A Implementation Details

Our base model F_θ and the expert adaptor E_θ are constructed as 8-layer and 4-layer Multilayer Perceptrons (MLPs) with 256 channels, respectively, featuring ReLU activations at each layer. For every image, 1024 rays are sampled, with the network generating 64 coarse and 128 fine points per ray. During the conflict-aware self-distillation process, an additional 1024 rays are sampled to derive supervision signals from the teacher model. Our implementation utilizes the PyTorch framework [2], leveraging the Adam optimizer [1] with an initial learning rate set to 5e-4, decaying exponentially down to 5e-5. The initial network F_θ is trained for 200k iterations on the original scenes from each dataset. Subsequently, the expert adaptor E_θ undergoes fine-tuning for 10k iterations on each newly introduced scene, utilizing an initial learning rate of 5e-4. All experimental procedures are carried out on a single NVIDIA GeForce RTX 3090 GPU.

B More Experiments

B.1 Synthetic Dataset Kitchen

Dataset We conduct a comprehensive evaluation of our algorithm using the *Dataset Kitchen*, simulating a room-level kitchen environment as depicted in Figure B.1. For the initial training phase, 200-300 multi-view images are captured from six strategically planned camera trajectories to ensure complete scene coverage and robust training. The dataset is divided into an 80% training subset and a 20% validation subset for performance evaluation. Modifications to the original scene are managed by capturing approximately 30 multi-view images per change, centered around the area of alteration. Among them, 10 images are uniformly sampled for training, while the remaining serve for evaluation purposes. The resolution of all captured images is 960×640.

Results We compare two baseline algorithms and the quantitative results are presented in Table 1, Table 2 and Table 3. From the tables, it can be concluded that direct fine-tuning causes drastic catastrophic forgetting, leading to a significant degradation in the rendering quality of the *old* scene. Although the naive memory replay mechanism mitigates this (e.g., with the lowest BTM value), it underperforms in the *new* scene and overall scene (e.g., FM value is not satisfied). Contrastingly, our proposed approach exhibits outstanding performance in the *new* scene, concurrently preserving the *old* scenes. The qualitative results, as depicted in Figure B.1, further demonstrate the ability of our method.

Table 1: Rendering results of atomic/composite/sequential operation in *Dataset Kitchen*. It showcases the PSNR on both new and old scenes following the corresponding operations. The best results are highlighted in **bold**.

Operations	ADD		DELETE		MOVE		REPLACE		SEQ	
	Old	*New*	*Old*	*New*	*Old*	*New*	*Old*	*New*	*Old*	*New*
FT	25.48	26.83	23.71	31.06	23.33	23.02	25.13	28.97	23.06	28.58
MR	29.00	25.94	28.11	26.88	28.59	19.97	28.95	29.17	27.63	29.41
Ours	**29.32**	**27.43**	**29.05**	**31.09**	**28.61**	**23.09**	**29.33**	**30.16**	**28.06**	**30.07**

Table 2: Rendering result of the sequential operation in *Dataset Kitchen*. Tasks 1-4 are defined to evaluate the model based on camera poses corresponding to four operations: add, delete, move, and replace, respectively. Models are evaluated after being tuned for previously completed tasks. The best results are highlighted in **bold**.

Testing on		FT				MR				Ours			
		Task1	*Task2*	*Task3*	*Task4*	*Task1*	*Task2*	*Task3*	*Task4*	*Task1*	*Task2*	*Task3*	*Task4*
Training on	*Task1*	26.83	-	-	-	25.94	-	-	-	**27.43**	-	-	-
	Task2	23.18	31.06	-	-	23.69	26.88	-	-	**26.77**	**31.09**	-	-
	Task3	19.54	25.32	23.02	-	23.71	26.98	19.97	-	**26.25**	**28.13**	**23.09**	-
	Task4	18.36	24.33	18.83	28.97	23.60	26.92	18.83	29.17	**25.97**	**27.92**	**22.29**	**30.16**

Table 3: Forgetting metrics in sequential operations.

Metrics	FM ↑	BTM ↓
FT	22.623	6.463
MR	24.630	**1.147**
Ours	**26.585**	1.810

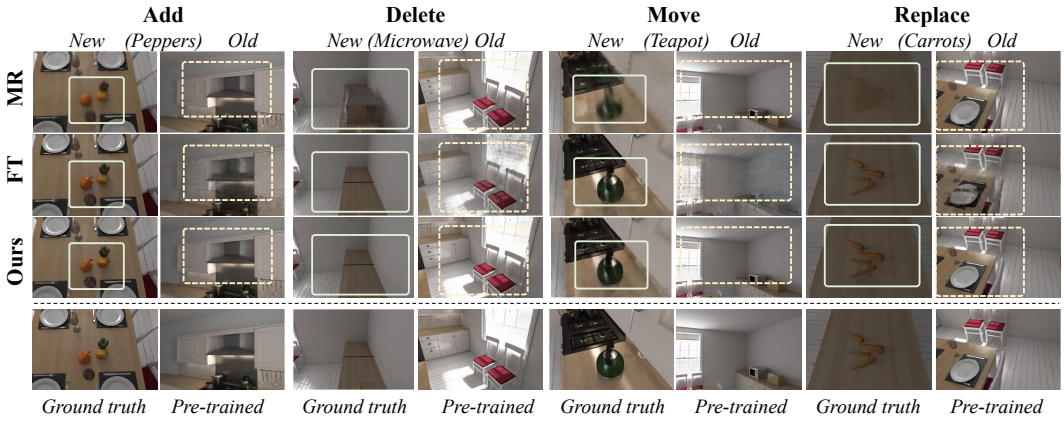

Figure 1: Visualization results after single-step operations in *Dataset Kitchen*. We evaluate 4 single-step operations: add, delete, move, and replace. The ground truth images and inference images from the pre-trained model are used to highlight the rendering quality on new tasks and quality degradation caused by forgetting severity.

B.2 City-level Dataset Rome

In many cases in the real world, we may only be able to capture a small number of images focused on the localized area that has been modified. This limitation often results from various factors such as time constraints, limited access to equipment and resources. For instance, acquiring data at the city level, whether through aerial or street-level photography, is a demanding and costly task. As a result, the costs of recapturing data of the whole scene after each scene change are substantial.

To illustrate the effectiveness of our model, we also construct a synthetic dataset *Dataset Rome* at the city level. We collect 200-300 aerial images around the Colosseum in Rome to reconstruct the initial scene, followed by two ATOMIC operations - ADD and DELETE. The ADD operation adds a black sports car at the center of the Colosseum, while the DELETE operation removes a tower outside. The original dataset is partitioned into training and validation sets, with the former accounting for 80% of the data and the latter 20%, to evaluate the algorithm's efficacy. After scene alterations, approximately 10 additional images are captured to fine-tune the models, all of which are resized to 640×360. Visualization results are shown in Figure 3. Our study has demonstrated that in scenarios where the cost of recapturing data is prohibitively high, our model can proficiently reconstruct new localized regions with limited amounts of new data and minimal time investment, while significantly preventing catastrophic forgetting phenomenon.

B.3 Real-scene Dataset Laboratory

To demonstrate the adaptability of our methods in diverse datasets, we also conduct data collection in a room-level dataset named *Dataset Laboratory* to establish the practical application of our algorithm in real-scene settings. We employ an iPhone 13 Pro to capture approximately 120 frames of a laboratory scene. The resulting images are resized to 640×360 and then divided into training/validation sets with a ratio of 50%/50% to assess our algorithm's effectiveness. Following this, we conduct ADD and DELETE operations on objects within the scene and captured around 10 additional images after each manipulation to emphasize the modified regions. We compute intrinsic and extrinsic camera parameters for each frame using an off-the-shelf open-source software COLMAP [3]. The qualitative

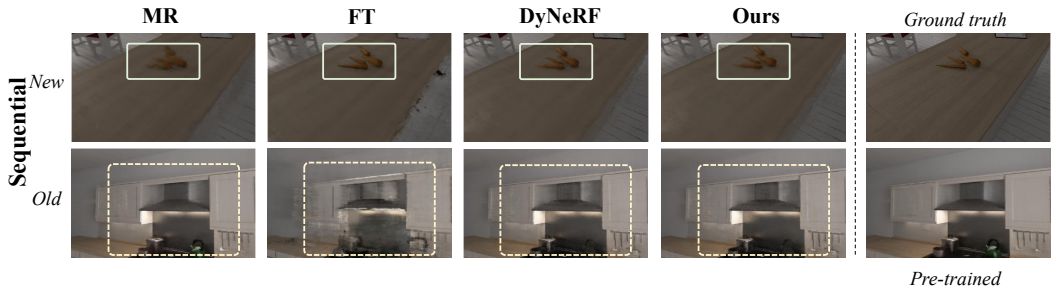

Figure 2: Visualization results after sequential operations in *Dataset Kitchen*. The images in the upper row are rendered after the model has been fine-tuned on four operations, while the images in the lower row test and demonstrate the forgetting issue on the old task..

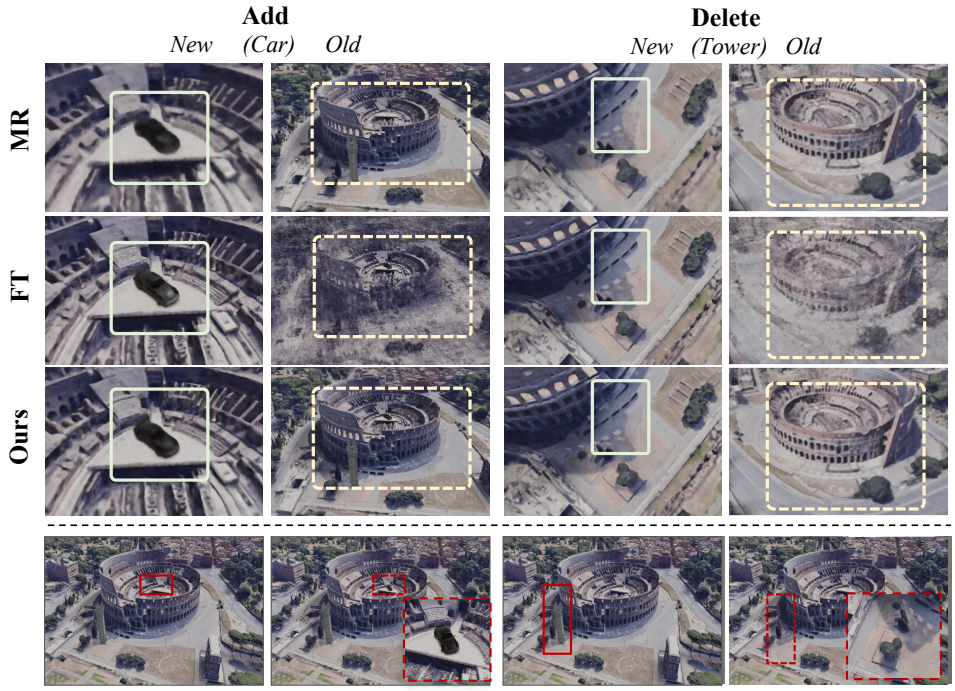

Figure 3: Visualization results after two ATOMIC operations in *Dataset Rome*.

results depicted in Figure 4 affirm that our model successfully adapts to scene alterations, such as introducing a yellow teapot or eliminating a doll, with minimal novel data while maintaining performance in unmodified regions.

B.4 More analysis about DyNeRF

It should be noted that in the main paper, training DyNeRF incorporates old images from previous stages while our method only trains on new images. To ensure a fair comparison, we also conduct an experiment to train DyNeRF solely with new images in an additional experiment. From Table 4, not using old images in DyNeRF's training results in a significant decline in the previous task and problems of forgetting.

Figure 4: Visualization results in *Dataset Laboratory*.

Table 4: Comparison with DyNeRF without old data in *Dataset Whiteroom* and *Dataset Kitchen*. It showcases the PSNR on both new and old scenes following the corresponding operations. The best results are highlighted in **bold**.

Datasets	Methods	ADD		DELETE		MOVE		REPLACE		SEQ	
		Old	*New*	*Old*	*New*	*Old*	*New*	*Old*	*New*	*Old*	*New*
Whiteroom	DyNeRF w/o old	27.40	23.54	27.65	34.68	21.72	**30.21**	26.31	29.76	22.21	29.06
	Ours	**32.33**	**25.29**	**32.43**	**34.77**	**33.30**	29.97	**33.33**	**29.82**	**33.38**	**29.74**
Kitchen	DyNeRF w/o old	27.33	27.19	27.62	**31.28**	25.48	22.13	27.69	30.10	26.50	30.00
	Ours	**29.32**	**27.43**	**29.05**	31.09	**28.61**	**23.09**	**29.33**	**30.16**	**28.06**	**30.07**

C Limitations

We have incorporated an additional expert network into our system. However, the dynamic nature of the world without constraints may give rise to scalability issues such as storage problems for the expert network over time. Despite this, we have trained an expert network for each scenario, which allows us to facilitate the model's rollback at any given moment, thereby enhancing its practicality. Additionally, the experts are equipped with the ability to adapt to changes from the original scenarios. As a result, during model inference, each scenario is still matched with only one expert network, effectively eliminating scalability concerns during inference.

D Boarder Impact

This research work exhibits potential for widespread impact across numerous domains, leveraging the inherent temporal dynamism in real-world scenes. By pioneering an efficient method to adapt NeRFs to changes in the environment with minimal new data and retaining the memory of unaltered areas, we increase the accessibility and applicability of NeRFs in various scenarios. CL-NeRF's ability to offer high-quality novel views of both changed and unchanged regions could prove instrumental in fields as diverse as augmented reality, robotics, surveillance, and architecture, where understanding and adapting to scene changes is paramount. Moreover, by introducing a new benchmark and making our code available, we invite and facilitate further research and collaboration, driving innovation in this field. We anticipate that our work, by reducing the retraining burden, will pave the way for more applications of NeRFs, thus enriching the interaction between artificial intelligence and the physical world.

E More Visualization Results

For additional visualization results, please refer to the supplementary video (*supplementary video.mp4*) that we provide.

References

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

[2] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[3] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016.