3D-MOT using Neural Radiance Fields

Burak Cuhadar
Technical University of Munich
03720534
burak.cuhadar@tum.de

Abstract

In this project, we focus on 3D Multi-object Tracking problem where we aim to reconstruct the scene as well. To achieve that, we utilize the Neural Radiance Fields(NeRF) method which proved successful in recent years for the novel view synthesis task. However, the assumption for the NeRF method is that the scene is static. Therefore, it is not suitable for the 3D-MOT problem where there are dynamic objects in the scene. We assume there are only rigidly moving objects and propose to estimate the rigid pose of the objects jointly with the NeRF networks. In particular, we optimize different networks for each object and for the static background. We do not utilize any labeled data, but we use the ground truth object poses for a noisy initialization. On our synthetic dataset, we demonstrate our method can reconstruct the scene and estimate the object poses jointly. We also show that the objects and the scene are decomposed successfully.

1. Introduction

In recent years, the community has investigated implicit representations for scene reconstruction and novel view synthesis after the advent of Neural Radiance Fields(NeRFs) [7]. There have been many works trying to apply the NeRF method to different scenarios. One particular scenario where it is challenging to apply NeRF is found when the scene is dynamic because the basic assumption of NeRF is that the scene is static. However, there are already many works that have adapted NeRF for dynamic scenes [5, 11, 13, 18]. Some works that deal with dynamic scene reconstruction using NeRFs, also aim to decompose the scene into its individual objects [8, 10, 24, 25]. This object decomposability is especially useful for autonomous driving or mobile robotics scenarios, as those scenarios require distinguishing the individual objects in the scene such as cars, pedestrians etc. to be able to track their position and also make predictions about their motion.

In this work, we are interested in applying NeRF for a dynamic scene in an autonomous driving scenario, where we also aim to decompose the scene into the static(background) part and the dynamic parts. Each observed car is represented individually so that we can track them and also generate novel trajectories for each car. We utilize the method presented in "STaR: Self-supervised Tracking and Reconstruction of Rigid Objects in Motion with Neural Rendering" paper [25]. Differently from the STaR paper, we use our own dataset generated with CARLA simulator [1] and also apply the method for more than one object.

2. Related Work

NeRFs for Dynamic Scenes There have been many works that try to adapt the NeRF method for dynamic scenes, where there can be any non-rigid deformation observed on the scene. One such prominent example is called Nerfies [11]. They reconstruct dynamic scenes by optimizing a volumetric deformation field. Based on a frame-specific latent deformation code, the deformation field is optimized to warp observed points into a canonical 5D NeRF. Building upon the Nerfies method, HyperNeRF [12] lifts the NeRFs into a higher dimensional space and slice it depending on the latent deformation code specific to the frame by slicing surface field which again is modeled by an MLP, to allow discontinuities in the deformation field.

NeRFs for Scene Decomposition For some applications, decomposing the scene into its static and dynamic parts or into the observed objects in the scene may be desired. For example, a method called D2NERF [21] aims to decompose the observed scene from a monocular video into dynamic and static parts in a self-supervised fashion. The authors achieve it by representing static and dynamic parts seperately by two different NeRFs, where HyperNeRF represents the dynamic part [12] to model temporal changes. They also present a novel loss to aid the optimization for better decomposition. Another approach called Unsupervised Discovery of Object Radiance Fields(uORF) aims to decompose the scene into its background and objects from a

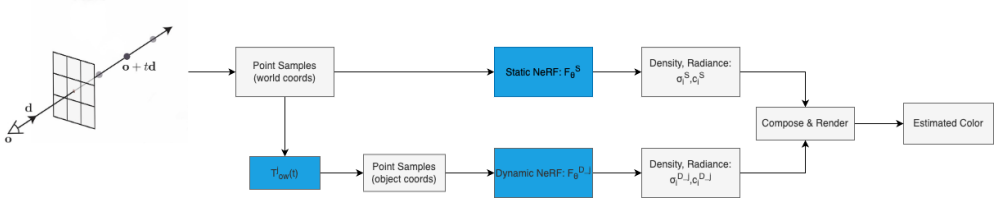

Figure 1. **Method Overview:** Blue color denotes the learnable parameters

single image by leveraging NeRFs and slot attention to infer object latent codes [24]. Similar to that work, ObSuRF [14] utilizes attention and NeRFs to decompose the scene from a single image. Differently from uORF, ObSuRF also uses a novel loss to use the depth information for faster training. All the methods mentioned above that decompose the scene into its objects work on static scenes where the objects do not move. The method called STaR [25] assumes there are only rigid objects that move between the frames where the rigid objects can be tracked and reconstructed with NeRFs in a self-supervised way. They represent the background and the object with individual NeRFs and allow the object's relative pose between the frames to be optimized on the Lie algebra. We build upon this work and going to investigate it in more detail in Section 3.

Object-aware NeRFs for Urban Scenarios Another line of work focuses on reconstructing urban driving scenes where the individual vehicles can be reconstructed independently. For example, in the Neural Scene Graphs [10], the scene is rendered using one NeRF for static background and one NeRF for the vehicles conditioned on object-specific latent codes. However, that method requires the scene graph of the scene, which means extensive labeling is required. Using only camera images and off-the-shelf algorithms to predict object tracks, camera poses and 2D image segmentations; Panoptic Neural Fields [4] has the capabilities of semantic and panoptic segmentation, object decomposition, and object bounding box optimization which the Neural Scene Graphs method lacks. Since we generally need large sclae reconstruction for urban scenarios, the SUDS method [18] can successfully reconstruct large-scale urban dynamic scenes into its static part and instances by using RGB images, sparse LiDAR and off-the-shelf 2D optical flow estimation methods.

Object-centric Learning With the proven success of the Attention [19] mechanism across many fields, there also emerged new kinds of methods for unsupervised object-centric learning based on Slot Attention [6]. In that work, the authors propose to encode the images using the Slot Attention module which uses self-attention to encode the image into distinct slots each explaining a part of the image, ideally the background or the objects. That way, the objects are discovered in unsupervised way. Based on this slot attention module, ObSuRF uses NeRFs as the decoder to render the objects and the background so that we get a radiance field representation of the scene and the individual objects [15].

3. Method

Our method is based on the Neural Radiance Fields [7] and STaR [25] methods. Firstly, we are going to explain the NeRF method as the preliminaries in Section 3.1 and then how we adapted it for dynamic scenes where there are rigid objects in motion in Section .

3.1. Neural Radiance Fields(NeRFs)

In NeRFs, the scene is represented as a 5D vector-valued function whose input is the 3D location and 2D viewing direction. The output of this function is 3D RGB color and volume density σ. The function is approximated by an MLP network $F_\theta : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$, but in practice two MLPs are optimized where one of them is used for sampling points on the ray for the second one. The color for each pixel is rendered by using quadrature rule on volumetric rendering equation [7]:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))\mathbf{c}_i \,,$$

$$\text{where } T_i = \exp(-\sum_{j=1}^{i-1} \sigma_j\delta_j).$$

Since generally much of the space if empty or occluded, naively sampling along a ray repeatedly would result in an inefficient way to optimize this network. To alleviate this, the authors also proposes a hierarchical volume sampling where for one network the samples on the rays are uniform and for the other network more samples are used in addition to uniform samples where those additional samples come from a distribution defined by the densities estimated by the first network. In addition to this, positional encoding is also leveraged to be able to more easily approximate a high frequency function such as the one we want to approximate.

3.2. 3D MOT using NeRFs

One of the assumptions of the NeRF method is that the scene is static and does not change from frame to frame. To be able to apply NeRF for a dynamic scene and also to track 3D rigid objects in the scene through time, we leverage the ideas presented in the Star and D2NeRF papers [25], [21].

We represent the static scene which does not contain any of the moving rigid objects and does not change from frame to frame as in the original NeRF method. To represent dynamic parts of the scene (individual vehicles in our dataset), we again user the vanilla NeRF method, but transform the input point samples by the optimized poses for each car before inputting it to the network:

$$F_\theta^S : \mathbf{r}(s_i), \mathbf{d} \rightarrow \sigma_i^S, \mathbf{c}_i^S \tag{1}$$

$$F_\theta^{D_j} : \mathbf{r}(s_i), \mathbf{d}, \xi_j(t) \rightarrow \sigma_i^{D_j}, \mathbf{c}_i^{D_j}; j \in V \tag{2}$$

where V is the predefined number of rigid dynamic objects, $\xi_j(t) \in \mathfrak{se}(3)$ is the transformation from time t to time 0, for the vehicle j. Therefore we treat the first frame as the canonical frame for the dynamic parts, so that the static and dynamic NeRFs are aligned. The transformation of samples is then done by $\exp(\xi_j(t))\mathbf{r}(s_i)$, which is given as the input to the dynamic NeRF networks.

To render RGB color, we use the alpha-blending approach presented in the NeRF paper by composing the outputs static and dynamic NeRFs:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(\alpha_i^S \mathbf{c}_i^S + \sum_{j=1}^{V} \alpha_i^{D_j} \mathbf{c}_i^{D_j}) \tag{3}$$

$$\text{where } T_i = \exp\left(-\sum_{j=1}^{i-1}(\sigma_j^S + \sum_{k=1}^{V} \sigma_j^{D_k})(s_{j+1} - s_j)\right) \tag{4}$$

$$\text{and } \alpha_i^S = 1 - \exp(-\sigma_i^S(s_{i+1} - s_i)), \tag{5}$$

$$\alpha_i^{D_j} = 1 - \exp(-\sigma_i^{D_j}(s_{i+1} - s_i)). \tag{6}$$

To be able to use (3) for color composition, we need to have the same samples for both static and dynamic NeRFs. Therefore during importance sampling [7], we use $\sigma_j^S + \sigma_j^D$

3.2.1 Loss Functions

Our total loss function is the following:

$$\mathcal{L} = \mathcal{L}_{RGB} + \beta\mathcal{L}_{transparency} + \gamma\mathcal{L}_{decomposition} + \eta\mathcal{L}_{static} + \lambda\mathcal{L}_{ray}$$

The first term is the MSE loss used in NeRF, which is the squared difference between estimated RGB and ground truth RGB values.

Transparency Regularization Following STaR [25], we regularize the estimated transparencies of the static and dynamic NeRFs to be close to 0 or 1 using entropy:

$$\mathcal{L}_{transparency} = \sum_{i=1}^{M}(\mathcal{H}(\alpha_i^S) + \sum_{j=1}^{V} \mathcal{H}(\alpha_i^{D_j})), \tag{7}$$

where $\mathcal{H}$ is the binary entropy function.

Decomposition Regularization The STaR method uses a regularization term to encourage a disentagled decomposition of the scene. It is achieved by a loss term that disencourages the static and dynamic volumes having large occupancies at the same time:

$$\mathcal{L}_{decomposition} = \left(\overline{\alpha}_i^S \log \overline{\alpha}_i^S + \overline{\alpha}_i^D \log \overline{\alpha}_i^D\right)(\alpha_i^S + \alpha_i^D), \tag{8}$$

where $\overline{\alpha}_i^S = \alpha_i^S/(\alpha_i^S + \alpha_i^D)$.

Static Regularization The InfoNeRF method [2] defines the entropy of a single ray using the estimated density for the samples on the ray:

$$\mathcal{L}_{static} = -\sum_{i=1}^{N} p(\mathbf{r}_i) \log p(\mathbf{r}_i) \tag{9}$$

$$\text{,where } p(\mathbf{r}_i) = \frac{\alpha_i}{\sum_j \alpha_j} = \frac{1 - \exp(-\sigma_i\delta_i)}{\sum_j 1 - \exp(-\sigma_j\delta_j)}. \tag{10}$$

Then they add this term to the loss function to penalize density distributions that are not focused on a single location. They also mask the non-hitting rays according to the estimated opacity before using this term for all rays [2].

Ray Regularization To prevent density floaters in the dynamic NeRFs, we leverage the ray regularization loss proposed in D^2NeRF [21] and define for the object j:

$$\mathcal{L}_{ray}^j(\mathbf{r}) = \max_{t \in [t_n, t_f]} w^j(\mathbf{r}(t)) \tag{11}$$

$$\text{,where } w^j(\mathbf{x}) = \frac{\sigma^{D_j}(\mathbf{x})}{\sum_i \sigma^{D_i}(\mathbf{x}) + \sigma^S(\mathbf{x})} \in [0, 1] \tag{12}$$

Then $\mathcal{L}_{ray}$ is defined as the mean of $\mathcal{L}_{ray}^j$s over all objects and ray samples.

Dynamic Regularization We also tried the dynamic regularization term used in EmerNeRF [22] paper, which is simply the mean of the dynamic NeRF densities. However, our experiments showed that this term couldn't be minimized during the training. Therefore, we do not include this in our final implementation.

3.2.2 Optimization

Rigid pose optimization The STaR method optimizes the poses using an analytical Jacobian that can be computed during the forward pass. However, in this work we use the PyPose library [20], which provides LieTensor modules to allow the computation of gradients for $\mathfrak{se}(3)$ algebra.

We follow the 3-stage optimization presented in STaR [25]: Appearance initialization, optimization for the first k frames, and online optimization.

Appearance Initialization To provide an initialization for our method, we train the static NeRF using images from only the first frame, until the RGB MSE loss for the fine network reaches the threshold m_1.

Pose Initialization During our experiments, we noticed that initializing object poses with identity does not lead to convergence. Therefore, we initialize the object poses with ground truth poses with added noise. To achieve that, we sample from the normal distribution with a mean of 0 and a standard deviation of 1 meter and add that to the ground-truth poses. For noisy rotation, we sample from the normal distribution with a mean of $\pi/32$ radians and a standard deviation of $\pi/16$ radians and add that to the y-axis angle.

Online Training After appearance initialization training, we start optimization of the poses jointly with the NeRF parameters. We first allow the optimization of the first k poses, and after fine NeRF MSE loss reaching a certain threshold m_2, we start to increment k one by one. We increment k when, again, the fine MSE loss reaches m_2. However, when the dynamic objects are much smaller than the static scene, the m_2 threshold is reached too quickly before the poses are optimized properly. Therefore, we also set a minimum number of iterations to be trained for before incrementing k, which we call N_{online}

4. Experiments

In our experiments, we want to test the success of our method in tracking 3D objects jointly with radiance field optimization. To this end, we show our results for scene reconstruction, novel view synthesis, 2D IOU for object decomposition between frames, and 3D IOU for object pose tracking.

Implementation Details We use PyTorch and PyPose to implement our method. For optimization, we use the Adam optimizer [3]. We list the hyperparameters used during our experiments in Table 1. Our NeRF implementation follows NeRF-Pytorch [23], and [9] for the MLP implementation. We use 256 samples for uniform sampling and 256 samples for importance sampling. For further details, we provide the implementation for our method: https://github.com/burakcuhadar/3D-MOT-using-Neural-Radiance-Fields.

k	m_1	m_2	N_{online}	β	γ	η	λ
5	9e-4	1e-3	70K	1e-3	1e-3	1e-5	1e-5

Table 1. Hyperparameters

Dataset We created synthetic datasets for evaluation using CARLA simulator [1]. We have both one-object and two-object datasets where the objects move between frames. In both datasets, we have 50 camera views for training, 6 for validation, and 12 for testing. In the one-vehicle dataset, we have a video of 16 frames; in the two-vehicle one, we have 12 frames.

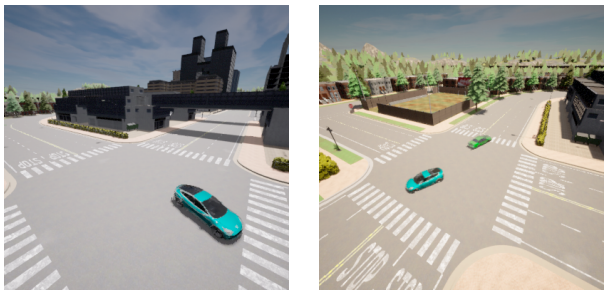

Figure 2. Example views from our dataset

Baseline Similar to STaR [25], we implement a NeRF method for the baseline where the network gets the normalized timestamp as an input in addition to the view direction and 3D location. This provides a basic NeRF method that can be generalized to dynamic scenes. We call it NeRF-time in the tables.

4.1. 4D Novel View Synthesis Evaluation

We compare our method with the Nerf-time baseline and our STaR reproduction on the novel view synthesis task. In Figures 6 and 7, we first report the renderings across different timesteps from a given camera view and then report the renderings for a fixed timestep from different camera views, respectively. In Figure 6, our STaR reproduction fails to provide realistic renderings from novel views. We can see that the rendered cars contain a lot of artifacts, and along their trajectories, we observe artifacts of the same color as the cars. The reason is that the static and dynamic NeRFs could not be decomposed successfully. Comparing our method with Nerf-time, we also see that some details of

the cars can be rendered better by our method, such as the bottom parts of the cars. This is especially apparent when we magnify the renderings of the vehicles in Figure 8. In the second row of the figure, we observe that the car is rendered more realistically than the Nerf-Time and STaR methods.

In Figure 7, the difference between our method and Nerf-time is more pronounced for car renderings. Our method can render cars with more detail and fewer artifacts. In these renderings, we also again see that our StAR reproduction performs poorly, meaning that our additional regularizers help significantly.

Quantitave evaluation results are reported in Table 2. We measure the metrics first on the final renderings and then only on the scene's static and dynamic parts separately. We observe that the Nerf-time baseline performs better than our method regarding the PSNR metric on the final rendering and the static parts but ours performs better than Nerf-time in terms of SSIM and LPIPS in the one-vehicle dataset. In the dynamic part evaluation, our method is better than the baseline regarding PSNR and SSIM in the one-vehicle dataset but worse than that in the two-vehicle dataset regarding the PSNR metric. We also observe that our STaR reproduction performs poorer than ours on these metrics.

4.2. Relative Object Pose Estimation Evaluation

We report the estimated relative pose of the vehicles with respect to the first frame. For that, we use the ATE and RPE metrics [16]. In addition, we also transform the ground truth bounding box of the first frame with our estimated relative poses and compare it against the ground truth bounding box in the subsequent frames to measure 3D IOU and report the mean in Table 4.

	One-vehicle	Two-vehicle		
		Mean	First car	Second car
ATE	0.146	0.424	0.540	0.308
RPE	0.182	0.769	1.271	0.267

Table 3. Pose Evaluation Metrics

We also provide visualization of the estimated trajectory against the ground truth one and the initialized trajectory in Figure 3. Although the pose initialization is very crude, we can successfully optimize the poses converging closely to the ground truth poses.

Mean	First Vehicle	Second Vehicle
0.924	0.932	0.917

Table 4. 3D IoU Comparison for two-vehicle dataset

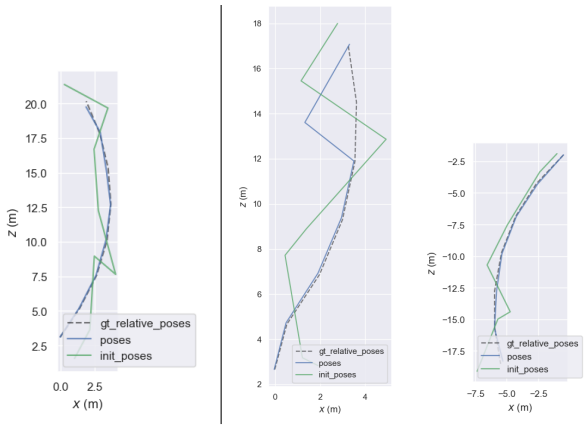

Figure 3. **Comparison of the ground truth trajectory with the estimated trajectory and the initialization trajectory.** Left: one-vehicle dataset. Right: two-vehicle dataset.

4.3. Object Decomposition Evaluation

To measure the object decomposition ability of our method, we get a 2D mask of the objects from their NeRFs using the estimated densities. In particular, we assume the pixel is occupied if the cumulative transmittance of the last sample on the ray going through that pixel is less than 0.1. We report the Intersection over Union(IoU) between the ground truth object mask and the predicted object mask in Table 5. In addition, we provide some examples for our object mask against the ground truth object mask in Figure 4. We can also render each NeRF individually to get the static background and separate moving vehicle renderings. Figure 5 shows some examples of those decomposed renderings.

	One-vehicle	Two-vehicle
2D IOU	0.79	0.67

Table 5. Object Decomposition Evaluation

4.4. Ablation

4.4.1 Regularization Terms

We provide our ablation study for the regularization terms in Table 6. As suggested in STaR [25], we use the regularization terms defined by Equations 7 and 8. We denote the experiment where we only use those entropy regularization terms in the first line of each section in Table 6. As a result of those experiments, we see that the ray regularization(Equation 11) and static regularization(Equation 9) terms improve the metrics of our method. The dynamic regularization term did not have any positive effect. Our experiments show that combining entropy regularization with static and ray regularization gives us the best method.

	Sequence	One-vehicle			Two-vehicle		
Composition	Metric	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
	NeRF-time	**26.29**	0.869	0.321	**23.73**	**0.833**	**0.313**
	STaR [25]	25.98	0.871	0.312	23.40	0.818	0.333
	Ours	26.23	**0.874**	**0.306**	23.65	0.829	0.314
	Sequence	One-vehicle			Two-vehicle		
Static	Metric	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
	NeRF-time	**26.55**	0.871	0.316	**23.81**	**0.834**	**0.307**
	STaR [25]	26.32	0.873	0.306	23.65	0.821	0.322
	Ours	26.43	**0.875**	**0.302**	23.75	0.830	0.308
	Sequence	One-vehicle			Two-vehicle		
Dynamic	Metric	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
	NeRF-time	17.64	0.596	**0.004**	**19.62**	0.659	**0.006**
	STaR [25]	17.14	0.583	0.055	15.98	0.492	0.056
	Ours	**18.58**	**0.665**	0.033	19.31	**0.661**	0.045

Table 2. **Quantitative comparison of our method to baselines in novel-view synthesis task:** These results are measured on our test views.

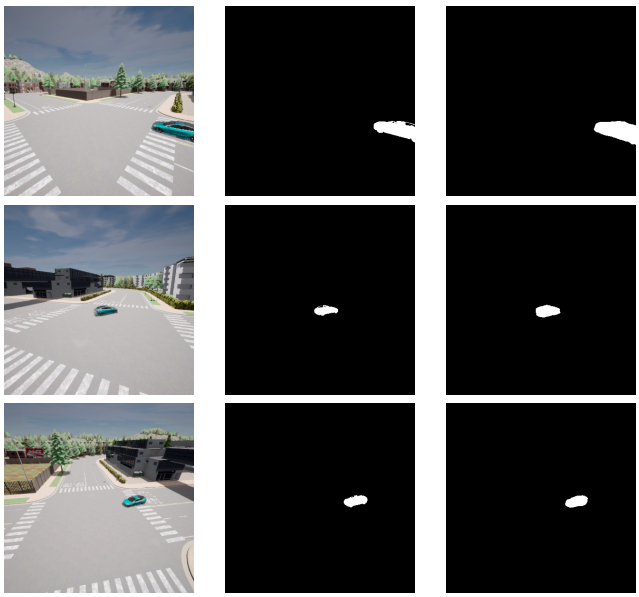

Figure 4. **Object masks for one-vehicle dataset.** From left to right: Ground truth image, ground truth object mask and the predicted object mask

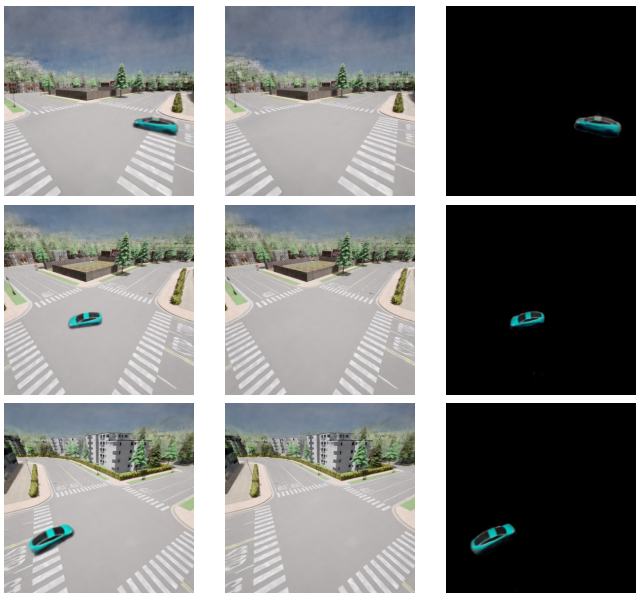

Figure 5. **Decomposition of the scene into the static background and the dynamic vehicle for the one-vehicle dataset.** From left to right: Composed rendering, static background and the vehicle

4.4.2 Pose Optimization Library

We experimented with pose optimization libraries that can work within the PyTorch framework: PyPose [20] and LieTorch [17]. We first started using the LieTorch library, but our experiments failed with pose optimization. Therefore, we switched to another alternative called PyPose, which was better documented than LieTorch and has active support and development. We successfully implemented our approach using PyPose.

4.4.3 Sliding Window Optimization

Our method fails to optimize for long sequences as described in 4.5. Therefore, we experimented with sliding window optimization, where instead of optimizing for all the frames up until the current frame, we optimize only for the frames in a window with a particular length ending at the current frame, and we slide it by one frame further when we reach the m_2 threshold described in 3.2.2. We experimented

Figure 6. **Qualitative comparison of our method to baselines in novel-view synthesis task for two-vehicle dataset across the first five frames.**

	Metric	PSNR ↑	SSIM ↑	LPIPS ↓
Composition	Ours(only entropy reg.)	23.40	0.818	0.333
	Ours(entropy + dynamic reg.)	23.39	0.818	0.332
	Ours(entropy + ray reg.)	23.51	0.824	0.320
	Ours(entropy + static reg.)	23.62	0.828	0.316
	Ours	**23.65**	**0.829**	**0.314**
Static	Metric	PSNR ↑	SSIM ↑	LPIPS ↓
	Ours(only entropy reg.)	23.65	0.821	0.322
	Ours(entropy + dynamic reg.)	23.63	0.821	0.321
	Ours(entropy + ray reg.)	23.69	0.826	0.312
	Ours(entropy + static reg.)	23.71	0.829	0.310
	Ours	**23.75**	**0.830**	**0.308**
Dynamic	Metric	PSNR ↑	SSIM ↑	LPIPS ↓
	Ours(only entropy reg.)	15.98	0.492	0.056
	Ours(entropy + dynamic reg.)	16.07	0.498	0.056
	Ours(entropy + ray reg.)	17.33	0.575	**0.042**
	Ours(entropy + static reg.)	19.29	0.658	0.050
	Ours	**19.31**	**0.661**	0.045

Table 6. **Ablation study for regularization terms**

with windows of length 2,3,4 and 5 but still failed to work on longer sequences.

4.5. Limitations

Our experiments showed that our method cannot generalize for long video sequences. For example, in our one-vehicle dataset, after the 9th frame, our estimated poses start to diverge, and in our two-vehicle dataset, we observe this after the 8th frame. Therefore, we report our results until those frames and leave the investigation of improving pose estimation as a future work.

5. Conclusion

We reproduced the STaR method [25] and extended it for multiple objects in this work. Then, we successfully demonstrated its performance on our synthetic dataset. As a further work, this method can be tested on real-world datasets with many more moving objects than our dataset. Furthermore, the issue with pose estimation for longer se-

Figure 7. **Qualitative comparison of our method to baselines in novel-view synthesis task for two-vehicle dataset for the fifth frame seen from different views.**

quences should be investigated for the successful application of this method to real-world scenarios. Testing this method on datasets with ego-vehicle cameras would also be interesting, demonstrating its application to the autonomous driving problem.

References

[1] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator, 2017. 1, 4

[2] Mijeong Kim, Seonguk Seo, and Bohyung Han. Infonerf: Ray entropy minimization for few-shot neural volume rendering, 2022. 3

[3] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 4

[4] Abhijit Kundu, Kyle Genova, Xiaoqi Yin, Alireza Fathi, Caroline Pantofaru, Leonidas Guibas, Andrea Tagliasacchi, Frank Dellaert, and Thomas Funkhouser. Panoptic neural fields: A semantic object-aware neural scene representation, 2022. 2

[5] Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely. Dynibar: Neural dynamic image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4273–4284, 2023. 1

[6] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention, 2020. 2

[7] 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, 2020. 1, 2, 3

[8] Michael Niemeyer and Andreas Geiger. GIRAFFE: Representing scenes as compositional generative neural feature fields. *https://arxiv.org/abs/2011.12100*, 2020. 1

[9] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision, 2020. 4

[10] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes.

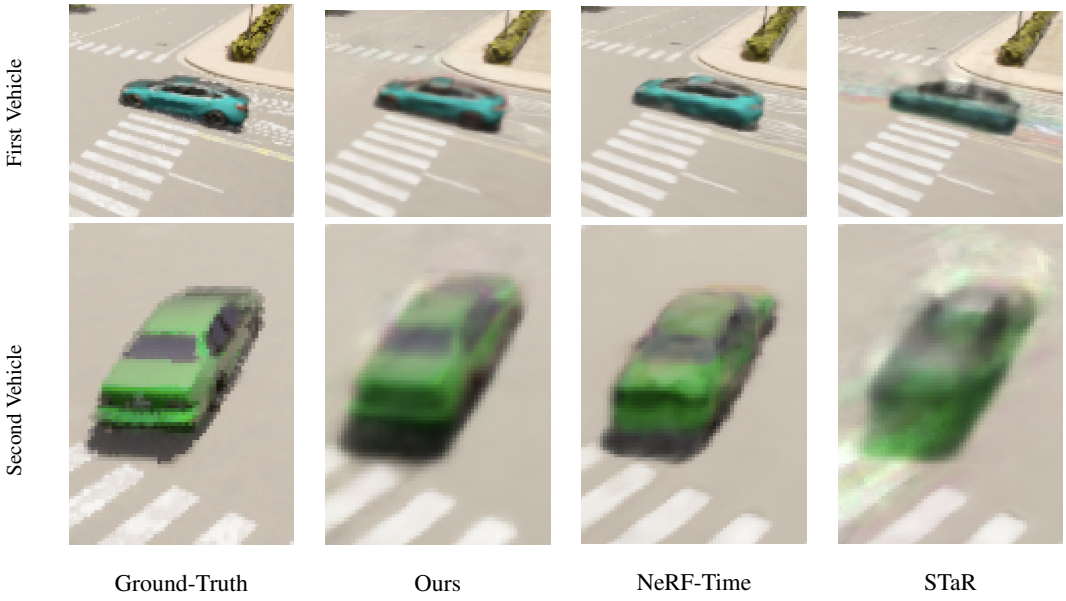

Figure 8. **Closer look to the renderings of our method and baselines in novel-view synthesis task for the two-vehicle dataset.**

https://arxiv.org/abs/2011.10379, 2020. 1, 2

[11] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan Goldman, Steven Seitz, and Ricardo Martin-Brualla. Deformable neural radiance fields. *https://arxiv.org/abs/2011.12948*, 2020. 1

[12] 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, 2021. 1

[13] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural radiance fields for dynamic scenes. *https://arxiv.org/abs/2011.13961*, 2020. 1

[14] Karl Stelzner, Kristian Kersting, and Adam R. Kosiorek. Decomposing 3d scenes into objects via unsupervised volume segmentation, 2021. 2

[15] Karl Stelzner, Kristian Kersting, and Adam R. Kosiorek. Decomposing 3d scenes into objects via unsupervised volume segmentation, 2021. 2

[16] Jrgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. pages 573–580, 10 2012. 5

[17] Zachary Teed and Jia Deng. Tangent space backpropagation for 3d transformation groups, 2021. 6

[18] Haithem Turki, Jason Y. Zhang, Francesco Ferroni, and Deva Ramanan. Suds: Scalable urban dynamic scenes, 2023. 1, 2

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. 2

[20] Chen Wang, Dasong Gao, Kuan Xu, Junyi Geng, Yaoyu Hu, Yuheng Qiu, Bowen Li, Fan Yang, Brady Moon, Abhinav Pandey, Aryan, Jiahe Xu, Tianhao Wu, Haonan He, Daning Huang, Zhongqiang Ren, Shibo Zhao, Taimeng Fu, Pranay Reddy, Xiao Lin, Wenshan Wang, Jingnan Shi, Rajat Talak, Kun Cao, Yi Du, Han Wang, Huai Yu, Shanzhao Wang, Siyu Chen, Ananth Kashyap, Rohan Bandaru, Karthik Dantu, Jiajun Wu, Lihua Xie, Luca Carlone, Marco Hutter, and Sebastian Scherer. Pypose: A library for robot learning with physics-based optimization, 2023. 4, 6

[21] Tianhao Wu, Fangcheng Zhong, Andrea Tagliasacchi, Forrester Cole, and Cengiz Oztireli. D^2nerf: Self-supervised decoupling of dynamic and static objects from a monocular video, 2022. 1, 3

[22] Jiawei Yang, Boris Ivanovic, Or Litany, Xinshuo Weng, Seung Wook Kim, Boyi Li, Tong Che, Danfei Xu, Sanja Fidler, Marco Pavone, and Yue Wang. Emernerf: Emergent spatial-temporal scene decomposition via self-supervision, 2023. 3

[23] Lin Yen-Chen. Nerf-pytorch. https://github.com/yenchenlin/nerf-pytorch/, 2020. 4

[24] Hong-Xing Yu, Leonidas J. Guibas, and Jiajun Wu. Unsupervised discovery of object radiance fields, 2022. 1, 2

[25] Wentao Yuan, Zhaoyang Lv, Tanner Schmidt, and Steven Lovegrove. Star: Self-supervised tracking and reconstruction of rigid objects in motion with neural rendering, 2020. 1, 2, 3, 4, 5, 6, 7, 8