nearest distance balls, suggesting a need for a new method rather than naive matching.

Statistic binning We create a predefined number of bins for query x_{t-1} and classify the current query x_{t-1} to each bin where the corresponding color and density values are taken. The current query x_{t-1} that cannot be classified into the bins from the previous frame is directly fed into the implicit function and inference process.

However, these methods all tend to damage the rendering quality because reusing color and density values from the previous frame by a heuristic criterion induces difficulty in capturing the effect depending on the position of the object, such as global illumination or color change due to the direction of the object. We report the inference time of original neural scene graphs and other heuristic methods in Table 1 and the qualitative results in Fig. 1.

3.3. Temporal Neural Scene Graphs

Regarding Sec. 3.2, in terms of quality or memory, simply using the same color and density in a similar position from the previous frame cannot be the solution for a fast inference. Therefore, we propose a method that improves inference time and reduces quality degradation at the same time.

In Neural Radiance Field (NeRF), the model receives the position x and the viewing direction d and returns the color and density of the queried position and direction.

$$[c(x_o), \sigma(x_o)] = F_\theta(x, d) \tag{3}$$

In neural scene graphs, the model receives an additional object latent vector and object position as given in Eq. (1).

We propose Temporal Neural Scene Graphs that additionally returns the score value $s(x_o)$ which indicates the probability of the output of the current frame being used to compute the output of NeRF for the next frame.

$$[c(x_o), \sigma(x_o), s(x_o)] = F_{\theta_o}(l_o, p_o, x_o, d_o) \tag{4}$$

The color and density values with high reuse scores have a high probability to be used in the next frame. Therefore, they can be stored in the memory and reused when rendering the following frames. With our method, the model is expected to learn the geometry and color information of the scene, and simultaneously learn to pinpoint and extract the constant components shared among dynamic objects and scenes across a short continuous range of time. By adjusting the score threshold, reuse percentage can be controlled which provides the flexibility between time-quality trade-off.

Fig. 3. shows the overall pipeline of our method. We divide the spatial support set or query space $\mathfrak{X}$ of each neural radiance field representing an object, into three dimensional bins. We also create a hash table consisting of the object coordinate $(x, y, z)_o$ of the center of each bin and (r, g, b, σ, s) value stored in the bin. In the first frame, $(r, g, b, \sigma, s)_{x_1}$ can be obtained by feeding the sampled queries x_1 to implicit functions. We allocate each x_1 to the corresponding bins and the obtained $(r, g, b, \sigma, s)_{x_1}$ are used to update the values in the hash table. Bins without any corresponding queries x_1 are left empty.

In the second frame, sampled queries x_2 are slightly different from x_1 as the position of the object and the world camera have changed. However, assuming that there is not much movement of the object and the camera, most of the queries x_1 and x_2 are placed in similar positions. After allocating each x_2 to their corresponding bins, we could consider three cases for each $x_{2,j}$ ($x_{2,j} \in x_2, \forall j$): when the bin has the $(r, g, b, \sigma, s)_{x_{2,j}}$ value and the score s is higher than the score threshold (orange bins in Fig. 3), $(r, g, b, \sigma, s)_{x_{2,j}}$ is retrieved and reused. We refer this $(r, g, b, \sigma, s)_{x_{2,j}}$ value as $(r, g, b, \sigma, s)^r_{x_{2,j}}$. When the bin has a lower score s than the score threshold (green bins), $x_{2,j}$ and $d_{2,j}$ are fed to the implicit function to obtain $(r, g, b, \sigma, s)_{x_{2,j}}$ which is referred to as $(r, g, b, \sigma, s)^q_{x_{2,j}}$. When there is no $(r, g, b, \sigma, s)_{x_{2,j}}$ value in the bin since there was no query inside the bin in the first frame (grey bins), the same process as the second case is proceeded.

For rendering the second frame, $(r, g, b, \sigma, s)_{x_2} = (r, g, b, \sigma, s)^r_{x_2} \cup (r, g, b, \sigma, s)^q_{x_2}$ are integrated along the ray.

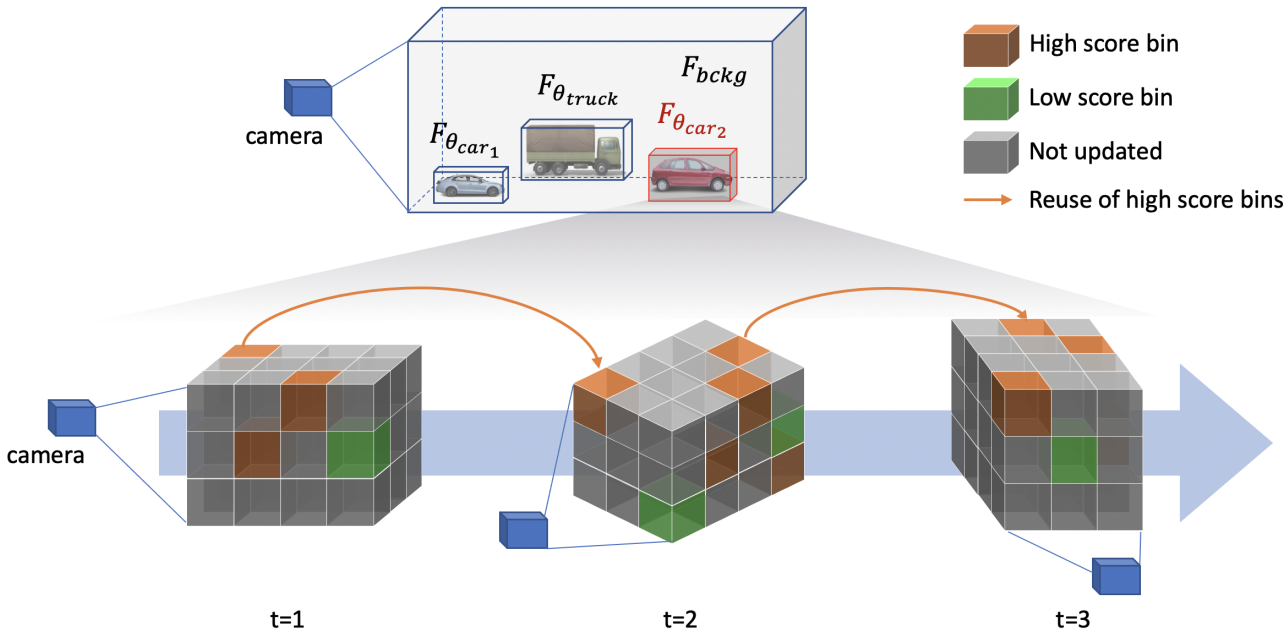

Figure 3: Neural radiance fields for each object are divided into three dimensional bins. When the bins have $(r, g, b, \sigma, s)_x$ value from previous frame and the score s is higher than score threshold (orange bins), $(r, g, b, \sigma, s)_x$ is retrieved and reused. When the bin has lower score s than score threshold (green bins) or when there is no $(r, g, b, \sigma, s)_x$ value in the bin since there was no query inside the bin in the previous frames (grey bins), x and d are fed to the implicit function to obtain $(r, g, b, \sigma, s)_x$.

Simultaneously $(r, g, b, \sigma, s)^q_{x_2}$ is used to update the hash table by allocating x_2 to the corresponding bins and storing the values in the bins. In the third frame, the updated hash table is utilized as $(r, g, b, \sigma, s)^r_{x_3} \cup (r, g, b, \sigma, s)^q_{x_3}$ are used to rendering the image, and the hash table is updated with $(r, g, b, \sigma, s)^q_{x_3}$. Reusing $(r, g, b, \sigma, s)^r_{x_t}$ at rendering the t-th frame, the number of queries fed to implicit functions is significantly reduced, thus reducing the inference time. When $(r, g, b, \sigma, s)^r_{x_t} \gg (r, g, b, \sigma, s)^r_{x_t}$ the inference time converges to the time used to allocate the points to the bins, which is minimal compared to the whole inference time without reusing.

If there are only few bins to be updated, our method would not effectively improve the inference time. Also, score s would not be trained properly. We observe the hash table after the whole training process, and we found that 99.8% of the bins are filled. This is consistent with our assumption that there is not much movement of the object from frame to frame and large redundancy would exist between frames. Also, as training progresses, the network captures the consistent parts of the object over time with higher confidence, increasing the score s. Therefore the reuse rate also increases, and a higher s used in the training process leads to more stable training.

In order to allocate only one $(r, g, b, \sigma, s)^q_{x_{i,j}}$ to each bin, we consider the length of bins from ray sampling, where each query is sampled through bin sampling on ray proposed by original NeRF. We used a bin size of 10. In addition, rendering output images with the reusing mechanism and training from scratch was detrimental to training an implicit function as the network could not learn (r, g, b, σ, s) elements jointly. That is, the network could not fully grasp the relation between (r, g, b, σ) and s only from rendered images. The network was confused at separating the object's color and density information from time flow information and created a blurred image with color trajectories following the object's movement. Therefore, we trained a network to some extent without the reusing process, then trained another number of iterations with the reusing process to ensure that the network could learn the object's color and density information first and then learn consistency information across time sequentially.

We define the loss in Eq. 5 as the total squared error between the reference values C and predicted color $\hat{C}$ rendered through ray marching where a batch of rays $\mathfrak{R}$ are sampled.

$$L = \sum_{r \in \mathfrak{R}} ||\hat{C}(r) - C(r)||_2^2 + \lambda \frac{1}{||S||_2^2} \tag{5}$$

Using only reconstruction loss would make all score values collapse to zero and the network would calculate all

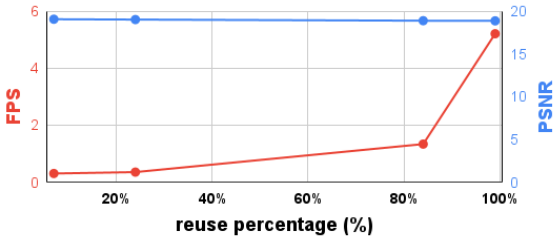

Figure 4: Trade-off between inference speed(FPS) and image quality(PSNR). FPS significantly improves as the reuse percentage increased, while the PSNR maintains a similar value.

queries without reusing, trying to achieve the high quality regardless of the inference time. Therefore we put a regularization term to ensure that certain parts of queries are to be reused. Through these two terms, the network learns the optimal trade-off between image quality and inference time, maintaining a high-quality image while utilizing the reusing process. λ is a coefficient of regularization term which is set to 1. S is a vector concatenating all the score s in a batch.

4. Experiments

This section provides validation of our proposed Temporal Neural Scene Graph method. We first show our model achieves optimal time-quality trade-off, greatly improves the inference time while showing minimal degradation on image quality. We also perform quantitative and qualitative comparisons of rendered images along with inference time analysis. Our model is trained using videos from KITTI object tracking dataset. Afterward, we compare our modelwith baseline methods, Neural Scene Graph[16] and NeRF[14]. NeRF is an implicit function that models a static scene without dynamic objects. Hence, NeRF is vulnerable to dynamic and multi-object scenes, and comparison with our model would not be appropriate. To compensate for this shortcoming, we also use a model called NeRF+time, which is essentially a NeRF model with a time variable as an additional input parameter to consider the dynamic transformation of a scene as time progresses.

KITTI dataset provides multi-object tracking information captured by two camera viewpoints, including multi-object labeling and calibration data. Specifically, we use the 0006 dataset in multi-object tracking evaluation, and time steps range from 65 to 120.The image size is 1242 x 675 pixels.

We trained all baselines for 730,000 iterations while our model was trained for 700,000 iterations and fine-tuned for 30,000 iterations using NVIDIA GeForce GTX 1080Ti.

Method	FPS
TNSG (99% reuse)	1.580
TNSG (7% reuse)	0.302
NSG[16]	0.296
NeRF[14]	0.074
NeRF + time	0.064

Table 2: The comparison of inference speed(FPS) between our TNSG and baselines. We report inference time based on the *0006 example* of KITTI MOT dataset.

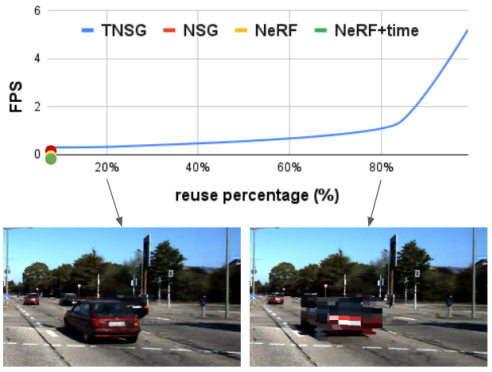

Figure 5: Reuse percentage vs FPS to show the trade-off between image quality and time. With high reuse percentage, TNSG can significantly improve the inference speed.

4.1. Time-Quality Trade-off

TNSG provides flexible adjustment between inference time and image quality while moving along the optimal trade-off line. As shown in Fig. 4, as TNSG puts more weights on the inference time, the improvement of speed is significant while degradation of the image quality is minimal. We could consider the situation where the inference speed takes priority and high-resolution images are not needed. With high reuse percentage, TNSG generates images or video clip in improved speed while sacrificing the minimal quality degradation. Likewise, when the image quality takes priority while inference time is not considered important, TNSG still can generate realistic images with a low reuse percentage.

4.2. Inference Time Analysis

We evaluated inference time of baselines for inference time comparison. We use the an additional reuse score and binning method to reduce the inference time effectively. Binning selectively uses the information of the previous frames. As shown in Table 2 and Fig. 5, the inference time of our model (99% reuse) is overwhelmingly improved compared to other models as expected. Our model

and Neural Scene Graph separate the background and dynamic objects to form a radiance field while NeRF delivers queries to the entire radiance field. NSG and our model take advantages in inference speed as they use a relatively small amount of queries concentrated on objects compared to NeRF that use large amount of queries scattered in the whole scene. Using reuse score and binning, TNSG efficiently decides which bins to use for the information inferred from the previous frames. Depending on the degree of reuse, the FPS is improved by about 5% to 500% compared to the Neural Scene Graph.

4.3. Quality Analysis

4.3.1 Quantitative Validation

We use the standard metric for quantitative evaluation: Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM)[22], Learned Perceptual Image Patch Similarity (LPIPS)[26]. PSNR represents the ratio of noise to the peak signal that a signal may have and is used for the purpose of evaluating loss information of an image generated in the image processing field. SSIM is designed to evaluate differences and similarities in human visual quality and used as an indicator that quantifies the effect of the degree of distortion for structural information in an image on perception. SSIM compares the luminance, contrast, and structure of the two images. LPIPS is defined as the distance between the two images using the network based on the assumption that the trained network has a lot of correlation with perceptual judgment. The lower the LPIPS, the better, and the higher the other two, the better. Moreover, we utilize tOF and tLP metrics[3] for temporal-consistency of reconstruction for adjacent frames. The tOF metric examines errors in motion of the reconstructed video by comparing the optical flow to the ground truth and the tLP metric examines the perceptual distance using LPIPS difference.

Table 3 summarizes the quantitative evaluation of our model and four baseline models. We quantitatively validate our method with comparisons against Scene Representation Networks (SRNs) [20], NeRF, and a modified variant of NeRF on novel scene and reconstruction tasks(NeRF+time), and NSG. For a fair comparison, baseline models are learned from the beginning and compared based on the results after 730,000 iterations. For SRN results, the reported value is used. Our model shows better performance than other models all metric except NSG, but also does not lag far behind in terms of performance compared with NSG. We note that our method can dramatically reduce the inference time compared to the existing baseline NSG, which supports that the trade-off of this slight degradation is sufficiently acceptable. The benefits of the increasing inference speed we are pursuing are much greater.

	TNSG	NSG[16]	NeRF[14]	NeRF+time	SRN[20]
PSNR↑	19.09	19.40	18.63	15.43	18.83
SSIM↑	0.708	0.807	0.684	0.369	0.590
LPIPS↓	0.243	0.154	0.265	0.427	0.456
tOF↓	2.946	2.737	4.383	13.84	-
tLP X 100↓	2.464	2.249	4.599	23.56	-

Table 3: Quantitative Results. We report PSNR, SSIM, LPIPS, tOF and tLP results on scenes from KITTI Dataset for SRN, NeRF, a modified NeRF variant with an added time input(NeRF+time), neural scene graph(NSG) and our method TNSG. For PSNR and SSIM, higher is better; for LPIPS, tOF and tLP lower is better. It can be seen that our model shows better performance than all baseline models except NSG for each metrics, and also shows no significant degradation compared to NSG.

4.3.2 Qualitative Results

For qualitative computation, we trained the same image sequence for all methods. We exclude some frames for image sequence, and evaluate that each method can reconstructs those frames. The results of the qualitative comparison are shown in Figure 6.

NeRF renders ghosting and blurred images because a slight change in the camera's pose causes a difference in the viewing direction. NeRF+time reduces the ghosting of the image, but still lacks detail and suffers from blurry, uncertain predictions. Neural Scene Graph synthesizes images reflecting dynamic objects and static backgrounds without changing the viewing direction appropriately. It also effectively aligns shadows and light reflections in the image. TNSG generated images as clear as NSG while inference time is much faster. This is due to the use of reuse scores and binning of TNSG which efficiently allocates queries for reusing and other queries for inference. Queries in remote objects have high reuse scores because there is little change in pixel or density values for a distant object, even with a large movement. On the other hand, queries in close objects have low reuse scores as outputs of the implicit function are sensitive to movement.

In Figure 6, TNSG has no object ghosting and blurring that is previously observed in NeRF and NeRF+time. A remote red car is relatively consistent across time and this causes active use of binning without compromising image quality much. A close object such as the dark red car in the front has less binning and maintains high quality using color and density value calculated from the network at each frame. As in Fig 4, PSNR value is relatively maintained with the increase in FPS.It can be inferred that TNSG restricts the information lost due to the higher reuse percentage finding efficient allocation of queries and avoiding large degradation of image quality. As a result, TNSG generates images of comparable quality compared to NSG.

Figure 6: Qualitative results on reconstruction of a scene from the KITTI dataset. TNSG maintains comparable high quality as NSG at the same time greatly improves the inference time. TNSG efficiently allocate queries for reusing and for inference such as queries in a remote object have high reuse scores while queries in a close object have low reuse scores.

5. Conclusion

Neural Scene Graph (NSG) tackles the challenge of representing dynamic, multi-object scenes synthesizing novel view video and novel scene reconstructed video with high quality. But the inference time is costly because NSG computes all sampled queries on sampled ray every frame. From the observation that there is not much movement of the object between consecutive frames, we proposed TNSG distributing high score for consistent values across time and efficiently reusing values from previous frames based on the score. Instead of feeding all queries to implicit function, TNSG forwards necessary queries and greatly improves inference time. By adjusting the score threshold, TNSG explore the trade-off between quality and inference time and flexibly weight more necessary elements depending on the situation. However, if the object or the camera moves rapidly, there is a limitation of our method that the inference time cannot be significantly reduced as the consistent part across time would be minimal. Also, when there are large number of objects, the memory cost due to binning would increase linearly. For the future work, as it is important to determine the score threshold, score threshold could be learned during training or determined by scheduling such as a gradual warm-up.

References

[1] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):105–112, 2011. 1

[2] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 425–432, 2001. 2

[3] Mengyu Chu, You Xie, Jonas Mayer, Laura Leal-Taixé, and Nils Thuerey. Learning temporal coherence via self-supervision for gan-based video generation. *ACM Transactions on Graphics (TOG)*, 39(4):75–1, 2020. 7

[4] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017. 1

[5] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012. 3

[6] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, Thomas Funkhouser, et al. Local implicit grid representations for 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6001–6010, 2020. 2

[7] Kiriakos N Kutulakos and Steven M Seitz. A theory of shape by space carving. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 1, pages 307–314. IEEE, 1999. 2

[8] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Transactions on Graphics (TOG)*, 37(6):1–11, 2018. 2

[9] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6498–6508, 2021. 3

[10] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *arXiv preprint arXiv:2007.11571*, 2020. 2

[11] Arun Mallya, Ting-Chun Wang, Karan Sapra, and Ming-Yu Liu. World-consistent video-to-video synthesis. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VIII 16*, pages 359–378. Springer, 2020. 3

[12] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7210–7219, 2021. 1

[13] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019. 2

[14] 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 *European conference on computer vision*, pages 405–421. Springer, 2020. 1, 2, 6, 7

[15] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3504–3515, 2020. 2

[16] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2856–2865, 2021. 1, 2, 3, 6, 7

[17] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021. 3

[18] Steven M Seitz and Charles R Dyer. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 35(2):151–173, 1999. 2

[19] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2437–2446, 2019. 1

[20] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *arXiv preprint arXiv:1906.01618*, 2019. 7

[21] Michael Waechter, Nils Moehrle, and Michael Goesele. Let there be color! large-scale texturing of 3d reconstructions. In *European conference on computer vision*, pages 836–850. Springer, 2014. 2

[22] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003. 7

[23] Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. Nex: Real-time view synthesis with neural basis expansion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8534–8543, 2021. 2

[24] Daniel N Wood, Daniel I Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H Salesin, and Werner Stuetzle. Surface light fields for 3d photography. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 287–296, 2000. 2

[25] Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. *arXiv preprint arXiv:1612.00814*, 2016. 1

[26] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 7

[27] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018. 1