CARNEGIE MELLON UNIVERSITY

School of Architecture
College of Fine Arts

Thesis

Submitted in Partial Fulfillment of the requirements for the degree of

Master of Science in Computational Design

TITLE:

3D Art Research with Radiance Field and Style Transfer

AUTHOR:

Yuchen Cao

ACCEPTED BY ADVISORY COMMITTEE:

[illegible] Daniel Cardoso Llach	Principal Advisor	DATE	May 10, 2023
[illegible] Ardavan Bidgoli	Advisor	DATE	May 10th, 2023

ABSTRACT

Architects and designers desire to put real-life scenes in the digital world to assist their work, such as experimenting with textures, meshes, and lighting conditions. Artists may create prototypes of art pieces based on digital scenes. The above practices mutually require 3D-style transfer technology. However, existing 3D reconstruction method such as photogrammetry suffers from slow computation, defects for large scenes with sparse inputs, and lacks support to content-aware post-processing. Existing style transfer mainly focuses on the 2D image style transfer, which does not work well in the 3D scene. This thesis aims at a lightweight and simple method to solve problems and meet practical needs. It takes Neural Radiance Field(NeRF) as a backbone and combines the 2D style transfer with the photo-realistic 3D scene from NeRF to achieve 3D style transfer. In methodology, it discusses the generation of the dataset and different approaches to combine NeRF and Style transfer network, optimization to improve the results. Besides, this thesis shows a valid comparison of different approaches from both analytic and aesthetic aspects. It further explores the applications that can utilize such techniques to achieve architecture design and art creation in a fashion of 3D-aware style control.

Keywords: NeRF, Style Transfer, 3D Reconstruction, Rendering, Photogrammetry, Computational Design

ACKNOWLEDGEMENTS

This thesis research would not have been possible without the support from the Carnegie Mellon University GuSH Research Grant, CD Research Support Micro-grant, and The Frank-Ratchye Further Fund Microgrant. I appreciate the support from Prof. Daniel Cardoso Llach, track chair of the Master of Science in Computational Design, Alycia Barney, Financial Assistant for the School of Architecture, David Koltas, Business Manager & Assistant Head for the School of Architecture, and Dr. Harrison Apple, Associate Director of Frank-Ratchye Studio, who have actively helped me to apply for the funding procedurally, and provide me reliable resources to push the thesis forward.

I am very thankful to each member of my Dissertation Committee, who has provided me with professional and generic suggestions and guidance on technological research, academic writing, problem analysis & solving, to help me tackle each challenge during the research and find correct directions. Prof. Daniel Cardoso Llach, my teacher and chairman of my committee, has offered me full support from computational design knowledge, and professional writing, to the research experience to learn system-level techniques and explore the thesis topic. I also want to appreciate the great help from Dr. Ardavan Bidgoli, who has given me plenty of guidance in application-level development, algorithm research, and even personal career suggestions. I feel so honored to stay around kind and generous committee members and accept support from them.

During my study at Carnegie Mellon University, I am proud of myself to carry out the slogan "My heart is in the work", I learned not only knowledge, but more importantly, to be devoted, devote oneself to one thing, to one domain, to one skill, and be excelsior at it, be expert of it, and good luck will find you. I want to especially thank friends I met at CMU, who refresh my definition of smartness, passion, maturity, humbleness, and courage. Ph.D. students Jingyang Liu and Yuning Wu from the CD program have broadened my horizons of combing design and science and taught me to be patient and humble. Ayush Jain and Nikolaos Gkanatsios from Robotics Institute have taught me the spirit to dive deep into computer science to explore the unknown, and the confidence and courage to face challenges. My classmates Zishen Wen, Yumon Zhuang, Haoyu Liu, and Jiayin Wei, have shown me what is a good mindset and persistence to work hard and work smartly. I've become better because of all of you.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Chapter 1

INTRODUCTION

This thesis discusses a series of experiments and applications around 3D reconstruction and style transfer technology in the domain of architecture, design, and computer science. Before I dive into it, let's first think about two questions: Why does 3D reconstruction matter? Imagining an ongoing construction site, the environment evolves along the construction procedure, and each dynamic phase is irreversible. One solution to reconstruct such a dynamic environment is to set up a series of cameras on tripods to capture the scene simultaneously. It can certainly provide valid information, but the angle of the camera view is limited, and it doesn't always cover all of the regions of interest.

Now, think about a solution that allows us virtually reconstruct a 3D site that is not only as high-fidelity as physical ones but takes only a few minutes for computing the memory shot of the entire scene. This proposal sounds intriguing since architects would find it easier to research the structure and textures of existing architectures, and interactively modify the environment for different effects and compatibility!

And the second question: why does the style transfer matter? One can get a photo-realistic 3D reconstruction from the above methods but they can't really do much of it – they can only view it, and maybe delete some part of it. Style transfer provides a way to modify the material, color tone, and surrealistic transformation to the real scene. From the artists' perspective, once the 3D reconstructed environment is adjustable, it means creative works can be conducted, and if the solution is fast enough, it can become a valuable tool.

Let's come back to talk about the solutions to style transfer in a 3D environment. To research the methods to approach the above proposals, the geometric structure from the motion and the radiance field neural network methods are experimented with and compared. From the experiment, it is clear that the learning-based NeRF(Neural Radiance Fields) has proven itself a better solution regarding fidelity, computing efficiency, and flexibility. Therefore, the NeRF-based method, which was first published in 2020 and became a popular topic in the academic circle focused on computer vision, becomes the focus and backbone of this research. The novel overfitting neural model opens a gate of reconstructing 3D scenes, re-rendering in free views, generative 3D models, and 3D style control, leaving lots of open end topics to explore and research.

The previous solutions either focus on 2D style transfer or 3D reconstruction, this thesis proposes and experiments with a series of new methods for combining the 2D style transfer with a 3D radiance field to achieve a 3D style transfer solution. To give a taste of it, the radiance fields to store spatial information enable the direct processing and embedding of extra information into the latent space, and here this thesis aims at mixing the feature from 2D style images into the photo-realistic 3D scene, which can be proceeded to not only reconstruct the physical world environment but also change the style variance, control the weight, combine multiple styles, and convert the 3D scene to styled meshes, styled point clouds to fuse with other scenes. And I believe such technology can benefit artists, designers, and architects in their creative tasks.

In the end, academic analysis is performed around the methods' performance of time efficiency, accuracy, artifacts and things to be improved in the future. And art pieces generated from such methods are presented to validate that they can assist in designing and creating art pieces. As an assistive tool for creating art, I also present my own art pieces as a programmer with no art background, and there is a VR application to experience the 3D stylish world as an example of the practical usage of such technologies.

Chapter 2

BACKGROUND

2.1 Challenges and Demands from Applications

3D reconstruction is a popular topic in both computer science and architecture domains. From the perspective of uses, the capability of free navigation inside a 3D authentic scene can solve spatial problems in AR/VR applications, for example, instead of building all the fine-shaped 3D models and rendering them in real-time in a VR device, the 3D scene can reduce memory to store the complex shape model but a much-compressed radiance field and support VR user to walk around the scene with their feet. Another application of AR example is object reconstruction, think that I reconstruct a 3D model of a dancer, and what I expect most is their unique movement, one way to do this is motion capturing the dancer and aligning the motion to their pre-built digital twin. It requires not only expensive motion capture devices but also 3D model artists to fine-tune the motion with models. With the learning method, I can use multiple cameras to capture the dynamic object simultaneously per frame. This allows cheaper investment and higher fidelity, and I can posit the 3D kinetic dancer in any plane via an AR device. Let's also think about the problem from an architect designer's perspective, the omnivore of physical-world objects is frequently referred to by architects and designers, however, a 3D model of large-scale architecture can be hard to access. The learning-based method allows users to scan the architecture with a phone camera or a drone, and reconstruct the scene in minutes, the designer can directly build content upon the reconstructed models. What's more, a 3D scene can be used for Forensic research and photographer post-processing, and a 3D model can be used by designers and architects for visualization, fitting parts to other models, supervising the construction site, and even putting into a game environment. Last but not least, in computer vision research tasks, the flexibility of the RGB dataset is always a pain because the field of view is constrained by the camera angle used in the capture session, what if I want a different view of the same scene? I can use the reconstruction method instead of finding the exact same sensor and going to the exact spot to capture it again. In summary, 3D reconstruction plays an important role in various applications in our life.

2.2 Limitation of Geometry 3D Reconstruction Method

The traditional SfM(Structure From Motion) methods are constrained by heavy and slow computation for dense clouds, and such reconstructed 3D models have artifacts that easily discriminate them from realistic objects. Besides, the rendering task with such 3D models takes extra effort in tuning lumination and environment settings. The geometric-based photogrammetry method has been matured and widely used as an industrial-level solution, for example, software Agisoft Metashape, VisualSFM[20, 19], and ColMap[15, 16]. Such methods extract features of RGB images, match across different frames, and get 3D position by calculating triangulation from the correspondence of matched feature points, and use bundle adjustment, loop closure, and graph optimization methods to adjust the map and camera pose to be more accurate. The method requires a large number of features from each image and goes through matching for each feature, when the input resolution is high and the scene is significant (such as a teaching building), it takes tens of hours to calculate even with modern high-performance GPU and CPU. Although the reconstructed model is proven to have great details, it suffers from reconstructing the background(incompleteness because of insufficient data points or occlusion), the incoherence of textures through the different light conditions and color temperature, difficulty in re-rendering an authentic scene from the reconstructed model.

2.3 Development of Learning Method

As deep learning in computer vision tasks has developed rapidly during the past 10 years, some state-of-the-art learning-based methods have boomed up. They look at the 3D reconstruction problem from different perspectives: Take the first NeRF work published in 2020[10] as an example, their network is trained with a sequence of RGB images and their camera poses and the network is unique in a way to aim at over-fitting the model with a specific scene so that the network's parameters become a storage of 3D spatial information, as known as Radiance Field. In the test, it takes 5-dimensional inputs including 3D location and 2D viewing directions as input, the 5D tells the network model from which the virtual camera pose and angle to render the image and interpolate between keyframes to output a successive video, which provides customized camera angle different from the dataset used for training. Besides, they introduce the volumetric rendering method that can accurately render physically realistic images in a reasonable time(tens of hours) with modern GPU. The volumetric storage is differentiable and critical as a deep learning model. Compared to the geometric method[10], the two types of solutions have a similarly heavy computation time but the learning methods have a higher fidelity and can solve certain challenges of traditional geometric methods, such as glossy surfaces, mirrors, and the accuracy of complex geometry.

There are certain limitations[9] of the primary NeRF and geometric methods, where they can not process the occlusion and blind spots caused by collisions or the sparsity of inputs so well, which can lead to artifacts because the network can't present the scene it doesn't know. Methods such as PixelNeRF[21] and IBR-Net[18] are generative versions of NeRF that pre-trained models accept different scenes' raw and sparse images and still guarantee an authentic output by using prediction models to interpolate the "unknown" area based on the information gathered from the neighborhood pixels.

Another big issue of the dataset to feed to NeRF network is its variance in illumination and disturbance from the dynamic outlines, for example, while holding a camera to capture around architecture, the cloud blocks the direct sunlight and leads to different lumination on different camera angles; and in a popular sightseeing spot, lots of pedestrians can intrude in the camera view, and adds extra occlusion, even with prediction model, the model can't tell if the pedestrians are part of the architecture or not, it will eventually output shadows or the unwanted humans. To solve such an issue, a variant of NeRF like NeRF-W[9] effectively smooths out the illumination variance and removes the pedestrians.

And considering the perspective of 2D style transfer, the limitations are that the generated style is randomized, meaning one can not get exactly the same two outputs by running twice with the same input. This makes it impossible to use 2D style transfer on a 3D scene, since every view has a totally different style, and the overlapped area doesn't maintain its consistency.

Last but not least, with the most recent research and engineering work from NVIDIA, they present Instant-NGP[12] as a "5 seconds" open source CUDA implementation to change the game, it solves one of the biggest computation speed issues of such methods. Besides, it combines Normal maps, depth maps, and 3D models altogether, which opens lots of possibilities for my further research.

To conclude the above points, NeRF-related works present a brand new way by using a radiance field to represent the 3D information and using ray tracing methods to render the image. It makes great use of modern GPU and achieves a better performance than the traditional geometric.

Table 2.1 summarizes the brief advantages and disadvantages of the major 3D reconstructions mentioned above. And these are also the major methods that will be explored in this thesis.

	Type	Advantages	Disadvantages	Year
MetaShape	SfM	Friendly UI Accuracy	Hard to batch-process & Slow	2010
ColMap	SfM	Nice API Accuracy	Slow even using GPU	2016
NeRF-W	NeRF	Pytorch: easy to change code	Slow with high resolution hard to visualize	2021
Instant-ngp	NeRF	Good visualization Well optimized	CUDA: Hard to change code	2022

Table 2.1: Summary of existing 3D reconstruction methods.

2.4 Art-oriented Learning Method

To extend the backbone NeRF network, the artistic style transfer method is explored. There are different ways to insert the style embedding into the existing network: an image-based neural style transfer with constraints from the nearby frames to maintain the smoothness and coherency; and a radiance field level processing of end-to-end methods to take raw images input and camera poses and outputs, and an art image as reference to guide the overall style, and output the artistic videos and 3D models.

The first image-based methods can be embedded into the existing NeRF backbone in different positions, it can either pre-process the images before feeding them into the network, can also post-process the outputs from the NeRF, both ways are explored in this thesis to compare their performance. A typical style transfer network such as Stylizing Video by Example[6] provides a toolkit pipeline to allow artists to fine-tune several example frames with a painting tool and then propagate to the rest via computer graphics method of finding correspondence among adjacent frames. Such a method requires certain user input together with the art style reference in exchange for accurate and elaborate stylization among the frames. And there are also more automatic ways of relying on deep learning networks. For example, Real-Time Style Transfer[7] and Artistic Style Transfer for Videos[14] use neural style transfer model to automatically transfer a style from one image to a video without changing the content of the video. Such work takes the correlation of adjacent frames as constraints to maintain the coherency and avoid artifacts among frames.

The second end-to-end way can further automate the pipeline, and the radiance field embedding can not only provide a more distinguishable style alignment for different contents but also allow other purposes of 3D style change such as modifying the 3D model's appearance, where video rendering is a part of the output. Such work includes Artistic Radiance Fields[22], UPST-NeRF[2], and Stylizing 3D Scene via Implicit Representation and HyperNetwork[3] which all consider the problem from the perspective of latent space of NeRF model, i.e. the radiance field, and train the style embedding network at the same time along the NeRF network. However, introducing another network bring more difficulties in tuning the architecture and hyper-parameters, and slows down the network training.

Table 2.2 summarizes the brief advantages and disadvantages of each style transfer solution, and the first three methods will be discussed and explored in this thesis.

	Type	Advantages	Disadvantages	Year
2D style transfer	2D	Real-time	Inconsistency	2015
Video style transfer	2D	Consistency among short frames	Hard to deal with long input	2016
ARF	3D	Good 3D art style	Train network 2 times	2022
UPST-NeRF	3D	Consistency among views	Less art style	2022

Table 2.2: Summary of existing style transfer methods.

2.5 Multiple Network Combination

This research task involves multiple learning-based neural networks, therefore the combination and concatenation of different networks together as a universal pipeline are also explored and discussed. my goal is eventually an applicable end-to-end development without requiring users to worry about intermediate data, as long as they follow the easily understandable tutorial for capturing the dataset.

The methods of combining different networks are separated into two types in this thesis, one is end-to-end concatenation, and another is embedding the feature latent variables from different networks, training the loss of different networks together, and then outputting the styled 3D representations in the rendering section. Comparing the defects and performance of different combination methods would determine which is my final approach to be developed as an application. Overall, combining different networks is a challenging task because each network has a different scale, loss function, propagation method, and its own hyper-parameters, it's a tricky task to find in which position to combine two networks, how to turn two types of data into the same format, how to define the loss function, and if I need to freeze part of the network during the training, etc. Luckily, the style transfer network has a unique pattern to follow, and with the help of lots of related papers and open-source code, this challenge becomes solvable.

2.6 Novel Ideas

I have discussed the advantages of NeRF and the limitation of 2D style transfer. What would happen if I mix them together? This thesis conducts research on reconstruction problems based on the backbone NeRF network and combines it with 2D style transfer methods to explore the possibility of generating 3D-styled scenes. Although there are other works published in 2022 and 2023 focusing on similar topics, this thesis stands on its unique ways of comparison of different methods with a good number of experiments running with the same dataset that makes the results more convincing, a novelty in the combination of 2D style transfer and NeRF and apply post-processing to optimize the results. It also proposes and explores the potential applications of such methods to further deliver the practical use of these concepts.

Chapter 3

HYPOTHESIS

Research Questions

Based on the exploration from the last chapter, it is not hard to come to the research question: how can we combine existing solutions of 2D style transfer and Neural radiance field together to make the 3D style transfer possible? And how are such solutions different from the existing works, what are the advantages and disadvantages? And lastly, How does it benefit in developing related applications, and what application can be developed?

Importance

Why does it matter? A style-adjustable 3D scene will change the way people create 3D art pieces, view 3D scenes. For computational designers, it's a research topic combining art design and computer science, it'll be a handy tool to use code to do 3D art and also can be a backbone to develop more features.

Solutions

To address the above questions, this thesis first proposes 4 approaches to experiment with the idea of combining the 2D style transfer with neural radiance field, including concatenation of output of style transfer with NeRF, video style transfer with NeRF, concatenation of output of NeRF with style transfer, and embedding style feature and radiance field in latent space. It also explores the way to improve the consistency of adjacent styled frames and the denoising method.

Tests & Practices

Since the output is more leaned to the subjective sense, it is not easy to compare different methods, this thesis discusses the criteria of 3D style transfer by comparing 4 approaches with each other to show the improvements and limitations.

In the end, it uses one of the most fitting approaches to generate pieces of art and put them in a VR environment as a way to deliver its practical usage. And it proposes potential applications with such techniques that can have a great impact on the industry and the possible approaches to achieve them.

Chapter 4

METHOD

Overview

The method of this thesis research consists of two parts as shown in Figure 4.1: 3D reconstruction and style transfer. Each part contains multiple approaches to demonstrate the progress of exploration and comparison to find the best approach for my specific tasks.

First, it explores the 3D reconstructions with geometric and learning-based methods and compares them for each one's strengths and weaknesses. The geometric methods have been quite mature for the past 10 years and put into industrial applications, yet the learning method has rapidly grown for the recent 2 years and gained lots of attention from researchers under the wave of deep learning. my approach focuses on the learning-based solution for its great potential of adjustable 3D data.

Secondly, it experiments with the style transfer network and seeks to drive from 2D style transfer to 3D style transfer with the backbone of learning-based 3D reconstruction methods. Similarly, 2D style transfer is a mature solution for the past 5 years, yet 3D style transfer still has wide space for exploration. And then it uses different solutions to deal with defects during the experiment to improve the performance.

Thirdly, it shows examples of applications that can benefit from such techniques to validate the value of such approaches. And I comprise a pipeline from acquiring the dataset to training the network and getting the final result for practical purposes.

4.1 Geometric 3D Reconstruction

A traditional solution of 3D reconstruction aims at finding the same pattern from multiple 2D maps captured by various sensors, and estimating the 3D data, such a method is generally referred to as Structure from Motion(SfM) or Multi-view Geometry(MVG), as shown in Figure 4.2. Depending on the types of 2D maps, the RGBD data and stereo-RGB data directly provide the depth information, which can reduce the computation and improve the accuracy; the mono-RGB data would calculate the depth by finding the correspondence among nearby keyframes and doesn't have the scale information. And because of the popularity and convenience of

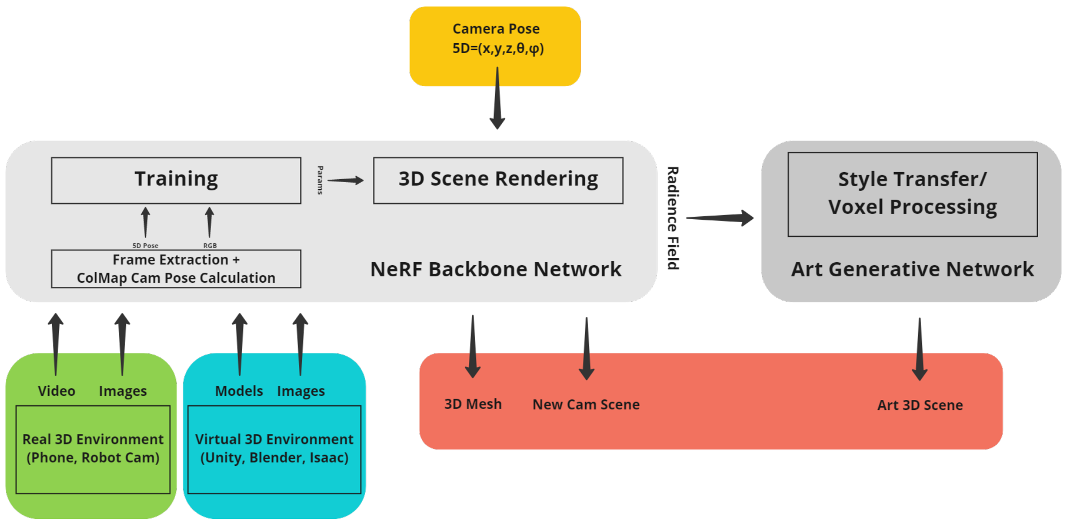

Figure 4.1: Diagram of 3D style NeRF pipeline.

(Image by the author)

the mono-RGB sensor, lots of popular photogrammetry platform provides accurate and high-fidelity solution for mono-RGB data. Examples are Agisoft Metashape[1], ColMap[19, 20], OpenMVG[11]. In this thesis, I use Agisoft to 3D reconstruct the same dataset as a comparison with the learning-based methods.

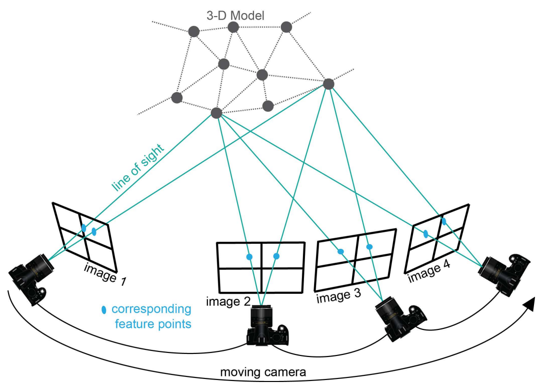

Figure 4.2: Structure from Motion.

(Shervais, K. (2016, October 18). Structure from Motion (SfM) Photogrammetry Field Methods Manual for Students. Education AT Unavco.Org. [pdf])

4.1.1 Structure from Motion

To illustrate the geometric 3D reconstructions mentioned above, it is necessary to look at the core algorithm of them – Structure from Motion. The SfM can explain how the geometric solutions work, show us why certain camera movements during the capture are bad for reconstruction, and give instructions for preparing the dataset for this thesis.

I will only give a brief introduction to SfM since it's general knowledge in computer vision and not my focus of this thesis. As shown in Figure 4.3, c and c' represent the center of two camera viewpoints. One can use any different feature extractors to process the image, here I focus on a popular one – Scale-invariant Feature Transform(SIFT). By comparing the features of two images, SIFT can find the matched feature, called correspondence, e.g. x and x' in the graph, then X is the 3D point of my goal. However, it doesn't know the relative position from c' to c, thus it couldn't directly get X, because it only knows X is a point along the line that connects c and x. Now if it knows the epipolar line that connects e and e', it is able to get accurate X.

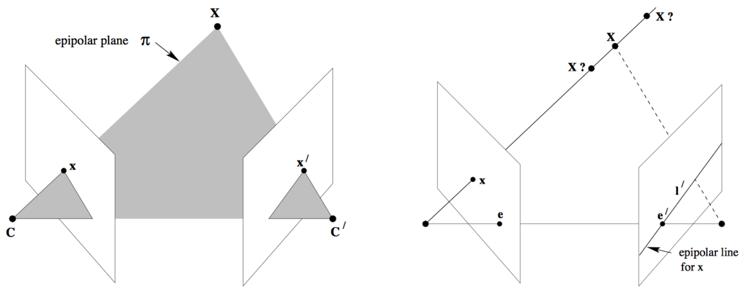

Figure 4.3: Demonstration of Epipolar lines.
(Deep Singh, C. (n.d.). Structure from motion. From https://cmsc426.github.io/sfm/)

In order to get the epipolar line to get the correct X as shown above, one solution is to take many sample-matched feature points in two images and use the Random sample consensus(RANSAC) to filter out outliers(mismatched points) and use the function below to calculate the relative pose from c' to c. Since the Fundamental Matrix F has 8 degrees of freedom, it at least requires 8 pairs of matched points. Once getting F, one can get ee' and X, as shown in Equation 4.1. Such progress applies to every point and image during the SfM algorithm, and it gives coordinates of both 3D points and cameras.

$$[x', y', 1] \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0 \tag{4.1}$$

4.1.2 Data Preparation

On the other hand, the exploration of SfM methods prepares for the dataset processing for the deep learning methods. In other words, this thesis will still rely on the SfM method to estimate the pose of the camera since most of my data are mono-RGB that doesn't have localization data. To capture the data, one can either mount the camera on the robot or hand-hold the phone camera depending on the scenario of applications. For example, if one wants to apply it to a construction site, it would be ideal for mounting it on the robot to gather the dataset while it's automatic navigation and assistance. And if one just tries to reconstruct a casual scene, they can simply use their phone. The critical part of dataset preparation is to maintain the intrinsic camera parameters identical throughout the entire process of capture, any lens adjustment, aperture, and shutter change can lead to false pose estimation for the later process. And since I use mono-RGB as input, there are also limitations for pure rotation movements that are hard for the SLAM algorithm to calculate triangulation.

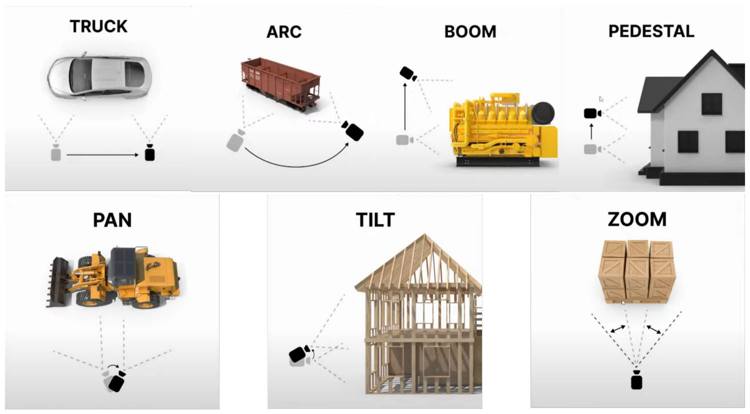

Figure 4.4: Dataset preparation examples of good captures(up) and bad captures(down).

(EveryPoint. (2022). NVIDIA instant nerf advanced tips. From YouTube:_xUlxTeEgoM)

With the RGB data captured by the phone camera, I experimented with ColMap to get results in Figure 4.5, the reconstruction data will be compared with learning methods, and also the camera pose will be used for training the learning methods that will be discussed in the next section.

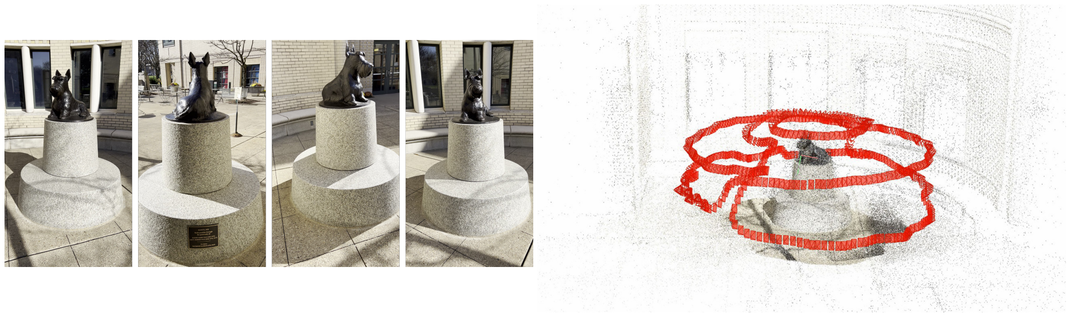

Figure 4.5: Scotty dataset(left) with SfM, and the output(right) of point clouds and camera poses.

(Image by the author)

4.2 Learning-based 3D Reconstruction

A series of learning-based methods have taken over top conferences since 2020 and constantly becoming faster and more accurate, such methods are generally called Neural Radiance Field(NeRF). As the NeRF has developed so fast, more novel research papers have proved to outperform the first network I used at the beginning of this project. So you will see a couple of learning-based NeRF networks get experimented with in this thesis until a solid back-bone network is found.

NeRF network generally aims at training a neural network to store the RGB and density information from the given dataset of RGB images and 5D camera poses. And for rendering, it uses Volume Rendering to accumulate the density and RGB value along each ray to get images from any perspective, as shown in Figure 4.6.

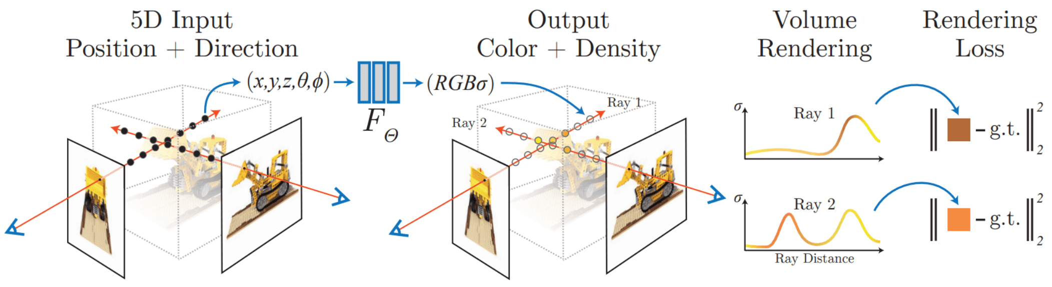

Figure 4.6: Neural Radiance Field.

(Karagiannakos, S. (2022, November 25). How Neural Radiance Fields (NeRF) and Instant Neural Graphics Primitives work. Sergios Karagiannakos. https://theaisummer.com/nerf/)

Notice in the graph, F_θ represents the neural network that contains the radiance field information, the forward passing of the network converts the input of (x, y, z, θ, ϕ), where x, y, z is the position of the camera, θ, ϕ is used to describe the direction of one ray from the camera position. I use C to represent the RGB value of a pixel, ρ to represent the density, $r = o + td$ to represent the ray for rendering a pixel, Δ means the distance between two sampling points along the ray, t_k means the kth sampling point along the ray, I can get the equation 4.2.

$$C(r) = R(r, c, \rho) = \sum_{k=1}^{K} T(t_k)\alpha(\rho(t_k)\Delta(k))c(t_k) \tag{4.2}$$

$$\alpha(x) = 1 - e^{-x}, T(t_k) = e^{-\sum_{k'=1}^{k'=k-1} \rho(t_{k'})\Delta(t_{k'})} \tag{4.3}$$

The above equation mainly describes that it accumulates the RGB and density from the camera's original point to the 3D radiance field to get the final RGB for a pixel.

And for the loss function, I can think of it this way, for each pixel of a given camera position, I can render an RGB value from the above forward propagation, and then I take the images of this position as ground truth and calculate their difference.

$$loss_{nerf} = ||F_\theta(x, y, z, \theta, \phi) - C_{gt}|| \tag{4.4}$$

4.2.1 NeRF-W

At the beginning of the project, I experimented with NeRF-W[9] published in 2021, it has a faster computation compared to the first NeRF paper[10] by Ben Mildenhall and Pratul P. Srinivasan, et al. The more important factor I considered in this thesis is that my goal includes the 3D reconstruction of the outdoor environment. In the outdoor environment, the color temperature, shadow, and lightness change frequently by both time and position of the camera, on the other hand, mobile objects such as pedestrians and other occluders are something not interesting in the scope of this thesis.

NeRF-W is built based on NeRF and has the capability of removing the disturbances such as pedestrians in the reconstruction by adding noise at the bottom of the image where the occluders appear most frequently and smoothing out the illumination and white balance by introducing the Generative Latent Optimization to replace the color embedding with an image dependent color radiance so that they can interpolate color between frames. In detail, one major difference between NeRF-W from NeRF is that they replace c with c_i, where c_i is image-dependent for each index of image i.

I was able to reconstruct the customized video sequence of the MMCH building on the CMU campus. However, it took over 10 hours simply train the network and render the photo-realistic images with 640x480 resolution in a modern GPU 2080 Super with 8GB memory. And there are certain defects in the output, so I decided to move to a better network.

4.2.2 Instant-ngp

In 2022, Nvidia published a state-of-the-art paper and open source code Instant-ngp[12]. From the perspective of the algorithm, it uses hash-coded multiresolution latent variables to speed up the computation, which simply means that each resolution layer has a pre-defined hash table, so when calculating the pixel in different resolutions, it doesn't have to maintain a large network with float points, but instead, just check the table with a smaller network, a smaller network means more parameters can be put into parallel computing and also faster computation itself. Besides, by doing so, it allows the network to learn the coarse and fine features altogether, which improves the quality and details of the radiance field. And from the perspective of engineering, it's implemented in CUDA to further fully use the computation units of the modern GPU to achieve faster computation.

I experimented with the code for reconstructing high-resolution photo-realistic rendering, 3D mesh, normal maps, depth maps, etc. And compare it with the geometric solution, I came to the conclusion that Instant-ngp is a better option for my goal to embed artistic 3D style in the 3D data. Therefore, the rest of the research and experiment in this thesis takes Instant-ngp as the backbone. Figure 4.7 below briefly summarizes the function of the NeRF.

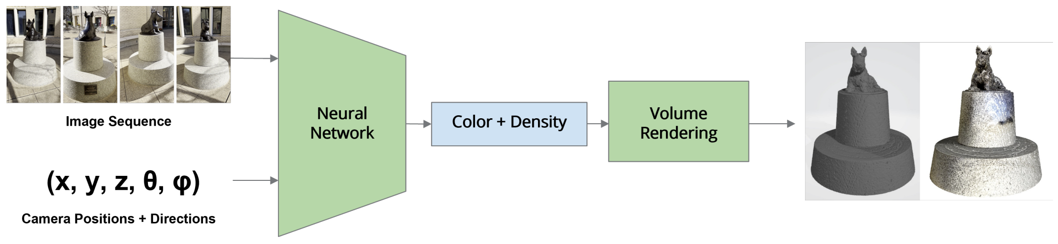

Figure 4.7: NeRF functionalities.

(Image by the author)

4.3 2D Neural Style Transfer

Next, I move toward style transfer. Image-to-image style transfer is not a new topic and has developed into a mature technology, such as Neural Artistic Style[4] and Real-Time Style Transfer and Super-Resolution[8]. Generally, the style transfer neural network contains two parts: style feature and content feature, two similar VGG-19 layers are used to extract the two features separately, and then the network embeds two features in the convolution layers, and the balance between style and content is fine-tuned by weight hyperparameters.

In detail, the loss function of content and style are composed differently. For content, I want to generate an image that has similarity to the original image, therefore I can pick a hidden layer in the middle, or just take the output image to do the pixel-level difference. Here I take the hidden layer l so that the loss can be calculated by Equation 4.5. P^l_{ij} means the generated features in layer l, Q^l_{ij} means the features of original RGB image in layer l.

$$loss_{content} = \frac{1}{2}\sum_{ij}(P^l_{ij} - Q^l_{ij})^2 \tag{4.5}$$

The loss of styles can't be calculated with the difference between generated images and style images, because I don't want the generated image shows content level similarity with the style image, but I want a certain pattern to be applied to the generated image. Therefore, I take the Gram Matrix of the style image to show the overall distribution of the features. I still take hidden layer l, where there are k channels, and H is the height, W is the width, I can calculate the loss function in Equation 4.6. g is the generated feature, s is the style feature in latent space. A is the value of one element of one filter in latent space.

$$loss_{style} = \sum_{l} w^l loss^l_s \tag{4.6}$$

$$loss^l_s = \frac{1}{HW}\sum_{i}^{H}\sum_{j}^{W}(G^l_{ij}(s) - G^l_{ij}(g))^2 \tag{4.7}$$

$$G^l_{ij} = \sum_{k} A_{ik}(I)^l A^l_{jk}(I) \tag{4.8}$$

After I get both losses, I can calculate the total loss by Equation 4.9. α and β refer to the weight.

$$loss_{total} = \alpha loss_{content} + \beta loss_{style} \tag{4.9}$$

In this thesis project, I first implemented a neural style transfer network in PyTorch and then referred to the open-source style transfer https://github.com/crowsonkb/style-transfer-pytorch to improve my model. The development of 2D style transfer serves as a benchmark to evaluate my 3D solutions, and also be repeatedly used in 3D exploration. The diagram of 2D style transfer is shown in Figure 4.8.

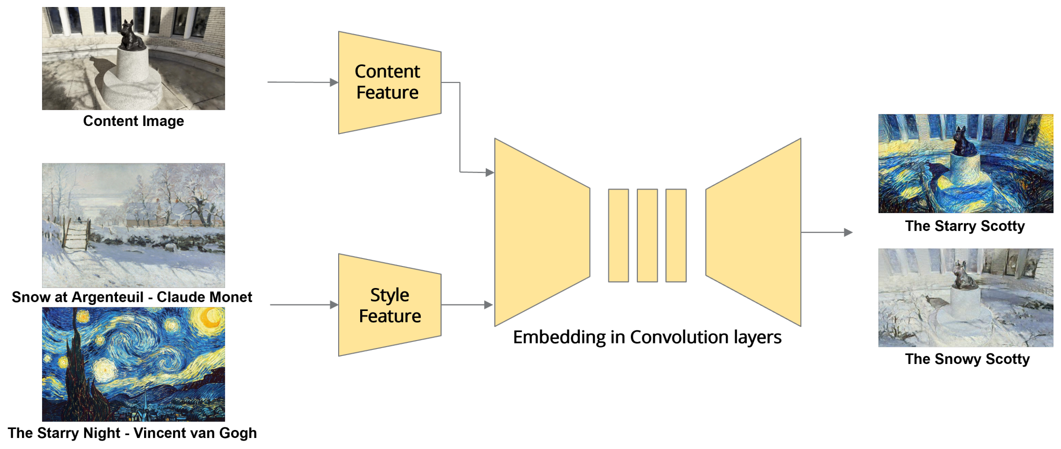
Content Image
Snow at Argenteuil - Claude Monet
The Starry Night - Vincent van Gogh
Content Feature
Style Feature
Embedding in Convolution layers
The Starry Scotty
The Snowy Scotty

Figure 4.8: 2D style transfer functionalities.

(Image by the author)

4.4 3D Style Transfer

To step into the 3D style transfer, first, it is reasonable to think about why it is a challenge to do style transfer in a 3D environment. Intuitively, 2D style transfer has done a great job and it even works in real-time videos, it seems the 2D image plus 3rd timeline dimension works well. But not really, if you look closely into such type of work, you can discover the inconsistency among frames, each image is styled in different details, and when two frames have no overlapping pixels, the same object doesn't have a consistent style pattern. Therefore, I explore three methods to combine the previous research and experiments for 3D style transfer, so that it applies the style to the object itself instead of just one image.

4.4.1 Style Transfer Concatenated with NeRF

The first experiment is to concatenate the output of the 2D style transfer network with the input of NeRF as shown in Figure 4.9. As the most intuitive way of thinking, the

NeRF network trains with a photo-realistic dataset and generates a photo-realistic 3D scene for free viewpoints of rendering. I can expect that the NeRF network trains with the stylish dataset and generates 3D stylish scenes. Therefore, I first use the 2D style transfer network to generate the 2D style images and feed them into the NeRF network. Since the inconsistency of the pixels among adjacent 2D stylish images lead to failure of finding the correspondence in calculating the camera poses, I still use the camera poses of the photo-realistic ones, and experiment with how well NeRF can handle the inconsistency.

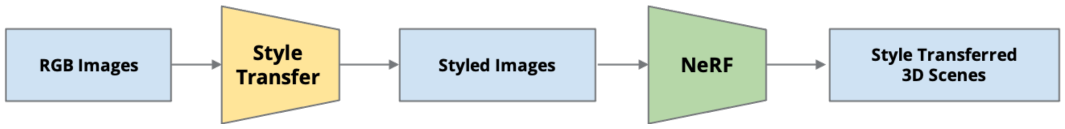

Figure 4.9: Framework of the concatenation of style transfer and NeRF.

(Image by the author)

In detail, it first calculates the camera pose of each RGB image by ColMap and then applies style transfer for each RGB image with the same resolution, and uses the camera pose and generated style RGB to train the NeRF.

There are 4 experiments proposed and conducted in this thesis to pursue the idea of 3D style transfer, a summary is shown in Table 4.1.

	Descriptions
Expt. 1	Concatenate output of 2D style transfer to input of NeRF
Expt. 2	Similar to Expt. 1, add adjacent consistency to style transfer
Expt. 3	Concatenate output of NeRF to input of 2D style transfer
Expt. 4	Embed 2D style feature and radiance field from NeRF in latent space

Table 4.1: Summary of four experiment methods.

4.4.2 Adding Video Style Transfer Constraints

From the first experiment, I already notice the potential issues that directly using 2D stylish images in NeRF can cause to the training process. I consider the methods that can maintain consistency among adjacent frames, a typical type of related work is either to train the long sequence of multiple frames together and add constraints to the style loss, such as Artistic Style Transfer for Videos and Spherical Images[13]. And a different way to train is to give the network a sample of the stylish frame and spread this style to its adjacent frames, such as EbSynth[5]. I experimented with both as shown in Figure 4.10, and get to the conclusion that such methods don't handle the 360 degrees of the scene so well, the ideal condition for video style transfer to effectively spread the consistency of styles is planar movement without many rotations.

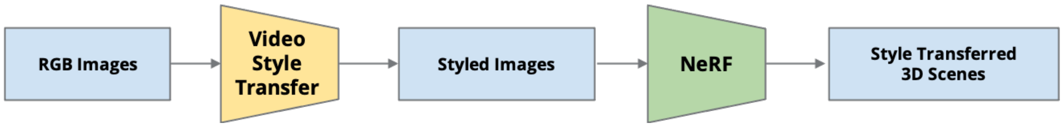

Figure 4.10: Framework of the concatenation of NeRF and video style transfer.

(Image by the author)

In detail, the video style transfer is very similar to the 2D style transfer loss function discussed above, except that they add one more loss constraint called temporary loss in Equation 4.10, which aims at adding constraints among the adjacent frames by detecting the consistency from forward-backward optical flow, it can be either short-term or long-term.

$$loss_{temporary}(x^i, w^i_{i-j}(x^{i-j}), c^{(i-j,i)}) = \frac{1}{W * H * C} \sum_{k=1}^{W,H,C} c_k * (x_k - w_k)^2 \quad (4.10)$$

x_k refers to the current styled image with k index, x^{i-j} is the i-j frame styled image, $w^i_{i-j}(x^{i-j})$ is the forward-backward optimal flow consistency weight, $c^{(i-j,i)}$ means the weights between frame i and frame $i - j$, where it is set to 0 in dis-occluded region else it is set to 1.

4.4.3 NeRF Concatenated with Style Transfer

The third experiment is inspired by the first experiment – How about I flip the order of concatenation? I concatenate the output of the NeRF with the input of the 2D style transfer network, as shown in Figure 4.11. This method doesn't directly generate 3D scenes but can be taken as a comparison and reference of how other methods make their difference.

In detail, I first train the NeRF with camera pose acquired from ColMap and original RGB images, and for the output of rendering, I apply style transfer for each of them.

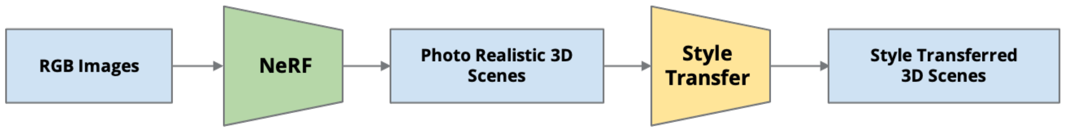

Figure 4.11: Framework of the concatenation of NeRF and style transfer.
(Image by the author)

4.4.4 Embedding Style and Radiance Field in Latent Space

The fourth experiment is inspired by the structure of the 2D style transfer neural network. In the 2D style transfer, it extracts the content feature and style feature and embeds them together in the latent space. Similarly, I can embed the 2D style feature and latent space of the 3D scene in NeRF together and train them together, as shown in Figure 4.12. Similar works have been published just in 2021 and 2022, such as Stylizing 3D Scene via Implicit Representation and HyperNetwork[3], ARF: Artistic Radiance Fields[22], and UPST-NeRF: Universal Photorealistic Style Transfer of Neural Radiance Fields for 3D Scene[2]. In this thesis project, I refer to the work of ARF.

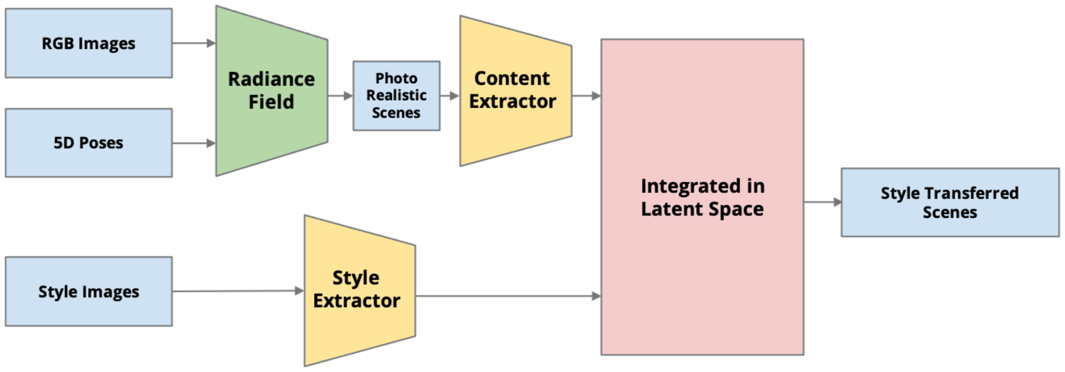

Figure 4.12: Framework of the embedding style feature in 3D latent space.
(Image by the author)

In detail, similar to the 2D style transfer's losses equations I discussed above, it uses the same way to extract the content feature and calculate the content loss in Equation 4.11, where R is rendered image from NeRF, and I is the original RGB image.

$$L_{content} = \frac{1}{2}\sum_{ij}(R^l_{ij} - I^l_{ij})^2 \tag{4.11}$$

The difference is the way to calculate the style loss. In the ARF paper, they use the Nearest Neighbor Feature Matching(NNFM) instead of the Gram Matrix, since Gram Matrix considers the global features, it can ignore local information, and the local information is important in maintaining the consistency in the 3D scene. This NNFM can focus more on the local features, the Equation 4.12 shows the style loss. S refers to the feature of the style image, and R refers to the feature of the rendered image. $D(x, y)$ calculates the distance of vector x and y.

$$L_{style} = \frac{1}{N}\sum_{ij}\min_{i,j} D(R^l_{ij} - S^l_{ij}) \tag{4.12}$$

$$D(x, y) = 1 - \frac{x^T y}{x^T x y^T y} \tag{4.13}$$

And I can calculate the full loss by Equation 4.14.

$$l_{total} = \alpha l_{content} + \beta l_{style} \tag{4.14}$$

Such a method requires training the NeRF network twice, for the first time, it trains the NeRF with the photo-realistic dataset and then renders the image from the train scene. And for the second time, it trains with the style feature and content feature together to generate a new radiance field. I can acquire good consistency among different frames since the style feature is added to the radiance field of the 3D scene.

Chapter 5

RESULTS

Overview

In this chapter, I put the experiment results together and compare their differences. The experiment divides into two parts: 3D reconstruction experiments and 3D style transfer experiments. And I analyze the performance and efficiency and choose the solution to conduct my own art pieces, show the demo of a potential VR application, and discuss the solutions and results.

Most of the experiments are run in a Windows system with a 3070Ti GPU, and 8GB VRAM computer, except the 4th experiment, which is run on an EC2 server with g4dn.4xlarge cloud server on Amazon.

And the dataset, as shown in Figure 5.1, is the hand-held capture of Scotty Statue on the campus of Carnegie Mellon University, Pittsburgh. The total number of input images is 561, each image has 1080*1920 resolution, in a wide lens mode(with distortion on the edge), in the same exposure, white balance, and focal lens, captured in portrait mode by iPhone 13 Pro. For the rest of the experiment that uses the Scotty Statue as a dataset, the parameters are all the same. The dataset for the NeRF-W experiment is captured in the same parameters except for the landscape mode and the number of images as 50.

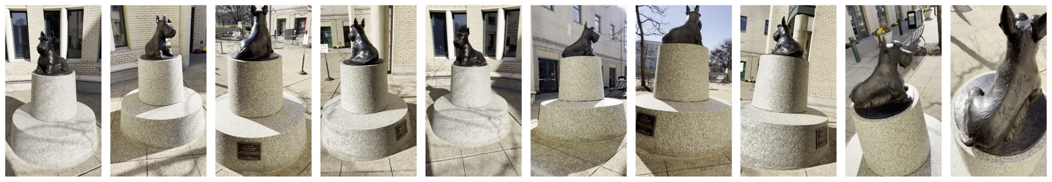

Figure 5.1: Dataset of hand-held capture of Scotty Statue at CMU.

(Image by the author)

5.1 3D Reconstruction Experiments

As the first part of the experiment, one geometric solution and two learning solutions of 3D reconstruction are conducted. I use 3D mesh and rendered images as examples to demonstrate the comparison of the results of such methods.

5.1.1 Geometric Method

I chose Agisoft Metashape as a representation of Structure from the Motion algorithm for the experiment. The geometric method has advantages in good granularity, smooth mesh, yet I will focus on its drawbacks. The major issue with such a method is the defects in incorrect 3D information for distant backgrounds and the collapse of the mesh for the area that has few correspondences from the inputs. Figure 5.2 demonstrates the defects mentioned above.

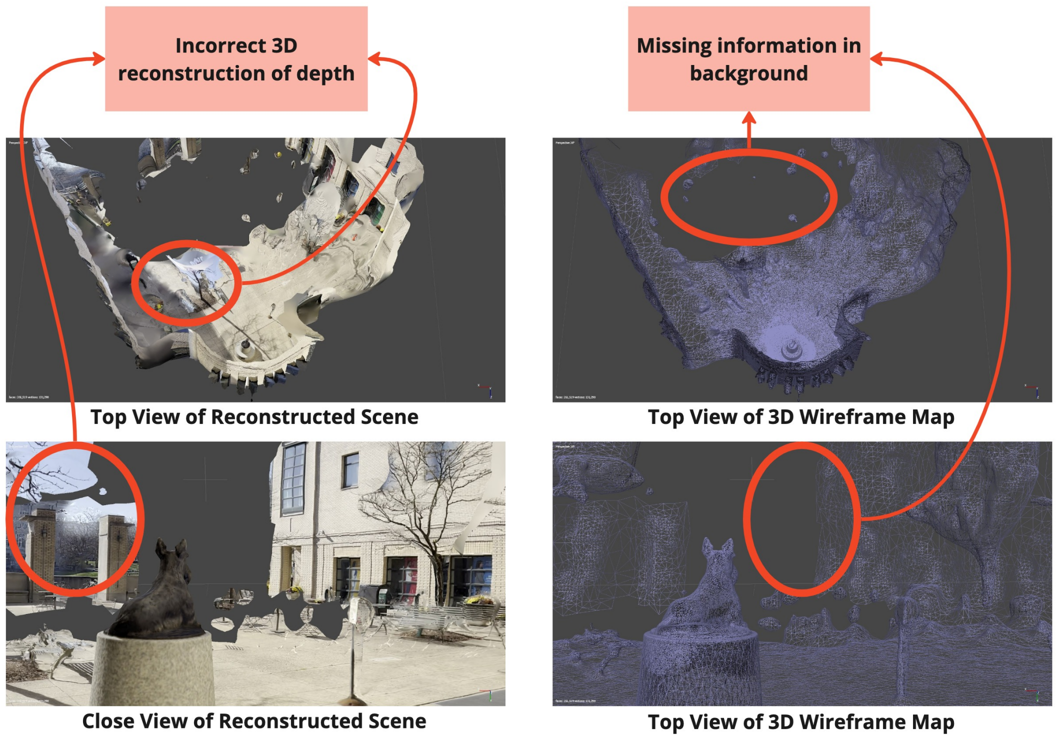
Incorrect 3D reconstruction of depth
Missing information in background
Top View of Reconstructed Scene
Top View of 3D Wireframe Map
Close View of Reconstructed Scene
Top View of 3D Wireframe Map

Figure 5.2: Defects of geometric methods.

(Image by the author)

From the circles on the left, I can see the background is in the wrong position, the sky, and the distant building is reconstructed in a very close place to the Scotty Statue. And the circles on the right show the large area of the background has no meshes or wireframes. And I will see the comparing rendered outputs from NeRF discussed below that have a better performance.

5.1.2 NeRF-W

The first experiment of NeRF I used was the code of NeRF-W, and because it was at the early stage of NeRF works, it suffers from computing efficiency and resolutions issues, which doesn't help it outperform the geometric solution in the area I are interested in, therefore I use it as an illustration of how to train the NeRF and using a small dataset with only 50 images.

To train the dataset in NeRF network, I first need to pre-process the raw RGB images to get camera poses by ColMap. And during the rendering, I customize any camera positions and orientations as inputs, shown on the left of the 5.3, and outputs the rendered graph on the right.

I can tell that the outputs are photo-realistic but in a very low resolution. And if I increase the resolution, the GPU throws exceptions as running out of memory and also takes an extremely long time. Therefore, I didn't move further to train other datasets, but moved on to the Instant-ngp.

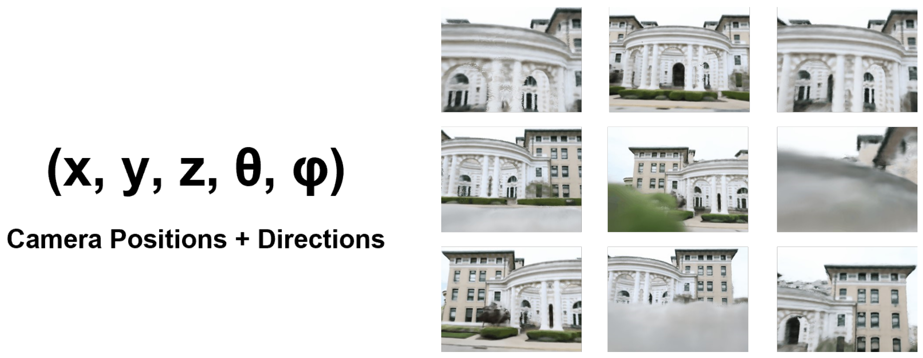

Figure 5.3: The input(left) and output(right) of the NeRF-W training, the example takes data of MMCH building in CMU.

(Image by the author)

5.1.3 Instant-ngp

I can use the exact same steps of pre-processing the dataset as described in NeRF-W. And this time I used the same Scotty Statue dataset as the geometric solution to better show the difference between the two solutions, because Instant-ngp has a better-optimized network architecture and usage of GPU.

And this network can not only generate photo-realistic images but generate mesh, normal maps, and depth maps. The examples of output from Instant-ngp can be found in Figure 5.4. And not only the resolution is $1920 * 1080$ but also the rendering is at the second level, meaning that it is a valid backbone for the 3D style topic exploration.

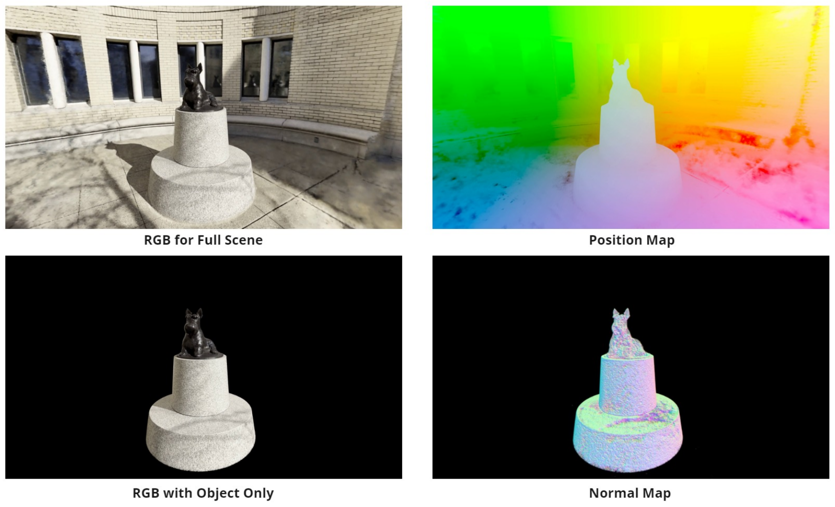

Figure 5.4: Different types of outputs from Instant-ngp network, including the RGB rendering, position map, normal map, and isolated object.

(Image by the author)

To compare the performance between the geometric method and the learning method, I first take the rendered RGB from a similar perspective and compare their difference, as shown in Figure 5.5.

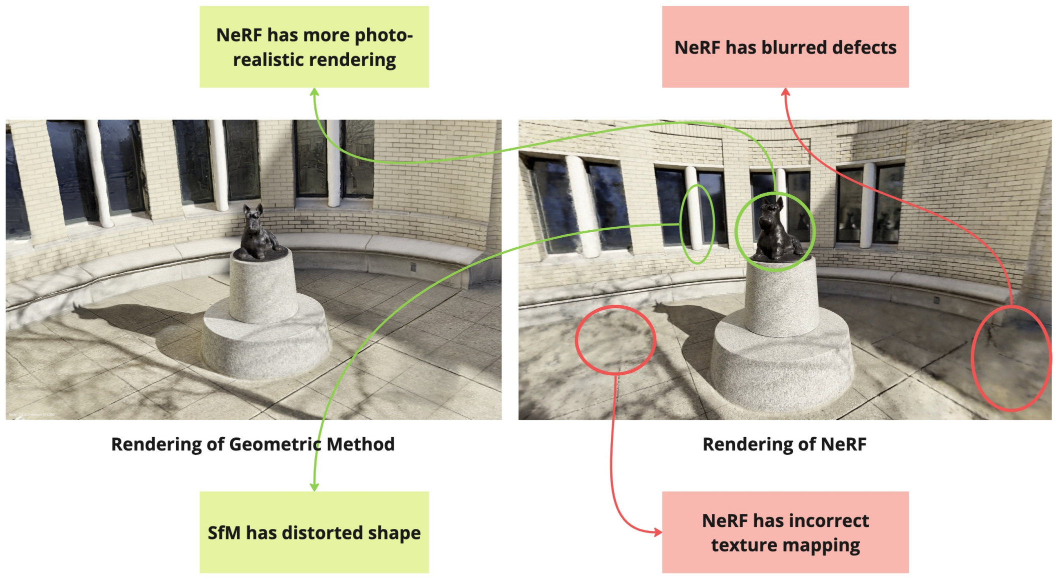
NeRF has more photo-realistic rendering
NeRF has blurred defects
Rendering of Geometric Method
Rendering of NeRF
SfM has distorted shape
NeRF has incorrect texture mapping

Figure 5.5: Comparison of RGB rendering generated from SfM and NeRF.

(Image by the author)

In the graph, the green annotations show the advantages of NeRF compared to SfM, the lighting, and illumination of the Scotty Statue from the NeRF are more photo-realistic, and the pillar from the SfM is distorted because it's a mesh-based solution, the optimization of the shape can inevitably distort the shape, whereas NeRF stores the RGB and density value directly. And the red annotations show the drawbacks of the NeRF, as mentioned above, NeRF stores the RGB and density directly, and such value can have a bias if the input has certain occlusion from certain views, or it is a result of interpolation between frames where the lighting condition changes, for example, the shadow of the tree might change because of the sun movement or wind, whereas the geometric method unwraps the texture on the mesh, therefore it gets less impact.

On the other hand, I compare the generated mesh of two methods, as shown in Figure 5.6. Note that both meshes are open in Blender software without any mesh-level or shape-level post-processing(They are what are generated from each method). From the overview, the meshes are nearly close, and in detail, the mesh from SfM has smoother surfaces. However, such a technique of smoothing the surface can be used as a post-processing of the mesh output of NeRF to achieve the same effect. Therefore, I can think the mesh from the NeRF has the same level of detail and qualities as the one from SfM.

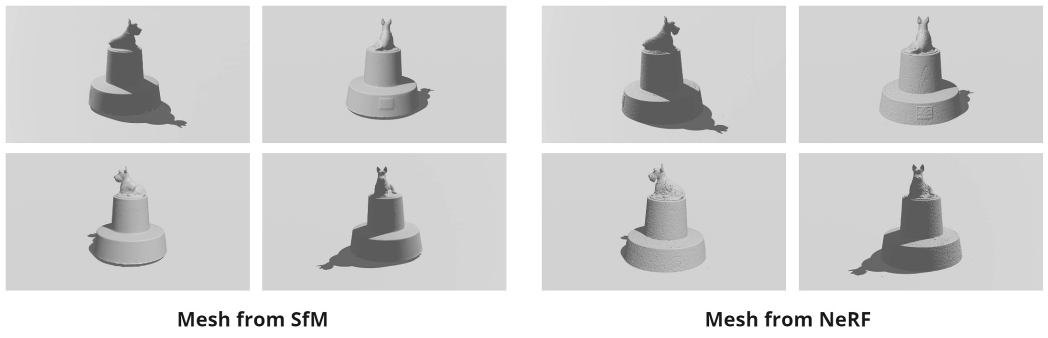

Figure 5.6: Comparison of mesh generated from SfM and NeRF.
(Image by the author)

From the perspective of computing efficiency, I record the related data for the three experiments mentioned above, as shown in Table 5.1. The column refers to the number of images for training, the time cost for training the resolution of the output respectively.

The above two comparisons and the table prove that NeRF can nearly finish the same task that SfM can do, and even do better in the aspects of background, photo-realistic

	Number of Images	Time Cost	Resolution of Output
SfM	561	20 hrs	1920x1080
NeRF-W	50	14 hrs	640x480
Instant-ngp	561	5 mins	1920x1080

Table 5.1: Computation efficiency of SfM and NeRF.

rendering, and computing efficiency. Besides, the NeRF allows us to manipulate the stored 3D information for the next section of the experiment with style transfer.

5.2 Style Transfer Experiment

Now I come to the second part of the experiment. I first experiment with 2D style transfer and then combine 2D style transfer with NeRF with 4 separate experiments.

5.2.1 2D Style Transfer

The 2D style transfer is implemented with VGG19 as a feature extractor and LBFG-S as an optimizer, the weights of style, and content is a hyper-parameter that can be tuned, and I used content weight as 0.015, style weight as 1. Figure 5.7 shows the input and output of the 2D style transfer. On the left, two example images are Snow at Argenteuil by Claude Monet and The Starry Night by Vincent van Gogh, and the content image of Scotty Statue; the right is the generated style images.

5.2.2 3D Style Transfer

Experiment 1

The experiment extends to 3D style transfer first by directly concatenating the 2D style transfer to the NeRF network. Since the 2D style transferred results do not have consistency among adjacent frames for the correspondences, using ColMap to calculate their poses can lead to drift, I use the same poses calculated from the photo-realistic ones.

Figure 4.11 shows examples of output, and it shows a good style transfer already, however, because of the randomized style, such work adds more noise and blur to the rendering.

Experiment 2

Based on the issue of noise and floaters observed in the first experiment, I think about methods to improve the consistency of styles among adjacent frames. The second experiment focuses on adding constraints among the frames and tries to

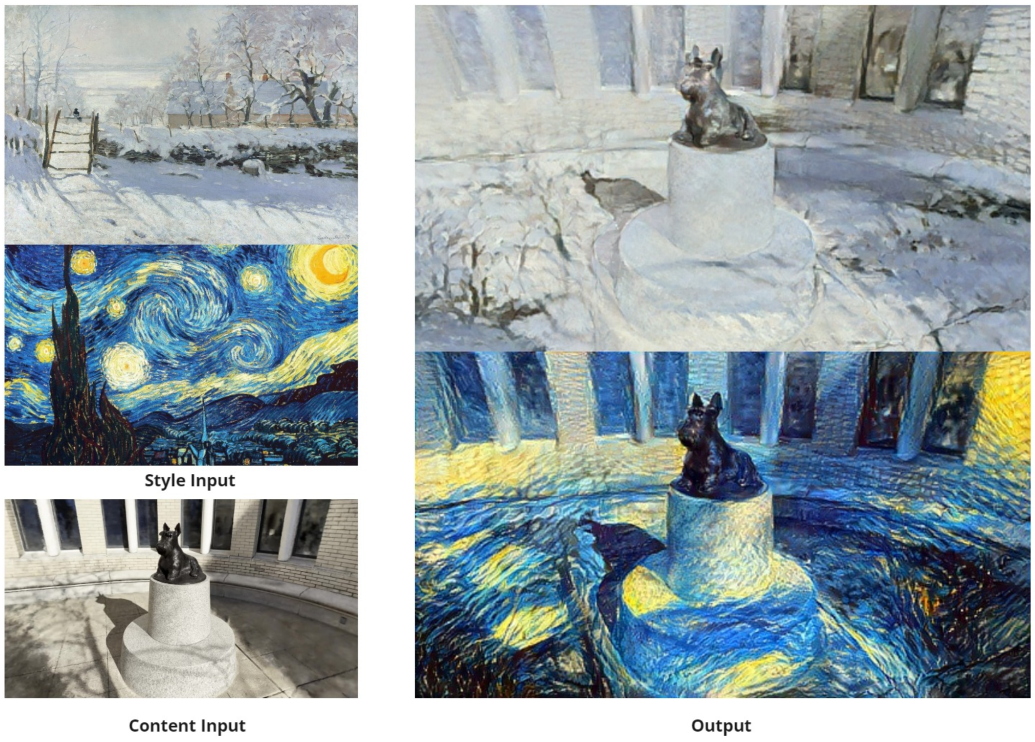

Figure 5.7: Input and output of 2D style transfer.

(Image by the author)

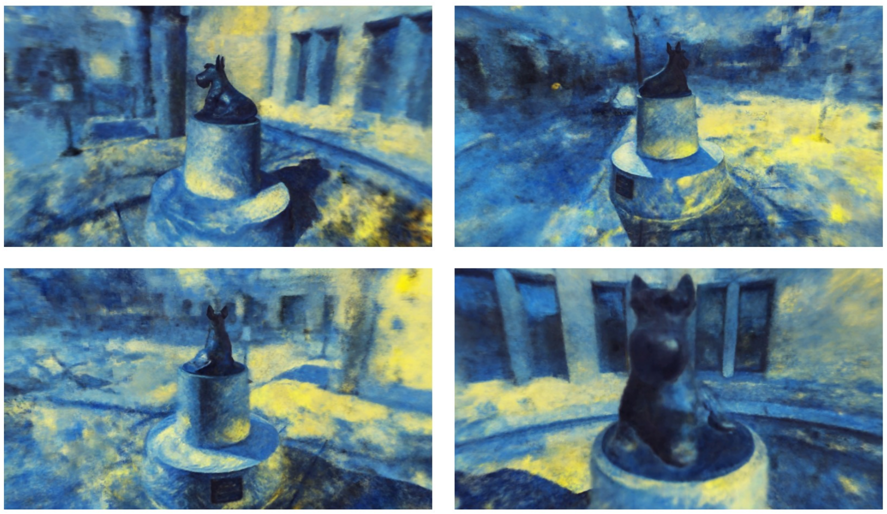

Figure 5.8: Output of method that concatenates output of style transfer with NeRF.

(Image by the author)

maintain the style pattern for the same part of the object. Video style transfer has the advantage of maintaining the pattern because it has the loss function for a sequence of frames. I use EbSynth to select keyframes as style references and propagate the style for the rest of the frames.

I compare the result from style transfer and video style transfer in Figure 5.9. The green box in the graph shows the positive sides of video style transfer, it maintains the style pattern on the Scotty Statue, whereas style transfer is just the randomized patterns. The red box shows the drawbacks of video style transfer, since it tries to maintain the style pattern, when the camera rotates at a sharp angle, much of the information in the background is lost, and the algorithm fills the style pattern in an interpolation way so that it becomes flat strips. And this becomes a severe issue for NeRF to reconstruct the background.

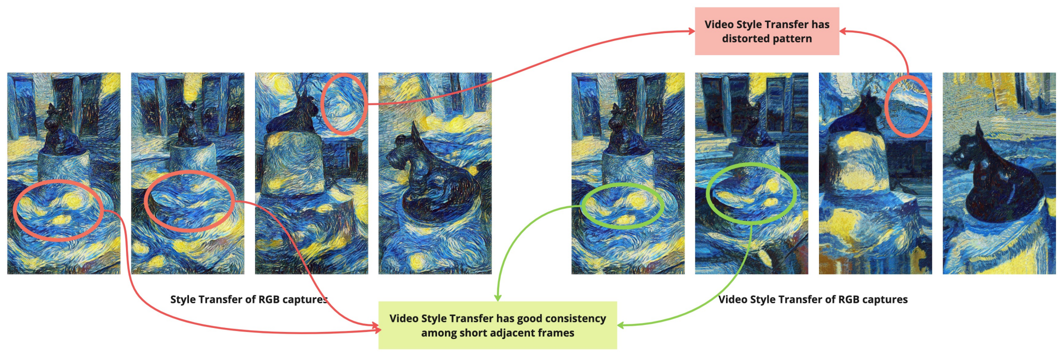
Video Style Transfer has distorted pattern
Style Transfer of RGB captures
Video Style Transfer has good consistency among short adjacent frames
Video Style Transfer of RGB captures

Figure 5.9: Comparison of input processing of 2D style transfer and video style transfer.

(Image by the author)

The output of combining video style transfer and NeRF is shown in Figure 4.12. And I can tell that the front Scotty statue has a good style from different perspectives, and less noise and floaters are there compared to the first experiment. For the background, the near wall is also well maintained, but the distant objects such as trees and buildings from the right top image, lose their content shape.

The video-style transfer is more viable for cases with less rotation but more translation and is good for spreading the style for the front objects. For the 360-degree dataset, it fails in maintaining the distant background.

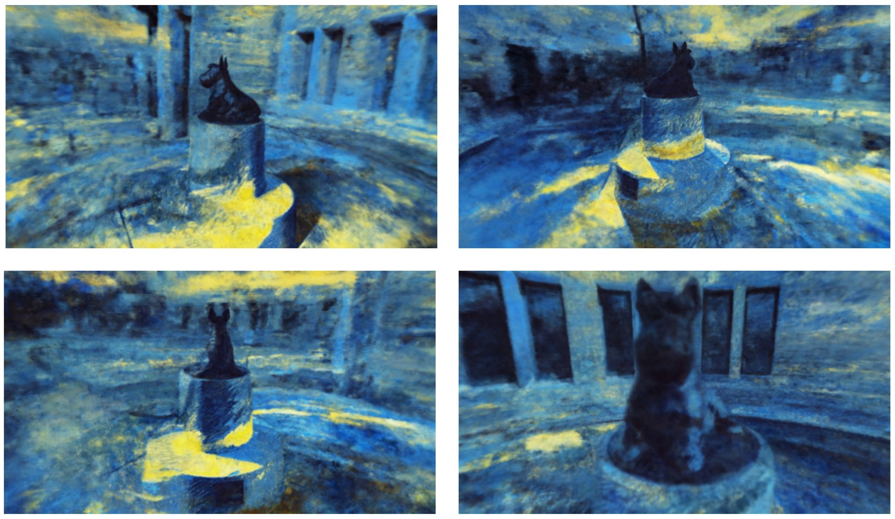

Figure 5.10: Output of method that concatenates output of video style transfer with NeRF.

(Image by the author)

Experiment 3

The third experiment is inspired by experiment 1, I just flip the concatenation of style transfer and NeRF, using the output of NeRF as input of style transfer. And I get the result as shown in Figure 4.13. The style is delicate, and it is recognizable to tell what style image is used for training, but if you watch closely, the pattern is always applied in the same position in terms of the image, it is always right side has more yellow stars and the left side has more blue sky. It barely maintains any consistency. This leads to a jumpiness for video visualization, however, as an art generation work, this type of defect can also mean a unique expression.

Experiment 4

The fourth experiment refers to the paper ARF [22], it is similar to the idea of 2D style transfer, to extract the content feature and style feature and embed them in latent space, but this time the content feature is extracted from the photo-realistic 3D radiance field.

Therefore, I first train the photo-realistic scene with the same Scotty dataset and then train the style transfer with style images together. I changed the hyper-parameters of resolution to $512 * 512 * 512$ and the number of background layers to 32 to fit in the 16GB GPU memory limits. And the output is shown in Figure 4.14.

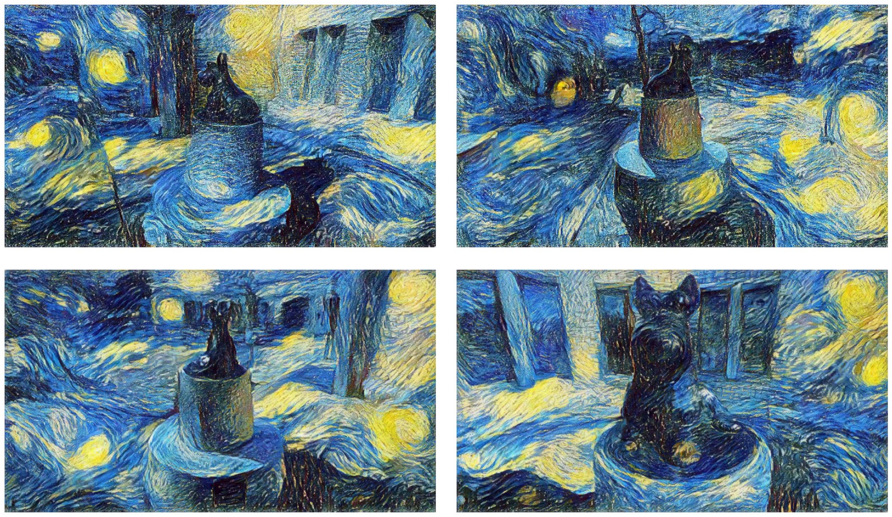

Figure 5.11: Output of method that concatenates output of NeRF with style transfer.

(Image by the author)

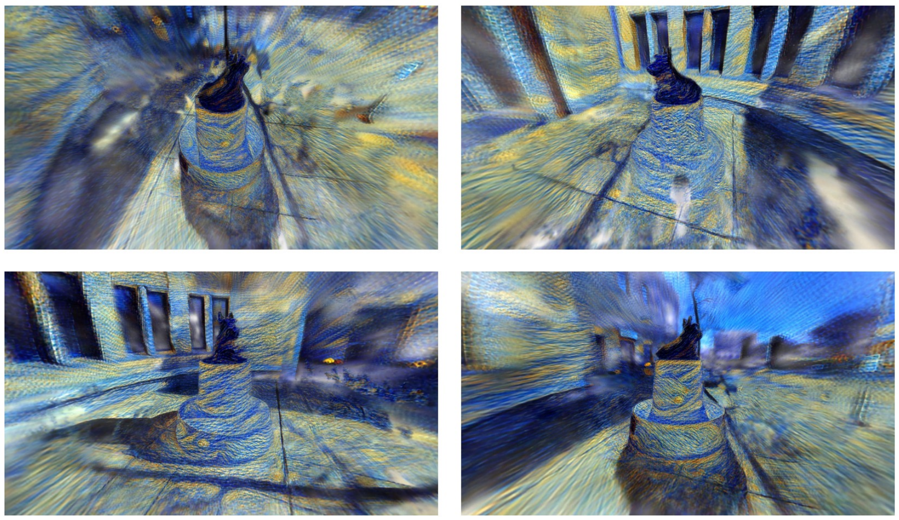

Figure 5.12: Output of method that embeds style feature with radiance field.

(Image by the author)

I can tell from the graph, the pattern has a good consistency among frames. However, it shows different types of noises in the image of the 360-degree dataset, the background shows a "zoom in" style blur, there are empty holes in the top right image, and most important of all, the pattern of style is almost lost, that the starry sky barely can be told from it.

To further explore this method, I experiment with different style images, and get the output in Figure 5.13. I can tell that for the last row, the snowy style is turned into strip lines on the Scotty statue and almost lose all of the semantic styles. And I can tell the same problem from other rows, it is just because others have a more distinguishable pattern that makes the output seem reasonable. And because this NeRF backbone is not a neural network, and is less optimized than Instant-ngp, it is also a hard engineering topic to achieve the same efficiency, needless to say it requires two pieces of training.

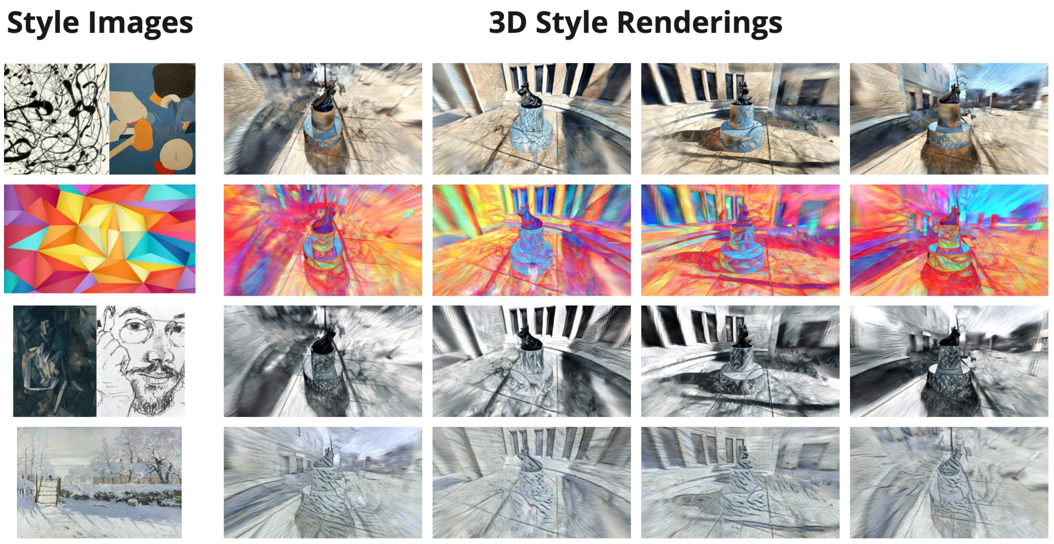

Figure 5.13: Demos of styled 3D scene.
(Image by the author)

So far, I have experimented with different methods and compared the results, and Table 5.2 of computation efficiency can further compare the 3D style transfer methods. The output is ∗19201080 resolution, 30 seconds with 30 fps. The ARF network is slow in radiance field training but fast in style transfer, The rest experiments are very fast in training the radiance field, but slow in style transfer except for video style transfer in experiment 2. And the rendering takes almost a similar time.

Denoise

Considering the output, compatibility, and efficiency, experiment one has the relatively best performance. Therefore, I added a denoise method to utilize the depth information, and decrease the RGB and density for the area between the camera and the object, since most of the floaters from this experiment exist in the air. And as

Time Cost	NeRF	Style Transfer	Rendering
Experiment 1	5mins	6hrs	40mins
Experiment 2	5mins	10mins	40mins
Experiment 3	5mins	6hrs	40mins
Experiment 4	3hrs	3hrs	1hrs

Table 5.2: Computation efficiency of four 3D style transfer experiments.

Figure 5.14 shows, the left image is the output from experiment 1, and the right image is the output of denoise method. I can tell the floaters get removed to some extent and the background and front objects become more clear.

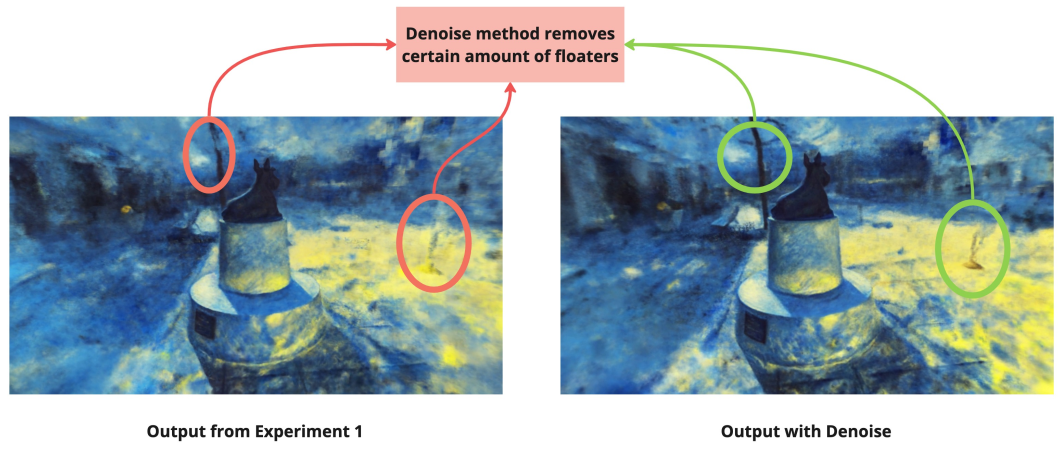

Figure 5.14: Denoise comparison. Red circles represent the noise from experiment 1(left). Green circles represent the effects of denoise(right).

(Image by the author)

Styled Mesh

I also generate the 3D mesh from styled 3D scenes to demonstrate the 3D style transfer further, as shown in Figure 5.15. The left four mesh images are generated from the photo-realistic scene in NeRF, and the right four mesh images are from the denoised method. I can tell that the mesh manages to maintain its shape and style at the same time, although it is less smooth than the left one, yet a smooth surface technique can be applied as post-processing to relax this issue. This step proves the experiment of style transfer is three dimensions since the previous demos are only 2D rendered images.

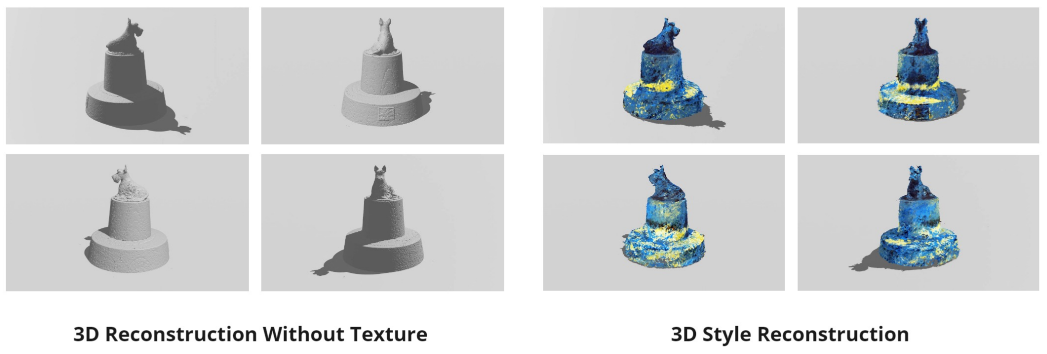

Figure 5.15: Comparison of mesh generated from NeRF and styled NeRF.
(Image by the author)

5.3 Artistic Pieces

I use the method(Experiment 1 + Denoise) explored above to create some art pieces as a demonstration of the practical use of this thesis research. As shown in Figure 5.16, the first two rows use style weight as 1.0 and content weight as 0.015 to keep more style for the output. The last two rows use style weight as 0.7 and content weight as 0.1 to keep more content for the output. This practice shows that one can tune the strength of the 3D-style scene based on their goals, for example, if one wants to create strong art styles, one can increase the style weight; if one wants to slightly change the color theme, one can increase the content weight.

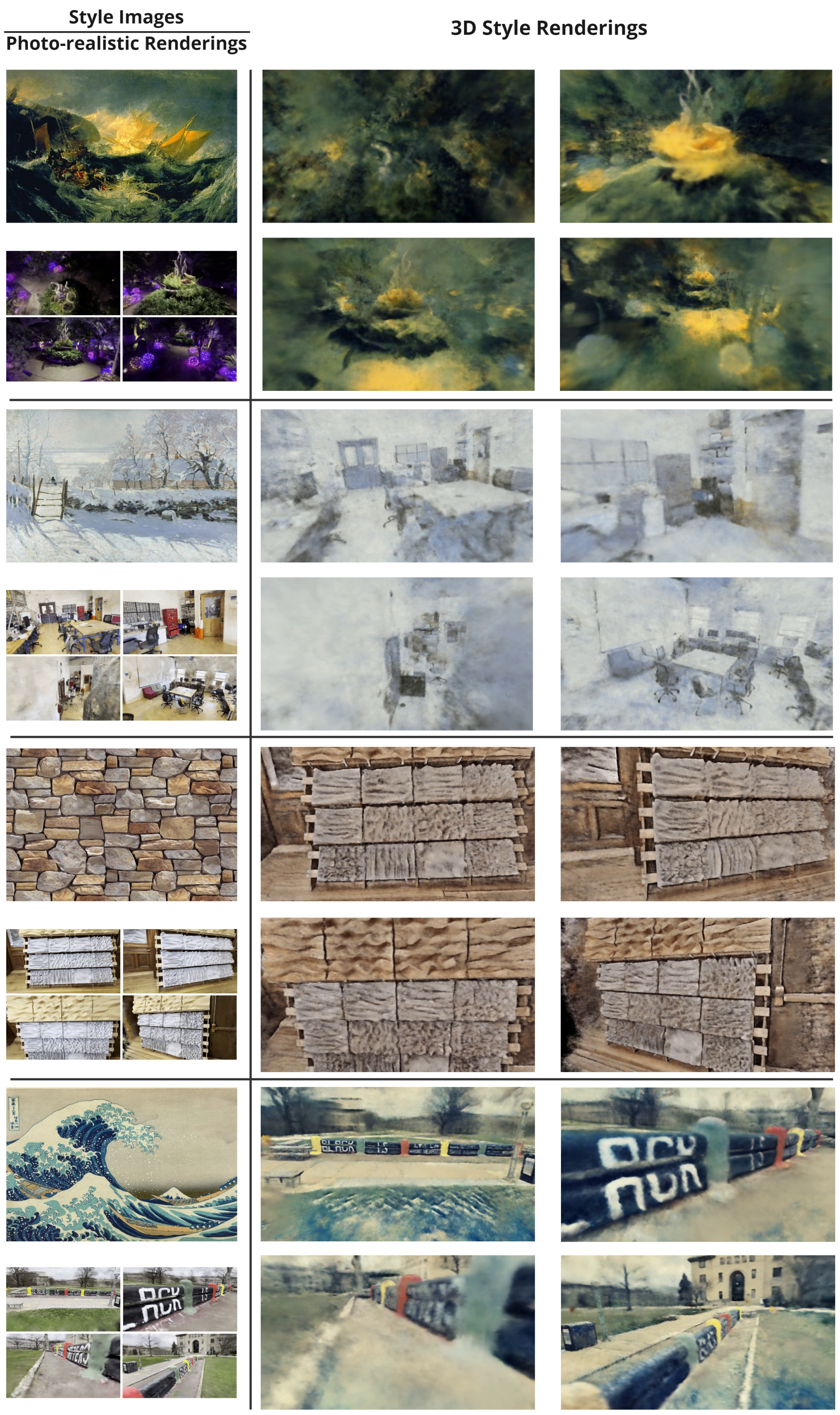

Figure 5.16: Demos of artistic pieces. The left side is style images, the first row of the right side is photo-realistic rendering, and the second row is styled rendering.

(Image by the author)

5.4 Applications

Based on the experiments and results mentioned above, this thesis proposes two applications that further demonstrate the uses of 3D style transfer.

3D Style VR

First, I propose the VR application with 3D style transfer, the demonstration graph is shown in Figure 5.17. The idea is to put the 3D-styled scene into the VR environment, and the users can navigate through the scene with controllers. Controllers support the function of zoom in/out, rotation, and moving. Another ideal function would be a switcher to pick different styles or show the photo-realistic scene, however, this function requires more engineering work that this thesis can not implement with limited time.

Given the above functions, any user can have an immersive experience by exploring the 3D scene. Designers and architects can change the style to experiment with different textures or materials for the same object, the navigation function also allow them to explore the entire scene freely to understand the geometric structure for reference and design purposes. On the other hand, artists can use it as a tool to design a creative VR experience or a 3D artistic scene.

3D Toy Blocks

The second proposed application is toy blocks, as shown in Figure 5.18. The idea of VR is to change the whole scene and have free manipulation of the camera view, and the idea of toy blocks is to manipulate the individual objects and combine multiple objects or scenes together.

The challenge in this application is how to combine multiple NeRF scenes together, here I take assumptions that all of the candidate NeRF scenes are trained with the same NeRF network, meaning their weights have the same network architecture. Currently, there are no promising solutions to combine multiple scenes together, although there is work like Block-NeRF from Waymo [17], which mainly focuses on decomposing a large-scale scene into multiple blocks and training each individual block separately, it does not work well to blending multiple scenes together. To do so, I first need to set up a unified scale and world coordinate standard for each scene. Since the mono-RGB input does not generate scale information, the scale parameter can also be tuned by users manually. And the second step is to set up the bounding box of the interested region, for example, I only want to extract the Scotty statue from the entire scene, embed it into the CodeLab scene, and then choose the priority

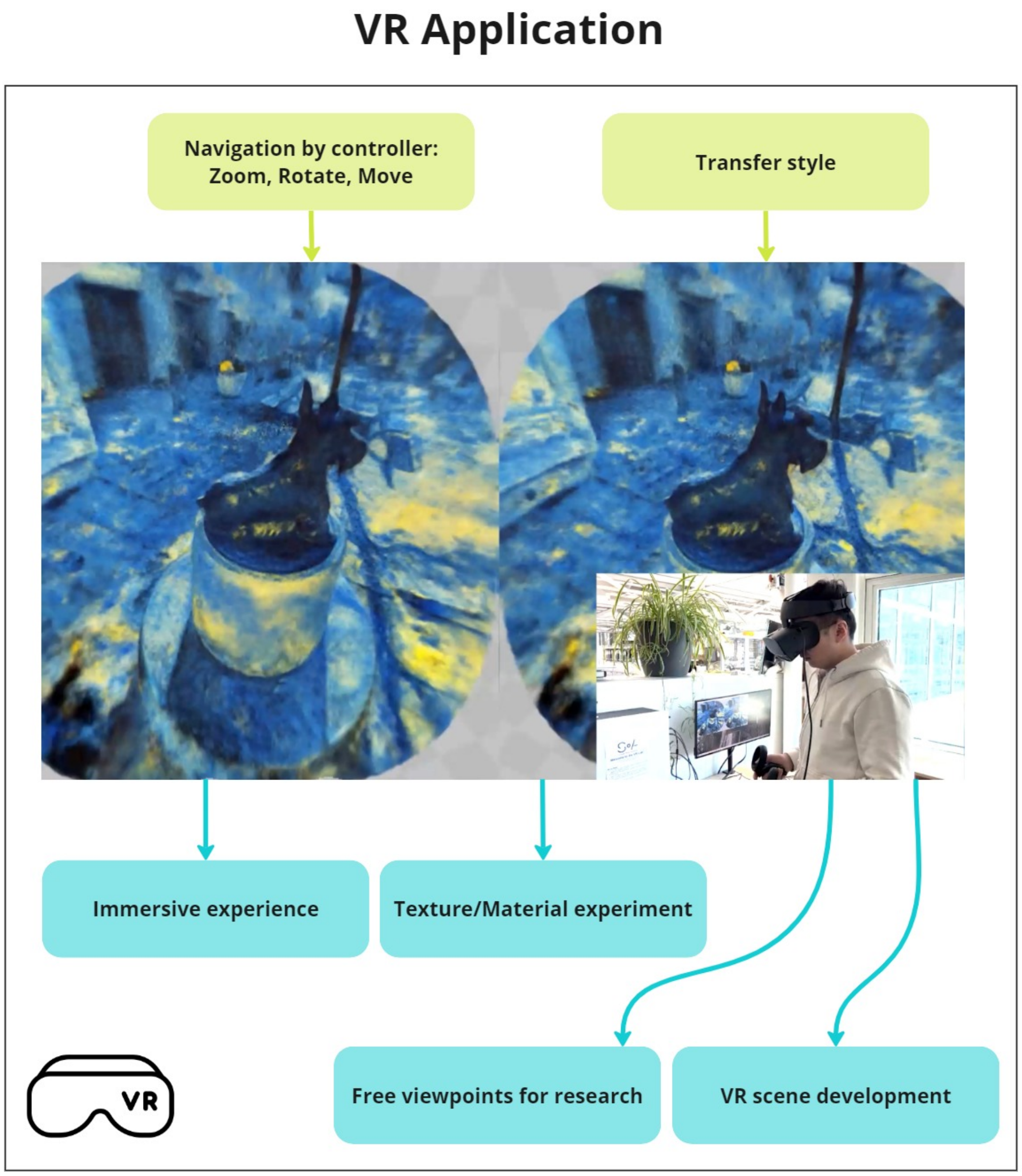
VR Application
Navigation by controller: Zoom, Rotate, Move
Transfer style
Immersive experience
Texture/Material experiment
Free viewpoints for research
VR scene development
VR

Figure 5.17: VR application diagram.

(Image by the author)

of layers to decide to use one NeRF's weight parameters to replace another, here Scotty statue should have the higher layer. Similarly, I can combine multiple objects into one scene, or merge two different scenes together.

If I can manage those challenges properly, this application can have a good potential to allow us to scan and generate digital twins of daily objects, changing their styles, and placing them freely in the digital environment. Architects can bring any objects into their 3D scene design without searching for if such objects have a 3D model or manually simulate one, but can just simply use their phones. And designers can now build art upon real objects easily, and create art scenes more efficiently.

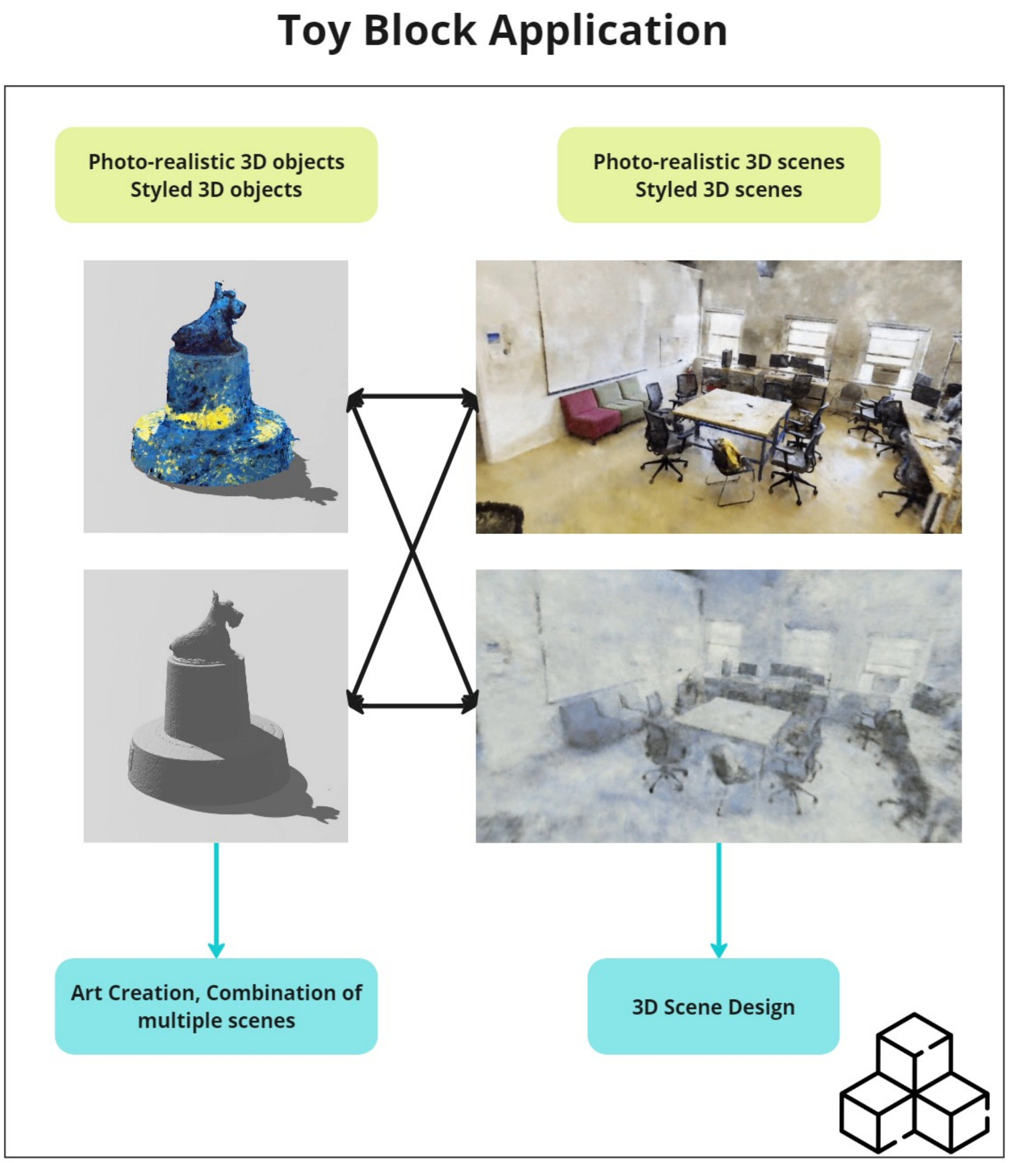
Toy Block Application
Photo-realistic 3D objects
Styled 3D objects
Photo-realistic 3D scenes
Styled 3D scenes
Art Creation, Combination of multiple scenes
3D Scene Design

Figure 5.18: Toy block application diagram.
(Image by the author)

5.5 Discussion

In this chapter, I present 4 experiments of 3D style transfer and each of them shows a different way of combining 2D style transfer and NeRF, including concatenation of style transfer and NeRF head to tail and vice versa, video style transfer to add constraints to the consistency among adjacent frames, and embedding style feature into radiance field in latent space.

Given the same dataset, the output is compared by the quality of style and content distribution, and also the time consumption for the data pre-processing, training, and rendering. It shows that the video consistency method has the degeneration of styles for big rotations since there are rare overlaps for the algorithm to deduct, and the embedded solution requires a large amount of GPU and has the "zoom-in" defects for a 360-degree scene. Besides the above reasons, I further evaluate the results from both perspectives of subjective art sense and engineering difficulties and choose experiment 1 for further development. The denoise processing is used to improve the quality of output.

Different outputs are presented from the experiments, and I also use them to produce some art pieces as a demonstration of the results of this research. It finds an interesting and valid way to make use of NeRF and 2D style transfer and deliver the idea of 3D style transfer. As the community of NeRF grows rapidly, and there are already some great publications focusing on the same direction as this research, I believe it still shows its contribution to the community.

In the end, this thesis discusses two proposed applications of immersive VR and toy blocks to show that such 3D style transfer can be a good potential to assist architects, designers, and artists in their creative work, and for non-professionals to have an immersive experience. Immersive VR is achieved at the level of free navigation with controllers, and it already receives positive feedback from user tests. There are proposed functions of real-time style switchers in the VR application. in the toy blocks application, the functions of NeRF scene merging require more research and engineering that goes beyond the scope of this thesis. However, I believe this application idea will become possible in the near future.

Chapter 6

CONCLUSION

6.1 Achievements

As proposed at the beginning of this thesis, this thesis aims at exploring a lightweight and easy-to-implement solution for 3D style transfer, by conducting various experiments, it manages to build up a pipeline to train a neural network containing the 3D styled scene, and it allows anyone to use their smartphone as a data caption tool, to generate their own styled scenes. Users don't need to address the issue with code or the architecture of the network, instead just input images for 3D scenes and the style respectively.

On the other hand, it aims at combining neural radiance field with 2D style transfer to achieve 3D style transfer to achieve a relatively fast process. The experiment shows that the time consumed in 3D reconstruction and style transfer is faster than traditional solutions of the sum of each, although there is an extra great amount of time spent in the data pre-processing sections, which can be potentially optimized with enough engineering solutions. From the aspect of the quality of the results, although there are outstanding papers got published during this thesis research, it still provides value in comparing and quantifying different solutions from both analytic and aesthetic perspectives, and the optimization methods are valid to be applied for different NeRF-related works.

Last but not least, it proposes potential applications that can be further developed into an application or a product. And it suggests its potential for non-professional designers and professionals to take it as an assistive tool.

6.2 Limitations

As a new and upsoaring technique, there are many limitations in terms of research and engineering, some might be solved soon, and some are still worth exploring:

- Incompatibility between different NeRF networks. Since most NeRF is neural networks, almost every NeRF has a different architecture, meaning the radiance field is stored in a different format. The transformation of weights from one network to another is as difficult as creating a new architecture. If there is a method that can solve this issue, NeRF community will expand and unite in a better way.

- GPU memory. NeRF is a type of neural network that becomes possible based on the development of modern GPU. Therefore, implementing algorithms on top of the NeRF requires even more GPU. A very frequent problem during the training is GPU out of memory.
- Performance. Although state-of-arts Instant-ngp has greatly improved the performance of NeRF training, another part of the algorithm has not, such as ColMap, style transfer. Such limitations make it hard to become an end-to-end solution.
- Noise. Although some companies such as LumaAI have shown a solution to remove the noise to their best, such a solution is not open-sourced. And it doesn't always promise a clean reconstruction. It's part of the features in NeRF work, and it brings difficulties for the advanced algorithms built on top of it.

6.3 Further Work

This type of work has a huge potential for various applications from my personal view. During the research and experiment, I believe the below applications are possible with enough engineering effort.

- Integrating the 3D style transfer network such as ARF into Instant-ngp. Instant-ngp provides multiple handy features for both research and application, besides it has a promising quality of trained radiance field. The integration can make the 3D style transfer one step further to an application.
- Further developing the denoise feature. As mentioned in the limitations section, noise is a unique feature of NeRF that distinguishes it from other 3D reconstruction methods. The method of denoising used in this thesis is rather simple, remove the RGB and density in low-confident areas that are between the camera and objects predicted by the depth. But it doesn't remove all types of noise, there should be a better solution.
- Developing features to combine multiple radiance fields. As one of the proposed applications in this thesis, the ability to combine multiple scenes and adjust their relative coordinates can make it a very useful tool to assist

architecture, designers, and artists. And thanks to its fast reconstruction speed, this technique can massively change the way I virtualize and interact with the environment.

BIBLIOGRAPHY

[1] *AgiSoft PhotoScan Professional (Version 1.2.6) (Software). (2016*). Retrieved from http://www.agisoft.com/downloads/installer/.*

[2] Yaosen Chen et al. "UPST-NeRF: Universal Photorealistic Style Transfer of Neural Radiance Fields for 3D Scene". In: *arxiv*. 2022.

[3] Pei-Ze Chiang et al. *Stylizing 3D Scene via Implicit Representation and HyperNetwork*. 2021. arXiv: `2105.13016 [cs.CV]`.

[4] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. "A Neural Algorithm of Artistic Style". In: *CoRR* abs/1508.06576 (2015). arXiv: `1508.06576`. URL: `http://arxiv.org/abs/1508.06576`.

[5] Ondrej Jamriska. *Ebsynth: Fast Example-based Image Synthesis and Style Transfer*. `https://github.com/jamriska/ebsynth`. 2018.

[6] Ondřej Jamriška et al. "Stylizing Video by Example". In: *ACM Transactions on Graphics* 38.4 (2019).

[7] Cheng-Bin Jin. *Real-Time Style Transfer*. `https://github.com/ChengBinJin/Real-time-style-transfer/`. commit xxxxxxx. 2018.

[8] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. "Perceptual Losses for Real-Time Style Transfer and Super-Resolution". In: *CoRR* abs/1603.08155 (2016). arXiv: `1603.08155`. URL: `http://arxiv.org/abs/1603.08155`.

[9] Ricardo Martin-Brualla et al. "NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections". In: *CVPR*. 2021.

[10] Ben Mildenhall et al. "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis". In: *ECCV*. 2020.

[11] Pierre Moulon, Pascal Monasse, and Renaud Marlet. "Global fusion of relative motions for robust, accurate and scalable structure from motion". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013, pp. 3248–3255.

[12] Thomas Müller et al. "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding". In: *ACM Trans. Graph.* 41.4 (July 2022), 102:1–102:15. DOI: `10.1145/3528223.3530127`. URL: `https://doi.org/10.1145/3528223.3530127`.

[13] M. Ruder, A. Dosovitskiy, and T. Brox. "Artistic style transfer for videos and spherical images". In: *International Journal of Computer Vision* (2018). online first. URL: `http://lmb.informatik.uni-freiburg.de/Publications/2018/RDB18`.

[14] Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. “Artistic Style Transfer for Videos”. In: *German Conference on Pattern Recognition*. 2016, pp. 26–36.

[15] Johannes Lutz Schönberger and Jan-Michael Frahm. “Structure-from-Motion Revisited”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[16] Johannes Lutz Schönberger et al. “Pixelwise View Selection for Unstructured Multi-View Stereo”. In: *European Conference on Computer Vision (ECCV)*. 2016.

[17] Matthew Tancik et al. “Block-NeRF: Scalable Large Scene Neural View Synthesis”. In: *arXiv* (2022).

[18] Qianqian Wang et al. “IBRNet: Learning Multi-View Image-Based Rendering”. In: *CoRR* abs/2102.13090 (2021). arXiv: `2102.13090`. URL: `https://arxiv.org/abs/2102.13090`.

[19] Changchang Wu. “Towards Linear-Time Incremental Structure from Motion”. In: *2013 International Conference on 3D Vision - 3DV 2013*. 2013, pp. 127–134. DOI: `10.1109/3DV.2013.25`.

[20] Changchang Wu et al. “Multicore bundle adjustment”. In: *CVPR 2011*. 2011, pp. 3057–3064. DOI: `10.1109/CVPR.2011.5995552`.

[21] Alex Yu et al. *pixelNeRF: Neural Radiance Fields from One or Few Images*. 2020. arXiv: `2012.02190 [cs.CV]`.

[22] Kai Zhang et al. *ARF: Artistic Radiance Fields*. 2022.

Appendix A

DEVELOPMENT FILES AND CODES

This chapter gives important snippets and command lines as samples, the rest of the code can be referred to GitHub https://github.com/CaoYuchen/NeRF3DStyle.

Command Lines

Below are the major command lines to use for 2D style transfer, Instant-ngp, and 3D ARF code, respectively.

```
# Batch processing style transfer
for i in $(ls data/scotty/images); do style_transfer data/scotty/
    images/${i} data/styles/14.jpg -o data/scotty_output1/${i} --
    devices cuda:0 -s 1920 -sw 0.5; done
```

Listing A.1: Style transfer command lines

```
# Generate dataset from video, 1 minute uses --video_fps 2 to get
    50-150 frames.
python /scripts/colmap2nerf.py --video_in <filename of video> --
    video_fps 2 --run_colmap --aabb_scale 16 --out <>

# Training to generate NeRF with Instant-ngp
./build/testbed --mode nerf --scene [path to training data folder
    containing transforms.json]

# Rendering mp4 video from the scene
python scripts/run.py --mode nerf --scene data/nerf/fox --
    load_snapshot data/nerf/fox/base.msgpack --video_camera_path
    data/nerf/fox/base_cam.json --video_n_seconds 5 --video_fps 60
     --width 1920 --height 1080
```

Listing A.2: Instant-ngp command lines

```
# train photo-realistic and style NeRF together:
./launch.sh <exp_name> <GPU_id> <data_dir> -c configs/custom.json
# Colmap for customized dataset:
bash proc_colmap.sh <img_dir>
```

Listing A.3: ARF command lines

Snippet Codes

Below are the major codes of the loss function of style transfer, and the hyperparameters for tuning the ARF.

```python
# Content loss function
class ContentLoss(nn.Module):
    def __init__(self, target, eps=1e-8):
        super().__init__()
        self.register_buffer('target', target)
        self.loss = ScaledMSELoss(eps=eps)

    def forward(self, input):
        return self.loss(input, self.target)
# Stlye loss function
class StyleLoss(nn.Module):
    def __init__(self, target, eps=1e-8):
        super().__init__()
        self.register_buffer('target', target)
        self.loss = ScaledMSELoss(eps=eps)

    @staticmethod
    def get_target(target):
        mat = target.flatten(-2)
        # The Gram matrix normalization differs from Gatys et al.
    (2015) and Johnson et al.
        return mat @ mat.transpose(-2, -1) / mat.shape[-1]

    def forward(self, input):
        return self.loss(self.get_target(input), self.target)
# Sum loss function
class SumLoss(nn.ModuleList):
    def __init__(self, losses, verbose=False):
        super().__init__(losses)
        self.verbose = verbose

    def forward(self, *args, **kwargs):
        losses = [loss(*args, **kwargs) for loss in self]
        if self.verbose:
            for i, loss in enumerate(losses):
                print(f'({i}): {loss.item():g}')
        return sum(loss.to(losses[-1].device) for loss in losses)
```

Listing A.4: Style transfer loss functions

(crowsonkb. (n.d.). GitHub - Crowsonkb/style-transfer-pytorch: Neural style transfer in PyTorch.)

```
{
    "reso": "[[128, 128, 128], [256, 256, 256], [512, 512, 512]]",
    "n_iters": 102400,
    "background_nlayers": 32,
    "background_reso": 1024,
    "cam_scale_factor": 0.9,
    "upsamp_every": 25600,
    "near_clip": 0.35,
    "lr_sigma": 3e1,
    "lr_sh": 1e-2,
    "lr_sigma_delay_steps": 0,
    "lr_fg_begin_step": 1000,
    "thresh_type": "weight",
    "weight_thresh": 1.28,
    "lambda_tv": 5e-3,
    "lambda_tv_sh": 5e-3,
    "lambda_tv_background_sigma": 5e-3,
    "lambda_tv_background_color": 5e-3,
    "lambda_beta": 1e-5,
    "lambda_sparsity": 1e-11,
    "background_brightness": 0.5,
    "tv_early_only": 0,
    "tv_decay": 0.5
}
```

Listing A.5: ARF hyperparameters

(Zhang, K., et al. (2022). ARF: Artistic radiance fields. In Lecture Notes in Computer Science (pp. 717–733). Springer Nature Switzerland. http://dx.doi.org/10.1007/978-3-031-19821-2_41)

Appendix B

OBSERVATIONS OF APPLICATION IN SOCIAL CONTEXT

This thesis proposes three applications of the research topic 3D style transfer. Because I only bring one of three proposals to the practice, I can observe directly for the practical one and indirectly for the rest two by describing the applications to others and gaining feedback.

For the VR application, users generally take a short time to get used to how to navigate through the scene by the controller without external hints. The positive feedback is that they feel immersive in the scene, can tell what style images are applied in the 3D scene, and appraise the visual effect of the scene. A couple of users are from art and architecture backgrounds that claim it is helpful to observe the scene from different perspectives and they can imagine it being used in immersive design cases. Due to the limited boundary of the scene, I observe it happens frequently that users go outside of the scene and take a while to find the way back because the region outside of interest consists of floating noise that gives little information about the direction.

For the toy block application, architecture students feel it is a good idea if they can combine multiple scenes quickly from daily objects into 3D software, such as Rhino, and Unity. They recommend turning this technique into an API that can directly be called from 3D software to bind processes from capture to generation in one click. And if it can work well, it can help them improve their efficiency.

For the 3D print-styled mesh application, because 3D color printing is quite expensive and time-consuming, I haven't found an easy way to test the demo. But some experienced architecture students observe the generated 3D mesh from this thesis, and predict that the printed model would be rough and might lead to defects because the 3D printer can't incarnate the complex surface very well, but if given some post-processings to smooth out the surface and reduce the polygons, it can be as close to a high-fidelity one.