# NeRF with Prior Knowledge

Anonymous CVPR submission

Paper ID *****

## 1. Introduction

Although neural radiance fields (NeRF) [3] have achieved great success in novel view synthesis with surprisingly simple network design, generalization remains a problem for image synthesizing in novel views. To mitigate such problems, researchers have proposed a number of approaches, such as CodeNeRF [1], PixelNeRF [4], and ViT+NeRF [2]. All these approaches utilize encoding of extra information, i.e., prior knowledge, to improve the quality of synthesized images in novel views. The newly published work, ViT+NeRF [2], claimed that previous works suffer from certain kinds of limitations in their generalization abilities, which can be overcome by the hybrid architecture of Vision Transformer+NeRF. Although the newly proposed architecture is theoretically appealing as it utilizes Transformer's ability to model long range dependencies, and it provides both local image features and global features as prior knowledge, the work is relatively new and requires further evaluations. Moreover, many of the papers about NeRF only validate their choices of design by evaluating their performance on SRN dataset, which contains simple scenarios of one category objects in low resolution, it is still not clear if a varity of NeRF models can still attain their performance in more photo-realistic, complex scenarios with higher resolution images. In this project, we propose to compare ViT+NeRF [2] with previous works to see if the newly-proposed ViT+NeRF architecture can effectively boost the model's generalization abilities. We will also test NeRF models' performance on simple scenarios (SRN-cars) and more complex and photo-realistic scenarios to see how well they can adapt and generalize to new scenarios.

**Problem Statement**

The overall goal of this project is to test and compare different variants of NeRFs on different kinds of datasets to see if they can generalize well 1) to novel views or 2) to more complex scenarios.

More specifically, the goals of this project are:

1) Test ViT+NeRF [2] on datasets to collect evidence if ViT+NeRF has better generalization ability or not;

2) Test and compare CodeNeRF [1] and PixelNeRF [4] on simple (SRN) and complex (DTU) datasets and analysis their generalization abilities.

**Input** The inputs are images from SRN-cars dataset and DTU dataset with pose information.

**Output** The outputs are synthesized images in novel views, or videos.

**Assumption** For CodeNeRF, we assume that the model is trained with single view images with ground truth pose information, and during test no ground truth pose information is provided.

For PixelNeRF, we assume that the model can be trained with a single or few posed images.

For ViT+NeRF, we assume that the model can render novel views from one single input image without pose information, and one model can generalize well across multiple object categories.

**Motivation**

The motivations of this project are:

First, test the newly proposed work, ViT+NeRF [2], to see if Vision Transformer based architecture can obtain better generalization ability, i.e., 1) if ViT+NeRF can synthesize novel view images with one single un-posed image, 2) to what degree ViT+NeRF can generalize well across the boundary of different categories.

Second, test and compare different approaches, such as CodeNeRF [1] and PixelNeRF [4], to see 1) if their proposed design leads to better quality of synthesized images, and 2) if the quality improvements can be attained and generalized to more complex scenarios.

Third, have a better understanding what design leads to better performance and what design can generalize well to novel views and unseen realistic datasets.

## 2. Related Work

### 2.1. CodeNeRF

Given that the original NeRF is scene-specific, CodeNeRF proposed to implicitly learn disentangled representations of shape and texture embeddings of the objects to boost the model's generalization abilities. In order to do this, CodeNeRF utilizes an auto-decoder architecture to

learn the implicit representations of the objects, which is a 256 dimensional vector $(z_s, z_t) \in \mathbb{R}^{256}$ with disentangled shape components $z_s$ and texture componets $z_t$. Then, the embeddings are fed into a multi-layer perceptron $F_\Phi$ to generate the corresponding volume density $\sigma$ and RGB color $c$:

$$F_\Phi : (\gamma_x(x), \gamma_d(d), z_s, z_t) \rightarrow (\sigma, c)$$

Compared with the original NeRF where only the positional encoding of 3D location $\gamma(x)$ and the positional encoding of 2D viewing direction $\gamma(d)$ are fed into the MLP $F_\Phi$, CodeNeRF provides extra information $z_s$ and $z_t$. Taking advantage of the fact that volume density $\sigma$ depends only on 3D location $x$ and shape embedding $z_s$, and the RGB color additionally depends on viewing direction $d$ and texture embedding $z_t$, CodeNeRF designed a two-layer structured MLP where the first layer $F^s_{\Phi_s}$ predicts volume density $\sigma$ and an intermediate variable $v$ from shape embedding $z_s$ and the second layer $F^t_{\Phi_t}$ predicts color $c$ from the texture embedding $z_t$:

$$\begin{gathered} F^s_{\Phi_s} : (\gamma_x(x), z_s) \rightarrow (\sigma, v) \\ F^t_{\Phi_t} : (v, \gamma_d(d), z_t) \rightarrow (c) \\ F_\Phi : F^s_{\Phi_s} \circ F^t_{\Phi_t} \end{gathered}$$

With additional embedding information and two-layer MLP structure, CodeNeRF was able to decompose the learning process in a somewhat explainable way and deliever better generalization results. The better generalization ability is shown as 1) CodeNeRF can generate novel view images of the objects with different shapes and textures; 2) CodeNeRF can better maintain geometric and appearance consistency in occluded regions; and 3) CodeNeRF can reach on par performance with the method which requires pose information at test time. Meanwhile, the limitation is that the original paper of CodeNeRF only considers generalizing to unseen objects within a category.

## 2.2. PixelNeRF

Original NeRF needs to optimize the representation to each scene independently, which requires many calibrated views. To address such problems, PixelNeRF proposed to condition NeRF on image inputs in a fully convolutional manner. PixelNeRF leverages convolutional neural networks to extract feature volume $\mathbf{W} = \mathbf{E}(I)$ from the input image $\mathbf{I}$. For each query point $x$ and a camera ray with view direction $d$, a corresponding image feature $\mathbf{W}(\pi(x))$ is extracted from $\mathbf{W}$ by projection and interpolation. Then, the feature $\mathbf{W}(\pi(x))$ is fed into the NeRF network together with $x$ and $d$ to obtain:

$$f : (\gamma(x), d; \mathbf{W}(\pi(x)) = (\sigma, c)$$

When the query view direction is similar to the input view direction, PixelNeRF can rely more on the input for prediction; when the query view direction is not similar to the input view direction, the learned prior, $\mathbf{W}(\pi(x))$.

To support a better understanding of the scene, PixelNeRF also allows multiple view images as input. To support this, PixelNeRF appends a final layer $f_2$ behind the initial NeRF network $f_1$. In $f_1$, features, view directions, and locations from each image are passed into the network to compute the intermediate vectors $V^{(i)}$:

$$\mathbf{V^{(i)}} = f_1(\gamma(x^{(i)}), d^{(i)}; \mathbf{W}^{(i)}(\pi(x^{(i)})))$$

Then, the intermediate vectors are aggregated with an average pooling operator $\psi$ and fed into the final layer $f_2$ to obtain the prediction of volume density and color:

$$(\sigma, c) = f_2(\psi(V^{(1)}, \cdots V^{(n)}))$$

The advantage of PixelNeRF is that it utilizes image features to provide extra guidance for rendering when the query is not similar from input. Multi-view input further equipped PixelNeRF with a scene prior, which waives the requirement of NeRF's test time optimization and enhances PixelNeRF's ability to generate images in novel views for more complex scenes.

Researcher's opinions about PixelNeRF's generalization ability are somewhat divided. If generalization is defined as the ability to generate images of good quality in novel views, then PixelNeRF indeed can generalize well to novel views. Since it directly uses image features as priors, in fact it can generate sharper images when the target view is close to the input view.

However, the image feature priors can also leads to overfitting in some cases. This issue is especially prominent when PixelNeRF is required to generate the occluded part in the input image. An interesting example is self-occlusion, when PixelNeRF collected the image features from one side of the object, but is required to predict rendering outcome for the opposite side of the object. Since image feature priors are local features, it lacks the ability to model long range dependencies and cannot perform well for prediction of occluded parts.

## 2.3. ViT+NeRF

ViT+NeRF can be considered as a combination and extension of 1D approaches such as CodeNeRF and 2D approaches such as PixelNeRF. Similar to CodeNeRF, latent features are generated for the images to capture global features for better generalization. Similar to PixelNeRF, 2D convolutional features are also used to capture the local details.

ViT+NeRF proposes to leverage Transformer to provide global latent features, which can potentially mitigate PixelNeRF's issue of overfitting to local features. On the other hand, ViT+NeRF also uses CNN to extract image features,

so that it can utilize local features to provide better details. The global latent features are then passed down to a convolutional decoder to obtain the global features $W_G$ and merge with the local features $W_L$. Then, the merged features $\mathbf{W}$ are passed to the NeRF network to predict the rendering outcome:

$$(\sigma, c) = f(\gamma(x), d; \mathbf{W}(\pi(x))$$

Theoretically, ViT+NeRF might have better generalization ability because: 1) Transformer has the ability to model long range dependencies, which might provide global features that are not confined to shape and texture features as in CodeNeRF. 2) ViT+NeRF also leverage image feature priors to capture local details. In the paper, it is reported that ViT+NeRF can 1) render novel views from a single input image, 2) generalize to multiple object categories (within SRN dataset), 3) generate images with less bluriness, and 4) synthesize better images for the occluded regions.

### 2.4. Our Work

Each of these works highlight its merit in the paper: CodeNeRF highlights that it has better performance when pose and latent features are initialized far from the ground truth, and it also mentions that CodeNeRF can learn the latent features which helps maintain geometric and appearance consistency in occluded regions [1]. PixelNeRF provides slightly better evaluation results in numbers when compared with CodeNeRF. PixelNeRF allows training from multiple inputs to gather more scene information, and it allows to synthesize images in real world scenarios [4]. ViT+NeRF theoretically combines the strength of the above two, and it reports better quantitative results and qualitative results with fewer blurriness [2]. However, we still need a thrid party to compare and evaluate these methods to obtain a more unbiased opinion, which is the purpose of this project.

## 3. Approach

What we did in this project includes:

Qualitative and quantitative evaluation of CodeNeRF on SRN-Cars and DTU datasets;

Qualitative and quantitative evaluation of PixelNeRF on SRN-Cars and DTU datasets;

Qualitative evaluation of ViT+NeRF on SRN-Cars dataset.

The motivation is to test and compare different variants of NeRF's generalization ability and their performance on different kinds of datasets.

**Acknowledge of Code Usage**

For CodeNeRF, we used the authors; implementation at: https://github.com/wbjang/code-nerf.

For PixelNeRF, we used the authors' implementation at: https://github.com/sxyu/pixel-nerf.

For ViT+NeRF, we used the authors' implementation at: https://github.com/ken2576/vision-nerf.

**Implementation**

1) **The Dataloader of DTU dataset for CodeNeRF** CodeNeRF's original code only provides the dataloader for SRN datasets, to test CodeNeRF's performance on DTU dataset, I implemented the dataloader for DTU dataset. DTU dataset is organized differently from SRN dataset. While SRN dataset provides intrinsic matrix in intrinsic.txt for the scene and the poses information in pose/ $image_name.txt for each viewpoint in the scene, DTU dataset stores all viewpoints' scene geometry information in one file called cameras.npy. One of the main tasks of the dataloader is to decompose the projection matrix $P$ to obtain instrinsic matrix $K$, rotation $R$, and translation $t$. Then, we can recover pose information for each view from $R$ and $t$. After that, we prepare the image, pose and intrinsic matrix in the right format for the trainer.

2) **Mirgration to Colab** Since I encountered major computation bottleneck in training models on my own computer, I migrate the codes to Colab for online GPU resources. I adapted the codes so that they are runnable in Colab settings.

**Relation with Rendering and Image Generation** CodeNeRF, PixelNeRF, and ViT+NeRF are all models for rendering. They synthesize images in new views, which connects themselves to the domain of image generation.

## 4. Results

### 4.1. ViT+NeRF on Single Category SRN Dataset

To evaluate ViT+NeRF, we test it on category-specific view synthesis task on SRN Dataset. SRN dataset consists of 3514 cars and 6591 chairs that are split into training, validation, and test sets. There are 251 views for each object. During testing, the 64-th view is chosen as the input view, and all other views are target views. The input/output images' resolution is 128×128.

The original paper mentioned that their model converges after 500K iterations of training on 16 NVIDIA A100 GPUs [2]. We don't have enough computational resources to support such training, so, to better replicate the results presented in the paper, here we use the pre-trained models released in the authors website. The pre-trained models are trained on SRN-Cars and SRN-Chairs for 500K iterations.

**Qualitative Results on SRN-Cars Dataset**

Figure 1 shows images synthesized by ViT+NeRF for SRN-Cars datasets. For each object, the input image is from view No. 64, and we select target views No.0, 35, 50, 60, 70, 120, 140, 155, 200 for demonstration of the rendering outcomes. In comparison, Figure 2 shows the ground truth images in the corresponding view directions.

**Qualitative Results on SRN-Chairs Dataset**


Figure 1. **ViT-NeRF's Category-specific view synthesis on Cars.** View No.64 is used as input. Here we provide the rendering outcome for views No.0, No.35, No.50, No.60, No.70, No.120, No.140, No.155, No.200.


Figure 2. Ground truth images from views No.64, No.0, No.35, No.50, No.60, No.70, No.120, No.140, No.155, No.200.


Figure 3. **ViT-NeRF's Category-specific view synthesis on Chairs.** View No.64 is used as input. Here we provide the rendering outcome for views No.0, No.35, No.50, No.60, No.70, No.120, No.140, No.155, No.200.


Figure 4. Ground truth images from views No.64, No.0, No.35, No.50, No.60, No.70, No.120, No.140, No.155, No.200.

Figure 3 shows images synthesized by ViT+NeRF for SRN-Chair datasets. For each object, the input image is from view No. 64, and we select target views No.0, 35, 50, 60, 70, 120, 140, 155, 200 for demonstration of the rendering outcomes. In comparison, Figure 4 shows the ground truth images in the corresponding view directions.

### 4.2. PixelNeRF on Single Category SRN Dataset

To evaluate PixelNeRF, we test it on category-specific view synthesis task on SRN-Cars Dataset. During testing, the 64-th view is chosen as the input view, and all other views are target views. The input/output images' resolution is 128×128. To ensure replication of the results, we use the pre-trained model released by the author.

**Qualitative Results on SRN-Cars Dataset**

Figure 5 shows images synthesized by PixelNeRF for SRN-Chair datasets. For each object, the input image is from view No. 64, and we select target views No.0, 35, 50, 60, 70, 120, 140, 155, 200 for demonstration of the rendering outcomes. In comparison, Figure 2 shows the ground truth images in the corresponding view directions.


Figure 5. **PixelNeRF's Category-specific view synthesis on Cars** views No.64 is used as input. Here we provide the rendering outcome for views No.0, No.35, No.50, No.60, No.70, No.120, No.140, No.155, No.200.

| | PSNR | SSIM |
|---|---|---|
| **SNR-Cars** | 22.864 | 0.895 |

Table 1. Quantitative Evaluation of PixelNeRF on SRN-Cars Dataset

**Quantitative Results on SRN-Cars Dataset**

Here we use PSNR and SSIM as evaluation metrics. We compute the average PSNR and SSIM for 5 tested samples. Evaluation outcome can be found at Table 1.

### 4.3. PixelNeRF on DTU Dataset

PixelNeRF allows multi-view images as input and can synthesize images in real-world scenarios. Here we have qualitative and quantitative evaluations for PixelNeRF's performance on DTU Dataset with 1-view/3-view image(s) at test time. The model we used for evaluation is trained on DTU dataset with 3-view image inputs.

Both Figure 6 and Figure 7 show the rendering outcomes for view No.12, 24, and 36. Figure 6 shows the outcome when there are 3 views at test time, while Figure 7 shows the outcome when there is 1 view at test time.

### 4.4. CodeNeRF on Single Category SRN Dataset

**Qualitative Results on SRN-Cars Dataset**

To evaluate CodeNeRF, we test it on category-specific view synthesis task on SRN-Cars Dataset. During testing, the 1-st view is chosen as the input view, and all other views are target views. The input/output images' resolution is 128×128. Here we use a self-trained model which was trained on SRN-Chair dataset for 50K iterations.

Figure 8 shows CodeNeRF's rendering outcomes. In each image pair, the left one is rendered by CodeNeRF, and the right one is ground truth.

**Quantitative Results on SRN-Cars Dataset**

We tested CodeNeRF on 10 sample objects from SRN-Cars dataset. For each object, we computed PSNR and SSIM. Then we computed the average PSNR and average SSIM for the 10 objects from SRN-Cars dataset. The results are summarized in Table 2

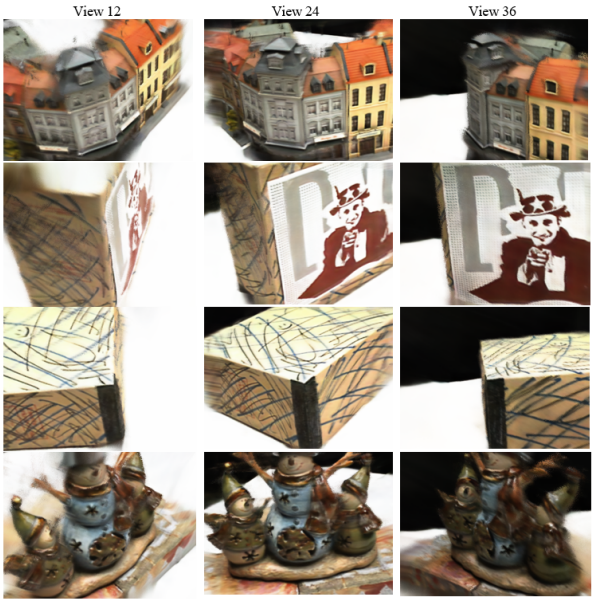


Figure 6. **PixelNeRF's view synthesis on real-world scenes** The model is trained with 3-view inputs. We also provide 3 views at test time. Here are the rendering outcome for views No.12, No.24, No.36.

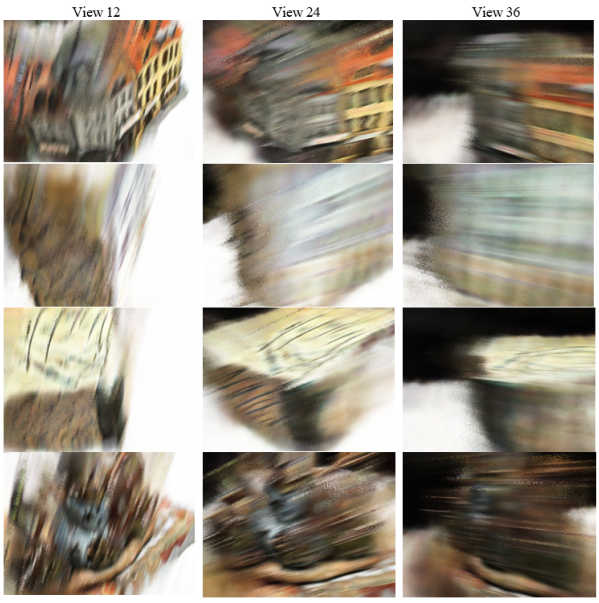


Figure 7. **PixelNeRF's view synthesis on real-world scenes** The model is trained with 3-view inputs. We only provide 1 view at test time. Here are the rendering outcome for views No.12, No.24, No.36.

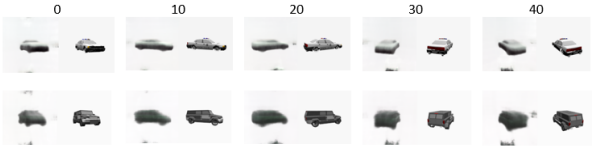


Figure 8. **CodeNeRF's Category-specific view synthesis on Cars** View No.1 is the input. Here we show the rendering outcomes for view No.0, 10, 20, 30, 40. In each pair of images, the left one is rendered by CodeNeRF, and the right one is the ground truth.

| | Sample 1 | Sample 2 | Sample 3 | Sample 4 | Sample 5 |
|---|---|---|---|---|---|
| **PSNR** | 19.092 | 18.792 | 19.50 | 20.640 | 12.594 |
| **SSIM** | 0.869 | 0.875 | 0.834 | 0.874 | 0.674 |
| | **Sample 6** | **Sample 7** | **Sample 8** | **Sample 9** | **Sample 10** |
| **PSNR** | 19.830 | 18.273 | 16.449 | 19.831 | 15.583 |
| **SSIM** | 0.847 | 0.839 | 0.777 | 0.892 | 0.741 |
| **Average PSNR** | 18.058 | | | | |
| **Average SSIM** | 0.822 | | | | |

Table 2. Quantitative Evaluation of CodeNeRF on SRN-Cars Dataset

# 5. Analysis

## 5.1. ViT+NeRF on Single Category SRN Dataset

Surprisingly, there is a significant performance gap between what we obtain and what was published in the paper. Of course this performance gap must be interpreted with caution. On our part, we did use the pretrained models released by the authors and followed author's instructions, but there still might be potential mismatches between authors experiment configurations and ours. Another possible reason for the performance gap is that transformer is data-hungry and needs more training to get full performance, and 500K iterations of training is just not enough. However, the authors did mention in the paper that their training converges at 500K iterations, so this scenario is not likely.

Although the specific reason why there is such a big performance gap remains unknown, based on our observations, we should say that ViT+NeRF's performance on the two SRN datasets is far from perfect. There are many evidence that ViT+NeRF doesn't generalize well according to our experiment results: 1) for SRN-Cars dataset, the rendered objects are not in the center. Indeed, the two sample results show similar patterns across different views, which means that the model might have learned and converged to certain scene geometry. However, the model obviously doesn't learn the correct scene geometry, which is not consistent with the paper's claim that ViT+NeRF can generate images in novel views with a single unposed image; 2) the views near the input views (in our example, that is view No. 60 and view No. 70) tend to have better image quality, while image quality significantly degrades for the views that are far from the input views. This means that the model

still can't generalize well to views that are far from input views; 3) the rendered objects lack details, have large areas of "white fog", have missing parts, and suffers from distortions. This means that ViT+NeRF might not be able to live up to the theoretical expectation to combine the strengths of CodeNeRF and PixelNeRF to get better generalization abilities.

### 5.2. PixelNeRF on Single Category SRN Dataset

**Comparison with the Original Paper**

We used the pre-trained model released by the authors, however, our evaluation results are not as good as what was reported in the paper (Our result: PSNR: 22.864, SSIM: 0.895; Paper's results: PSNR: 27.58, SSIM: 0.942). Since we only extract a very small sample from the dataset for evaluation, here we can't reach anything conclusive.

**Comparison with ViT+NeRF**

According to ViT+NeRF's paper, ViT+NeRF can outperform PixelNeRF on a number of datasets; Even if sometimes ViT+NeRF's evaluation score is slightly lower than PixelNeRF, ViT+NeRF still enjoys sharper images with few blurriness. From our observation, we can confirm that images synthesized by PixelNeRF suffer from blurriness, which is consistent with a number of critiques about CNN features. However, according to our qualitative results, PixelNeRF outperforms ViT+NeRF by a large margin. In PixelNeRF, at least all synthesized images have correct scene geometry. For PixelNeRF's self-occlusion issues, we do observe that in certain directions when self-occlusion happens, the synthesized images tend to be blurred with fewer details. However, we find little evidence that PixelNeRF will rely on CNN features heavily and make a wrong prediction when self occlusion happens.

### 5.3. PixelNeRF on DTU Dataset

Figure 6 shows that PixelNeRF can achieve remarkable results for view synthesis on real-world scenes. It is training with multi-views that makes this possible. Since the model is trained with 3-views, by providing 3-view inputs at test time, we create a similar task that PixelNeRF can deal with. However, if we create a more challenging task by providing only 1-view at test time, from Figure 7 we can see that PixelNeRF can't generalize well to this situation.

PixelNeRF's design choices raise a question for us: which kinds of "generalization ability" do we need? Should we accumulate more prior information like PixelNeRF, or should we expect a more expressive model to eventually learn everything like ViT+NeRF? We think this will be an open question for researchers in the future.

### 5.4. CodeNeRF on Single Category SRN Dataset

Our qualitative and quantitative results tend to be not as good as what is reported in the paper. This is partly because we only trained the model for 50K iterations. It won't be a fair comparison if we compare a CodeNeRF model trained with 50K iterations with a ViT+NeRF model trained with 500K iterations.

However, we do find out that in images synthesized by CodeNeRF, shapes and colors of the objects tend to have higher variability. It is not clear if this is due to lack of training, or if this is due to the disentangled design of the latent features.

## 6. Limitations and Future Works

### 6.1. Limitations

In this paper, we obtained some results about CodeNeRF, PixelNeRF, and ViT+NeRF's performance on different types of datasets. However, the evaluation and comparison are only preliminary and far from complete. There are several limitations of this work:

1) **Training**. Due to the bottleneck of computational resources, we only trained CodeNeRF for 50K iterations. So, it is hard to carry out a fair comparison.

2) **Performance Gap** We still don't know exactly why there is such a big performance gap between what is reported in the paper and what we are able to replicate. Hence, we still can't draw any conclusion about ViT+NeRF's generalization abilities.

3) **Testing Generalization Ability** We didn't evaluate CodeNeRF and ViT+NeRF on the real-world scene dataset DTU. We also didn't include SRN agnostic dataset to see which approach can better cross the boundary of different categories.

### 6.2. Future Works

1) **Investigating the Performance Gap** We should figure out why in our evaluation ViT+NeRF didn't perform well. Theoretically speaking, ViT+NeRF can potentially combine both CodeNeRF's and PixelNeRF's strengths, and it shouldn't have performed worse than PixelNeRF or CodeNeRF. However, if it is confirmed that under certain circumstances, ViT+NeRF do have worse performance, then this is an interesting problem which might leads to new thoughts for network design.

2) **Thorough Comparison** We need to compare each approach's generalization ability more thoroughly. We should include SRN-agnostic for testing to see which approach can better cross the boundary of categories within SRN dataset. We should test all approaches on DTU to see if they can adapt and transfer to more realistic and complex scenes.

3) **Self Occlusion** We need to define a metric to measure to what extent the synthesized images are adversely affected by self occlusion.

4) **Formal Definition of Generalization** What we need to do is to include a formal definition of generalization

and doing systematic study about that. In fact, there are many aspects of generalization abilities. Doing well with few clues is generalization; getting good performance under radically novel views is generalization; coping with more realistic and complex scenarios is generalization ... We should have a clear definition what generalization is, and which kinds of generalization abilities do we need.

## 7. Conclusion

In this paper, we evaluated and compared CodeNeRF, PixelNeRF, and ViT+NeRF under different types of datasets.

For PixelNeRF, we confirmed that its synthesized images suffer from blurriness, which some researchers believe is because of the convolutional features. However, we didn't find concrete examples that PixelNeRF might generate radically wrong predictions under self-occlusion.

For CodeNeRF, our evaluation shows that its synthesized images have large variabity in colors and shapes. We suspect that this is because that CodeNeRF adopts disentangled latent features of shapes and colors. Since we only trained the model for 50K iterations, this evidence is still not conclusive.

ViT+NeRF aims at combining the strengths of both PixelNeRF and CodeNeRF. However, in our evaluation it didn't perform well. One possibility, is that Transformer still can't fully replace scene geometry constraints, and making sound predictions from one single unposed image, is still too ambitious. However, we will continue to look into the possible reasons before we find something conclusive.

Based on our observations, we find that PixelNeRF has better generalization ability because 1) it can generate images in new views with fairly good quality, 2) it supports training with multi-views, and 3) it generalize well to real-world datasets.

PixelNeRF gets better performance by allowing multi-view training, that is, getting more prior information about the scene. On the other hand, ViT+NeRF proposes to get better generalization ability by adopting a more expressive model, vision transformer, and relying on as few clues as possible (one single unposed image). It remains unclear which approach can bring better generalization ability in a cost-effective way.

## 8. Team Contributions

Sihui Wang is the sole author of this work.

**Timeline**

**Oct.4 - Oct.17** Replication of demo results for NeRF, CodeNeRF, and PixelNeRF;

**Oct.18 - Nov. 1** "Re-implementation" (or, more precisely, repeat and try to modify the building blocks of the code in order to learn) of NeRF and CodeNeRF;

**Nov. 1 - Nov. 14** Learn the code of PixelNeRF and ViT+NeRF, attempt to implement distillation of Vision transformer;

**Nov. 15 - Nov. 19** Test ViT+NeRF's pretrained model on SRN-car dataset;

**Nov.20 – Nov. 26** Test PixelNeRF's pretrained model on SRN-car and DTU dataset;

**Nov. 27 – Dec. 4** Implement DTU dataloader for CodeNeRF and Training CodeNeRF for DTU dataset.

## 9. Code

https://drive.google.com/drive/folders/1oJkk1f5hWw_F-ndAl_hFj6-5HYAAj4gq?usp=sharing

## References

[1] Wonbong Jang and Lourdes Agapito. Codenerf: Disentangled neural radiance fields for object categories. 1, 3

[2] Lin Yen-Chen Wei-Sheng Lai Tsung-Yi Lin Yi-Chang Shih Lin, Kai-En and Ravi Ramamoorthi. Vision transformer for nerf-based view synthesis from a single input image. 1, 3

[3] Pratul P. Srinivasan-Matthew Tancik-Jonathan T. Barron-Ravi Ramamoorthi Mildenhall, Ben and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. 1

[4] Vickie Ye Matthew Tancik Yu, Alex and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. 1, 3