

國立臺灣大學電資學院資訊工程學研究所

碩士論文

Computer Science and Information Engineering

College of Engineering

National Taiwan University

Master Thesis

利用深度學習之影片深度估測建構 3D 多視角智慧風格
轉換

3D Novel View Style Transfer via Learning-based Video
Depth Estimation

張凱程

Kai-Cheng Chang

指導教授: 洪一平博士, 陳祝嵩 博士

Advisor: Yi-Ping Hung Ph.D., Chu-Song Chen Ph.D.

中華民國 111 年 9 月

September, 2022



國立臺灣大學碩士學位論文
口試委員會審定書

利用深度學習之影片深度估測建構 3D 多視角智慧風格
轉換

3D Novel View Style Transfer via Learning-based Video
Depth Estimation

本論文係張凱程君（學號 R09922025）在國立臺灣大學資訊工程
學系完成之碩士學位論文，於民國 111 年 8 月 23 日承下列考試委
員審查通過及口試及格，特此證明

口試委員：

洪一平

陳祝嵩

（指導教授）

傅立成

系主任

洪士灝






Acknowledgements

這一次碩論主題主要來自於科技部的前瞻計畫中「AI 智慧風格轉換於顯示系統」的子計畫，在這邊先感謝科技部提供資金給我們，讓我們有足夠資金應用於這個主題。由於這個子計畫牽扯到的領域包含三維空間的風格轉換，目的是希望使用者可以有 3D 藝術場景的體驗，若期望的場景可以透過 AI 進行風格轉換，可以大幅減少藝術設計的人力成本，也同時可以讓顯示設備有更多的功能，對未來 VR 或 AR 的發展更具前瞻性。而三維空間的風格轉換是近期才興起的，故該領域有很大的研究空間。

技術的部分，我要先感謝陳祝嵩教授以及洪一平教授，在 3D 視覺以及影像風格轉換等等領域中提供充足的知識，包括深度學習與 3D 重建的部分，教授給予我很多可以著手進行的點，使我在決定論文主題時有明確的方向，以及也謝謝教授很有耐心的隨時提示實驗進行上一些可以改進的地方，讓我受益良多，論文也要感謝教授進行指導，不僅協助我整理參考資料，也幫我在方法與實驗的敘述上給予寶貴的意見，我的表達能力也因此比以前的我進步很多。最後我要感謝口試委員：洪一平教授、陳祝嵩教授、以及傅立成教授，在口試的最後對我的成果表示肯定，也跟我說我這個主題可以應用的場合，讓我的成果更能發揮出它的價值。

再來就是感謝實驗室的學長以及跟我一起合作進行智慧風格轉換實驗的同



學，首先我要感謝曜至學長，我的論文主題中有使用到他提供的工具，而且曜至學長在研究上有提供他如何面對困難的經歷，讓我有很多地方值得參考。再來主要就是要感謝冠維，也是我主要的合作夥伴，他幫助我找出在風格轉換的研究上所遇到的問題，並且以他的經驗告訴我可以如何解決，讓我更清楚了解怎麼做研究，雖然過程困難重重，不過總算在最後有一個還不錯的成果，這邊非常感謝曜至學長與冠維提供的幫助。

然後就是實驗室的同學們，首先我想感謝竣宇、修瑞，從大學時期就互相幫忙，到了碩士班後也一起修課，一起討論並完成實驗室的計畫，再來是曜福學長以及韋勳，在我壓力大的時候願意跟我一起玩遊戲來舒壓，還有其他同學、學長姐以及學弟妹，他們非常的熱情，與他們的各個計畫合作也進行得很順利，讓我可以不用擔心我受到緊張的氛圍對研究時期的心理造成影響，現在的我過得很愉快，這邊感謝每一位實驗室的人，讓我可以平安度過這兩年的研究生生活，也很慶幸我是這個實驗室的一員。

最後就是感謝我的家人，他們從小就栽培我到現在，不惜一切也要在教育上給我選擇最好的。在我壓力比較大的時候，也是有他們的鼓勵以及安慰，而不是像某些家庭會有家長對兒女要求太高的問題，讓我在這兩年過得很安心。有他們一路上的支持，我才能如願的讀到碩士學位。

因為有大家的支持，才能成就現在的我，希望將來進入職場可以跟我們的實驗室一樣，有著好老闆以及好同事互相勉勵與扶持，為自己的未來一起打拼！



摘要

3D 智慧風格轉換於最近一兩年備受關注，而現有的 3D 風格轉換方法主要是透過預估影像中的場景在三維空間中的位子後再進行 3D 場景的風格轉換，並結合多視角來預估目標視角的成像。然而目前的 3D 風格轉換方法需要依賴耗時比較久的全局最佳化方法來預估深度與相機座標，因此，我們的目標是藉由較有效率的深度學習模型所生成的深度與相機座標，並且結合現有 3D 風格轉換模型來完成多視角影像估測。其中這個改變方法需要面臨的問題是不同時間點所產生的 3D 座標系統會不一致，而使得現有的 3D 風格轉換方法無法產生正常的轉換成果。而我們以建構局部且隨時間增加和刪減的 3D 點雲，加上改良現有的 3D 風格轉換方法，來產生既沒有破壞 3D 場景架構也不會因點雲變換而不停閃爍的風格轉換成果。這個方法最大的好處在於我們的架構不需要依賴特定的 3D 建構方法也能產生與現有方法相似的成果，減少了許多時間成本。我們同時也提供了 VR 體驗讓使用者可以觀賞 3D 風格轉換影片。

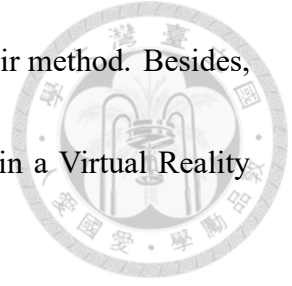
關鍵字：多視角、風格轉換、立體影片、虛擬實境



Abstract

3D style transfer has been investigated for about two years. Recently developed methods either used neural radiance field or point-cloud generation to apply for 3D scene style transfer and they combined novel view synthesis to predict target views. However, these works are faced with time-consuming problems because they need globally-consistent optimization. We aim to speed up by applying the learning-based structure-from-motion(SfM) module to SOTA. The naive combination causes problems due to the inconsistency of 3D coordinates between different views generated by local-optimized SfM-learners. To overcome this issue, we use the sliding-window point cloud set that adds points close to the current view and removes points far away from it to make sure the results in 3D space won't be affected by the 3D difference. Stylizing different point cloud sets may generate flickering results; therefore, we modified the style transfer module we use to deal with the flickering problem. The experiment shows that our reformed method can accomplish comparable visual results to the original style transfer module, while we

can utilize a much more efficient SfM constructor compared with their method. Besides, we implement novel view synthesis applications like stereo videos in a Virtual Reality system for the visual experience.



Keywords: Novel View Synthesis, Style Transfer, Stereoscopic Video, Virtual Reality



Contents

	Page
Verification Letter from the Oral Examination Committee	i
Acknowledgements	iii
摘要	v
Abstract	vi
Contents	viii
List of Figures	x
List of Tables	xiii
Denotation	xiv
Chapter 1 Introduction	1
1.1 Introduction of Style Transfer	1
1.2 Motivations	2
1.3 Contribution	6
Chapter 2 Related Work	7
2.1 Image and Video Stylization	7
2.2 Structure-from-Motion	8
2.3 Novel View Synthesis	9
2.4 3D Scene Stylization	10

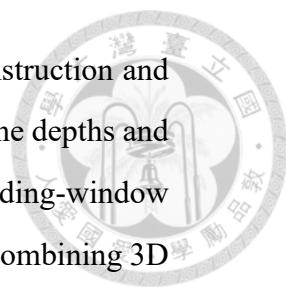
2.5	PointLSTM	12
Chapter 3	Methodology	13
3.1	Problem Definition	13
3.2	3D Point Cloud Construction	14
3.3	3D Scene Stylization	14
3.4	Novel View Rendering	18
3.5	Training	19
3.6	Implementation Details	21
Chapter 4	Experiments	22
4.1	Qualitative Results	22
4.2	Quantitative Results	23
4.3	User Preference	25
4.4	Efficiency	26
4.5	Discussion	27
Chapter 5	Conclusion	29
	References	31





List of Figures

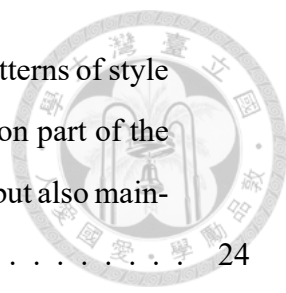
1.1	We introduce 3D novel view style transfer using an end-to-end pipeline, combined with the latest 3D style transfer model and a learning-based 3D constructor. In this work, we aim to address the efficiency issue in recently proposed approaches, while maintaining the stylized quality of novel views.	4
1.2	Temporal inconsistency The main difference between Huang <i>et al.</i> [16] and our work is that the former uses a static point cloud set, while our method uses the sliding-window point cloud set to overcome the ghost effect problem. We show that if we directly stylize the image sequences by Huang’s module, the stylized results may lead to inconsistencies in the red boxes. The main effect is strong and repetitive changes of contrast that causes visual discomfort.	5
2.1	Point feature learning. The diagram explains how PointLSTM learns. For each point p_i^t in current time t , it first finds k nearest points of the previous time (3 in this example). Then, the model learns pairwise virtual states $(\tilde{h}_{i,j}^t, \tilde{c}_{i,j}^t)$ between p_i^t and previous point p_j^{t-1} . Last, it generates the final states using global pooling among all pairwise virtual states.	12



3.1 **System Overview** Our pipeline consists of point cloud construction and stylization. For point cloud construction, we first generate the depths and poses of each frame using Deep3D [22]. Then, we use sliding-window back projection to assign point cloud sets to each frame by combining 3D points of adjacent frames. After that, we transfer point cloud sets to the desired style using the style transformation module. Last, we use differential rendering proposed by [49] to project 3D features to 2D feature maps and decode feature maps into RGB images. 15

3.2 **Stylization module** The transformation module converts the point cloud to a specified style by extracting channel-correlated features. For point cloud, we use PointNet++ to aggregate intra-frame point features since too intensive points may cause messy results such as unwanted shapes or textures. Then, the PointLSTM layer aims to learn inter-frame point relations and it can reduce the flickering problem. For style images, only CNN is needed to learn compressed features. Both routes are designed to reduce computational space and get refined representations of features. In particular, f'_t and \mathbf{F}'_s are transformed features of the point cloud set and style image. *cov* is the operation to calculate the covariance matrix of features. One advantage is that the covariance matrix can solve the issue of different sizes between point cloud features and style features. The other is that style information can be extracted by finding covariance statistics between different channels, which is a common way to express a style. Note that we only add one PointLSTM layer but achieve significant progress. 17

4.1 **Comparison of State of the Art** We compare each work with ours on the Tanks and Temples dataset. Our work can stylize closer to style images and preserve enough content. 23



- 4.2 **Further Insight of Stylization** The blue box shows noisy patterns of style image and the red box displays the ignorance of the reflection part of the pool. Our work not only reduces unimportant noisy patterns but also maintains the content of the reflection part of the scene. 24
- 4.3 **User Preference** The bar chart shows the preference comparison of the two methods. We let users vote which video shows less flickering and which video transforms more similarly to the referenced style. 26



List of Tables

4.1	Consistency Scores ($\times 10^{-3}$)	25
4.2	Execution Time	27



Denotation

SfM	運動推算結構 (Structure-from-Motion)
NVS	新視角合成 (Novel View Synthesis)
Stylization	風格轉換
MLP	多層感知神經網路 (Multilayer perceptron)



Chapter 1 Introduction

1.1 Introduction of Style Transfer

Style transfer is a technique that transforms any image we capture into an artistic style. For instance, if we want to make the captured photo look like Van Gogh's painting *Cafe Terrace at Night*, we don't need to imitate the style and draw the photo manually. Instead, we can bring the photo and Van Gogh's painting into the style transformation model and get the stylized result that keeps the content (e.g. shape, structure) of the photo but changes the style (e.g. color and texture) to that of Van Gogh's painting. The effect of style transfer can be shown in 1.1. Style transfer can be achieved using deep neural networks [38]. Image style transfer has been developing for over 5 years, including image-optimization-based methods [11, 24, 25, 27, 36] and model-optimization-based methods [3, 4, 12, 17, 19, 26, 32, 47]. Video style transfer is concerned not only with stylized quality but also with temporal consistency between adjacent image frames. To avoid flickering throughout the stylized video, the results should be similar between frames. Several works [2, 7, 10, 15, 23, 38, 48] have done using temporal loss or feature correlation. In recent years, 3D style transfer becomes a popular and evolving research topic. It combines the knowledge of both image style transfer and novel view synthesis. Novel view synthesis aims to generate the results of unseen views, which can be achieved by constructing the

3D space from the image set of the scene with different views. Then, the 3D style transfer aims to transform the style of the reconstructed 3D representation and generate unseen stylized novel views. The reason why this is popular is that we can experience stereoscopic stylized 3D scenes through virtual reality (VR) and augmented reality (AR) applications. Currently, novel view style transfer can be divided into single-image-based [31, 46] and multi-images(video)-based [5, 16, 18] methods.

1.2 Motivations

We focus on video-based novel view style transfer in our work. Our main motivation is to accelerate the entire pipeline of 3D style transfer methods without losing the visual quality. Here we discuss the approach we decided to take from two perspectives.

End-to-End Pipeline Recently, SfM-learners have been developed to predict the depths and camera poses of the given images by using CNN models. Compared to traditional approaches, SfM-learners can save a lot of time. For single-image novel view style transfer, many works have achieved the end-to-end pipeline. On the other hand, for multi-image novel view style transfer, the existing methods rely on the traditional 3D construction approach, COLMAP [40]. The advantage is that COLMAP generates globally optimized 3D information of the entire scene. That is, the 3D point cloud of different views can be exactly aligned through a world coordinate system. However, it takes too much time if we want to generate unseen stylized scenes. We thus aim to develop an end-to-end multi-images novel view style transfer method for acceleration.

3D Point-cloud-based Method Novel view synthesis can be implemented in two ways. Explicit methods aim to construct the visible 3D point cloud from 2D images, while implicit methods simply utilize MLP as the scene representation. In our method, we decide

to use the explicit stylization approach because of the following two points:

- In our experimental study, current methods using NeRF [29], a notable implicit scene representation approach, may spend one day to optimize for each scene, which cannot reach our goal of efficiency. Instead, we choose to use SfM-learners that construct 3D information of a scene in less than 30 minutes.
- For stylization, NeRF-based methods need to retrain another stylization module if we want to stylize a new scene. Nevertheless, point-cloud-based methods can directly apply the stylization module to the new-coming 3D scene since it is trained on multiple 3D scenes. Moreover, NeRF-based methods should spend about 30 seconds to render a novel view, while the rendering time can be reduced to less than 1 second in our method.

Now we are going to introduce how we implement it. First, we combine the SfM-learner with the stylization module proposed by Huang *et al.* [16]. It stylizes point cloud features using point cloud aggregation modules, and it renders stylized features of novel views. It shows that it can accomplish novel views style transfer with temporal consistency between different views. Then, we do some modifications to this pipeline to make it feasible to generate consistent and content-undamaged results. One of the challenges is that naively back-projecting the complete point cloud reconstructed by SfM-learners may cause ghost effects. Ghost effects appear when we render the point cloud that is imperfectly aligned through a world coordinate system. Current SfM-learners are unavoidable to face this issue since the 3D point cloud of a frame predicted by SfM-learners is view-dependent. It cannot ensure the 3D point clouds of different views is consistent. Therefore, we should adjust the style transfer module to make it suitable for any SfM approaches.

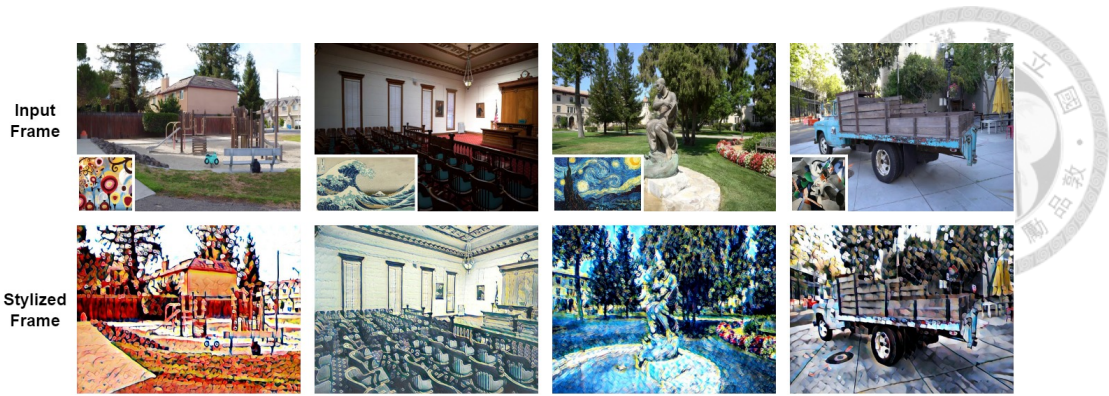


Figure 1.1: We introduce 3D novel view style transfer using an end-to-end pipeline, combined with the latest 3D style transfer model and a learning-based 3D constructor. In this work, we aim to address the efficiency issue in recently proposed approaches, while maintaining the stylized quality of novel views.

We first reduce the ghost effects problem by applying sliding-window back projection to construct time-variant point cloud sets. For each timestamp, the sliding window adds the 3D points of the incoming frame and removes 3D points far from the current frame. Since we observe that 3D consistency between adjacent views of the video is guaranteed for most SfM-learners, it is feasible to stylize such time-variant point cloud sets to avoid most of the ghost effects. Nonetheless, it is important to consider that the stylization of dynamic point clouds may be different in that it will produce flickering output for a video or a sequence of novel views. Figure 1.2 shows a lot of short-term inconsistency between adjacent views. Frequent flickering is harmful to the viewer’s eyes. How to deal with this problem is another important point we need to pay attention to.

One approach we adopt to the style transfer module is long short-term memory [8, 14] for point cloud sequences (PointLSTM) [30]. It is proposed to find the relation between different unordered point clouds. Gao *et al.* [10] shows that ConvLSTM [41] can learn the temporal information of the image sequence so that stylized patterns with similar features remain consistent. We leverage [10]’s idea, but replace image-based LSTM method with point-cloud-based LSTM method. We add a PointLSTM layer after the point cloud aggregation module of Huang *et al.* [16]. Figure 2.1 simply illustrates how it processes on



Figure 1.2: **Temporal inconsistency** The main difference between Huang *et al.* [16] and our work is that the former uses a static point cloud set, while our method uses the sliding-window point cloud set to overcome the ghost effect problem. We show that if we directly stylize the image sequences by Huang’s module, the stylized results may lead to inconsistencies in the red boxes. The main effect is strong and repetitive changes of contrast that causes visual discomfort.

sequential point cloud sets. In our experiments, PointLSTM can extract temporal features between different point cloud sets and thus successfully generate consistent stylized novel views.



1.3 Contribution

Overall, the main contributions of our work are listed below:

- We implement an end-to-end video-to-novel-view style transfer metric. In other words, we can use all learning-based models to generate stylized images of the desired view.
- We operate 3D style transfer on variable point cloud sets while maintaining style consistency across the timeline.
- The 3D construction method is not restricted to traditional time-consuming global-optimized approaches and has similar effects to recent global-optimized style transfer models.
- We also develop stereo video style transfer, a novel view application for a VR system allowing users to experience immersive 3D stylized scenes.



Chapter 2 Related Work

2.1 Image and Video Stylization

The purpose is to transfer a given photo-realistic image or video to a stylized image or video with content (contour, structure, shape, etc.) similar to the original one and style (color distribution, texture) similar to the reference style image.

Most neural style transfer methods use VGG [43] model as a feature extractor to generate feature maps of input images. [11, 24, 25, 27, 36] use image-optimization-based methods. They use feature and Gram matrix similarity as an optimization function to make input images look like style images while keeping their contents. Model-optimization-based methods can efficiently transfer input images to required styles. Huang *et al.* [17] proposes AdaIN that simply scales the normalized content input with the mean and variance of style features. Park *et al.* [32] introduces SANet that uses normalized VGG features as attention targets to match content and style feature, and it combines multiple SANet's outputs as a final stylized feature. Liu *et al.* [26] is similar to SANet but it takes both shallow and deep features into account for attention.

Video stylization requires attention to stylization quality and consistency over time. Chen *et al.* [2] warps the stylized frame to the previous one and compares the difference

between the two frames as a coherent loss. Gao *et al.* [10] uses ConvLSTM to improve the training performance so that it can train on multiple styles. Deng *et al.* [7] proposes MCCNet to learn correlations between style features and input features through coherent loss and it allows similar features from different frames to be transformed into consistent results.

Image and video stylization are restricted in image or video space. If we want to generate stylized results of novel views, these methods can't reach our goal because no 3D information is used.

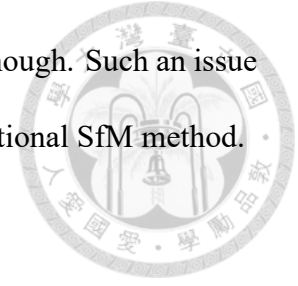
2.2 Structure-from-Motion

Given a sequence of images, structure-from-motion(SfM) aims to construct 3D information (like depths, poses, meshes) from 2D motions.

Traditionally, the pipeline of SfM is correspondence search and incremental reconstruction [40]. The first step includes feature extraction, matching, and geometric verification. This step aims to find the same matching points among overlapping images. Then the second step calculates the depths and camera poses of input images by triangulation [13] and bundle adjustment [45]. [6, 40, 44, 50] can generate globally-consistent 3D reconstruction of the entire image set. However, these methods require enormous computation and therefore cost a lot of time as the number of images increases.

Recently, SfM-learners [1, 22, 28, 52] emerge to predict depths and camera poses by neural network model. The advantage of these methods is that it solves the efficiency problem encountered by traditional methods. On the other hand, the drawback is global consistency cannot be fully maintained because we cannot guarantee that the 3D points of

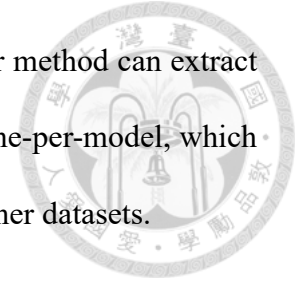
matched 2D features have the same position, even if they are close enough. Such an issue makes it hard to directly apply to the stylization work using the traditional SfM method.



2.3 Novel View Synthesis

Novel view synthesis (NVS) aims to predict the rendered image given an unseen view. There are two kinds of sub-problems: single-image NVS [37, 49] and multi-images NVS [29, 34, 35, 53]. Single-image NVS can generate arbitrary views near the source images. Synsin [49] estimates the depths of given single images and generates novel views using 3D projection and CNN generator. The main contribution they introduce is differentiable rendering to make neural-network models trained thoroughly. Naive rendering selects only the nearest points for each pixel to render; therefore, it only back-propagates over these points and the model cannot be fully trained. In contrast to naive rendering, Synsin uses a weighted sum of 3D points based on distance, allowing the model to back-propagate all points within the Z-buffer. Pixelsynth improves the generation range from Synsin using PixelCNN++[39] that can generate masked regions by using unmasked information as a reference. However, single-image NVS methods are limited to a small range of viewpoints changing. In addition, given the image of the current frame and the pose of the next frame, it cannot be ensured that the predicted image of the next frame is the same as the original image of the next frame. Multi-image NVS methods learn the relationship between different images. Riegler *et al.* [34] fuses the wrapped views to the target view and learned the blended result. NeRF [29] leverages the concept of volume rendering and lets the neural network learn the scene representation (density and color) with 5D (position and ray direction) inputs by MLP. It's no need to know the 3D point information of each 2D feature of input images. The former method can be applied in arbitrary input

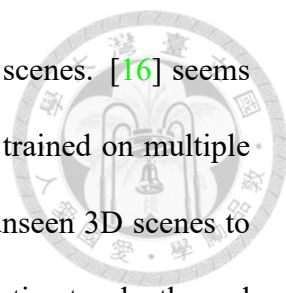
scenes, but the depth map is required for each novel view. The latter method can extract the illumination variance of different angles, yet it is trained per-scene-per-model, which means retraining is needed if we want to get novel view results of other datasets.



2.4 3D Scene Stylization

3D scene stylization combines modern novel view synthesis methods with style transfer techniques. As with video stylization, 3D scene stylization should take transfer consistency into account among novel views. Tseng *et al.* [46] first generates photo-realistic novel views and then applies occlusion-aware consistency loss to ensure the transferred results between novel views can be consistent. Mu *et al.* [31] directly stylizes the point cloud of the input image using AdaAttn to compute attention of features between 3D points and the style image. After that, the rendering model projects 3D stylized features to 2D space and decodes the rendered feature map into an RGB image. Both of them propose single-image-based 3D stylization of a sequence of novel views. However, our input data is a video, so we should take care of the consistency between generated image and the original image of the target pose in the video. Neither of these two tasks can handle the problem if the view changes significantly.

Multi-images-based [5, 16, 18] stylization can yield more diverse viewpoints than single-image-based methods. [5, 18] use NeRF as novel view generator. They either directly modify the weights of MLP that predict the appearance of the position or add a style module to learn stylized features with mutual learning technique. Huang *et al.* [16] uses SfM method to generate 3D information and stylizes directly on point clouds. According to our understanding, NeRF needs retraining when it meets unseen scenes,



which makes it waste a lot of time if we want to stylize on variable scenes. [16] seems to solve the problem because it's arbitrary-style-per-model and it is trained on multiple 3D scenes. It gives the model the ability to successfully transform unseen 3D scenes to consistent output in a sequence of views. Nevertheless, the method estimates depths and poses using the traditional SfM predictor COLMAP, which is time-consuming. Thus, the work of [16] doesn't bring too much benefit on time compared to the NeRF-based 3D stylization module. What we can do to solve the efficiency problem of [16] is to change COLMAP to any learning-based 3D constructor.

It is worth noting that learning-based 3D constructors have not yet been explored in combination with style transfer. The most challenging part is that the geometric inconsistency appears between different timestamps. If we directly back-project the point clouds of all views predicted by SfM-learners, the ghost effect may appear on the rendered image. How to reduce such effects while producing temporally consistent stylized results at the same time can be investigated in depth. In our work, the local-aggregated point cloud set mentioned in the introduction is utilized to reduce the ghost effect, while shifting the point cloud set is necessary for learning-based SfM methods. For example, if we want to generate target views near frame t of input video, we can project the point cloud set of frame t to target views. However, project it to target views near frame $t+k$ might lose a lot of information when k is large. A possible solution is projecting the point cloud set of frame $t+k$ instead so that the rendered results of target views have less blank space.

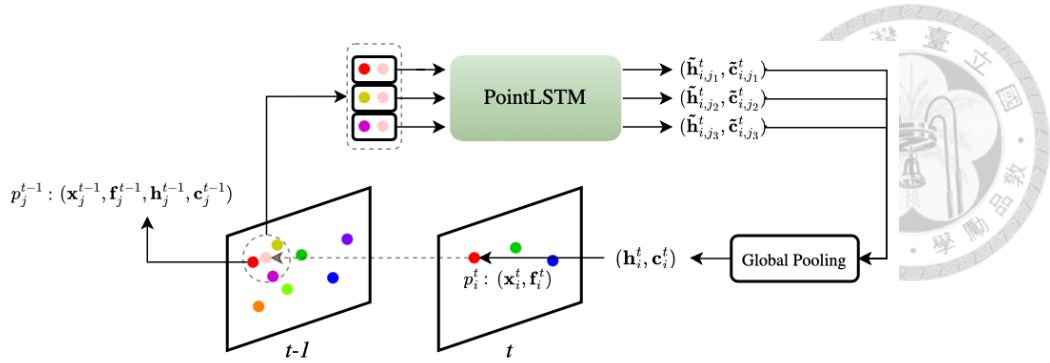


Figure 2.1: **Point feature learning.** The diagram explains how PointLSTM learns. For each point p_i^t in current time t , it first finds k nearest points of the previous time (3 in this example). Then, the model learns pairwise virtual states $(\tilde{h}_{i,j}^t, \tilde{c}_{i,j}^t)$ between p_i^t and previous point p_j^{t-1} . Last, it generates the final states using global pooling among all pairwise virtual states.

2.5 PointLSTM

Recurrent neural network(RNN) and long short-term memory(LSTM) are designed to solve time-related questions such as sentence prediction and video generation. [8, 41] combine LSTM with CNN model to deal with vision tasks, including video stylization. Currently, some works [9, 30, 51] discover that neural network model can learn the time-dependent features on 3D points. PointLSTM first searches the nearest points in the previous frame for each timestamp of points. Then it extracts intra-frame and inter-frame features of point clouds. [30] accelerates learning speed of PointLSTM by proposed point-share states. It's no need to record independent states of individual points. Instead, it calculates the point-sharing hidden state and cell state for each frame. PointLSTM is mainly used for 3D point gesture recognition, which means it has the potential to handle tasks related to point sequences. Dynamic point cloud sets applied in our method change frequently, so adding PointLSTM to the style transformation module we use is considered in order to find the similarity of point cloud sets of adjacent frames and get consistent stylized results in overlapped areas.



Chapter 3 Methodology

In our work, the end-to-end video-to-novel-view style transfer pipeline aims to perform comparable visual quality to current 3D style transfer methods, while enhancing efficiency. We improve the approach of [16] by :

- Replacing COLMAP with Deep3D [22] for acceleration.
- Applying sliding-window back-projection as the pre-processing of the 3D point cloud to reduce ghost effects.
- Introducing PointLSTM [30] to control temporal consistency.

An overview of our work is shown in Figure 3.1.

3.1 Problem Definition

Given an image sequence $\{I_t\}_{t=1}^N$ and an arbitrary style image S , our goal is to generate stylized results of target views that have a similar style to the reference style image and maintain temporal consistency. To achieve this, we first predict depths $\{D_t\}_{t=1}^N$ and poses $\{P_t\}_{t=1}^N$ of each frame $\{I_t\}_{t=1}^N$ and then calculate the corresponding point cloud sets $\{PC_t\}_{t=1}^N$ in 3D space. Next, we stylized point cloud sets by using a point-based trans-

formation module. Finally, we project point clouds to 2D images and generate stylized results of the novel view sequence.



3.2 3D Point Cloud Construction

Considering the efficiency issue, the 3D constructor we choose is Deep3D stabilizer proposed by Lee *et al.* [22] since the generation speed is significantly faster than most methods. For each frame $\{I_t\}_{t=1}^N$, it predicts depths $\{D_t\}_{t=1}^N$ and poses $\{P_t\}_{t=1}^N$ with a fixed intrinsic K . After that, we back-project image planes to 3D space. In our observation, simply projecting all point clouds will cause inconsistent overlapping; therefore, we use sliding-window back projection to determine the dynamic point cloud sets. Each frame $\{I_t\}_{t=1}^N$ has its corresponding point cloud set $\{PC_t\}_{t=1}^N$, where PC_t contains dense point clouds of frame $\{I_i\}_{i=t-w}^{t+w}$.

In addition, each 3D point should have its feature representation. In this work, we use the output from layer *relu_3_1* of VGG19 as the feature source, same as [16]. For each 3D point p back-projected from pixel (u, v) of frame I_t , its feature is assigned as the value at pixel (u, v) of feature map \mathbf{F}_t .

3.3 3D Scene Stylization

In several studies, stylization can be accomplished through transforming the distribution of input features into the distribution of style features [17, 23, 24]. Similarly, [16] learns to transform the distribution of point cloud features into the distribution of target style features. It proposes a point cloud transformation module to optimize the transfor-

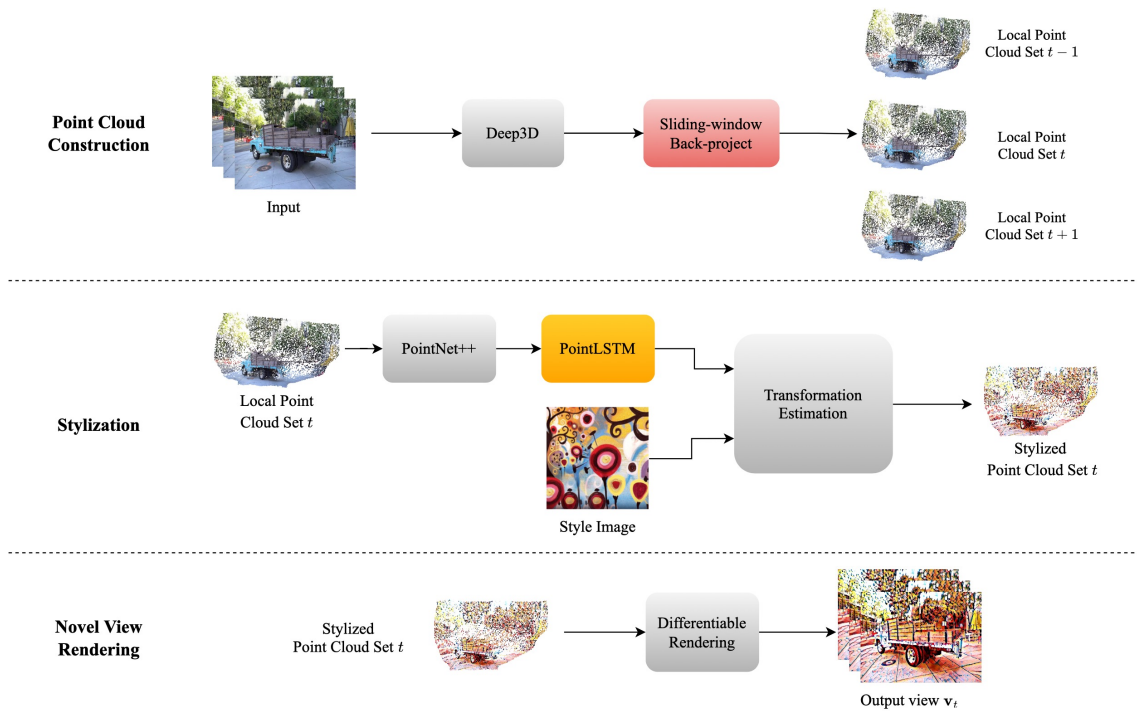


Figure 3.1: **System Overview** Our pipeline consists of point cloud construction and stylization. For point cloud construction, we first generate the depths and poses of each frame using Deep3D [22]. Then, we use sliding-window back projection to assign point cloud sets to each frame by combining 3D points of adjacent frames. After that, we transfer point cloud sets to the desired style using the style transformation module. Last, we use differential rendering proposed by [49] to project 3D features to 2D feature maps and decode feature maps into RGB images.

mation matrix \mathbf{T}_t that converts input point features to stylized features by the following equation:

$$f_t^{cs} = \mathbf{T}_t(f_t^c - \bar{f}^c) + \bar{f}^s, \quad (3.1)$$

where f_t^c and f_t^{cs} are input features and stylized features of point cloud set PC_t respectively, \bar{f}^c and \bar{f}^s are the mean of the input point cloud features and the style image features.

\mathbf{T}_t is calculated by extracting features of the 3D point cloud and style image. For style image, CNN, covariance computation, and MLP are applied to calculate style transformation matrix \mathbf{T}^s . For point cloud, not only covariance computation and MLP but also PointNet++ [33] is used to calculate content transformation matrix \mathbf{T}_t^c . PointNet++ aggregates the whole point cloud to a subset via sampling and grouping. It first chooses a subset of points as centroids using the iteration farthest point sampling method. Then, for each centroid, it extracts the local feature by grouping nearby points. The procedure above determines \mathbf{T}_t by multiplying \mathbf{T}^s and \mathbf{T}_t^c . Note that \mathbf{T}_t^c is different between timestamps because the point cloud set changes sequentially.

The main difference between [16] and our model is that we add PointLSTM [30] layer at the end of point cloud aggregation module, like Figure 3.2. The reason we add the layer is that the stylized result using the original transformation module may be inconsistent, even though the point cloud changes slightly. LSTM can extract temporal information and it's able to learn which part is similar between different point cloud sets. Through obtaining the relationships between different point clouds, stylizing results of different views consistently is feasible. The next paragraph shows the mechanism of PointLSTM.

For each point p_i in PC_t , it first searches neighboring points of PC_{t-1} . We mark the set of these points as $\mathbf{N}(p_i)$. Then, for each pair $(p_i, p'_j), p'_j \in \mathbf{N}(p_i)$, the equation below

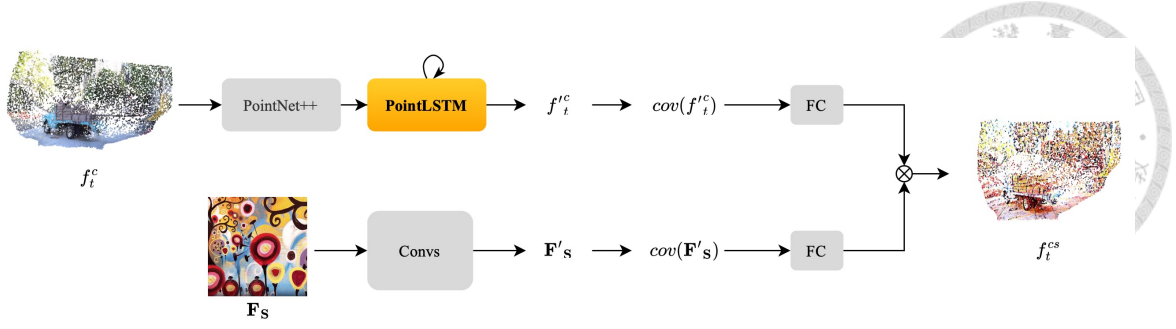


Figure 3.2: **Stylization module** The transformation module converts the point cloud to a specified style by extracting channel-correlated features. For point cloud, we use PointNet++ to aggregate intra-frame point features since too intensive points may cause messy results such as unwanted shapes or textures. Then, the PointLSTM layer aims to learn inter-frame point relations and it can reduce the flickering problem. For style images, only CNN is needed to learn compressed features. Both routes are designed to reduce computational space and get refined representations of features. In particular, $f_t^{c'}$ and F_s' are transformed features of the point cloud set and style image. cov is the operation to calculate the covariance matrix of features. One advantage is that the covariance matrix can solve the issue of different sizes between point cloud features and style features. The other is that style information can be extracted by finding covariance statistics between different channels, which is a common way to express a style. Note that we only add one PointLSTM layer but achieve significant progress.

formulates how to update hidden states and cell states:

$$\begin{aligned} \mathbf{y}_{i,j}^t &= [\mathbf{x}_i^t - \mathbf{x}_j^{t-1}; \mathbf{f}_i^t], \\ \tilde{\mathbf{h}}_{i,j}^t, \tilde{\mathbf{c}}_{i,j}^t &= LSTM(\mathbf{y}_{i,j}^t, \mathbf{h}_j^{t-1}, \mathbf{c}_j^{t-1}), \end{aligned} \quad (3.2)$$

where \mathbf{x}_i^t and \mathbf{f}_i^t are the 3D coordinate and feature of point p_i , $\mathbf{h}_j^{t-1}, \mathbf{c}_j^{t-1}$ are states of point p'_j , and $\tilde{\mathbf{h}}_{i,j}^t, \tilde{\mathbf{c}}_{i,j}^t$ are pair-wised states of current time t between (p_i, p'_j) . $[\cdot]$ is the operation of concatenation. The full operation of the LSTM function can be expressed as the following equations (\odot denotes the Hadamard product):



$$\begin{aligned}
\mathbf{i}^{(t)} &= \sigma(\mathbf{U}^{(i)}\mathbf{y}^{(t)} + \mathbf{W}^{(i)}\mathbf{h}^{(t-1)} + \mathbf{b}^{(i)}), \\
\mathbf{f}^{(t)} &= \sigma(\mathbf{U}^{(f)}\mathbf{y}^{(t)} + \mathbf{W}^{(f)}\mathbf{h}^{(t-1)} + \mathbf{b}^{(f)}), \\
\mathbf{o}^{(t)} &= \sigma(\mathbf{U}^{(o)}\mathbf{y}^{(t)} + \mathbf{W}^{(o)}\mathbf{h}^{(t-1)} + \mathbf{b}^{(o)}), \\
\tilde{\mathbf{c}}^{(t)} &= \tanh(\mathbf{U}^{(c)}\mathbf{y}^{(t)} + \mathbf{W}^{(c)}\mathbf{h}^{(t-1)} + \mathbf{b}^{(c)}), \\
\mathbf{c}^{(t)} &= \mathbf{f}^{(t)} \odot \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \odot \tilde{\mathbf{c}}^{(t)}, \\
\mathbf{h}^{(t)} &= \mathbf{o}^{(t)} \odot \mathbf{c}^{(t)}.
\end{aligned} \tag{3.3}$$

After getting pair-wised states, it uses global pooling method to extract $\mathbf{h}_i^t, \mathbf{c}_i^t$ of point p_i from all relevant pair-wised states:

$$\begin{aligned}
\mathbf{h}_i^t &= g(\tilde{\mathbf{h}}_{i,1}^t, \tilde{\mathbf{h}}_{i,2}^t, \dots, \tilde{\mathbf{h}}_{i,n}^t), \\
\mathbf{c}_i^t &= g(\tilde{\mathbf{c}}_{i,1}^t, \tilde{\mathbf{c}}_{i,2}^t, \dots, \tilde{\mathbf{c}}_{i,n}^t).
\end{aligned} \tag{3.4}$$

On above, n is the count of $\mathcal{N}(p_i)$ and g is the pooling function (max pooling is applied in our work).

3.4 Novel View Rendering

Given a set of novel views with intrinsic \mathbf{K}_v and extrinsic matrix \mathbf{P}_v for each view v , we project stylized point cloud sets to 2D feature maps using pytorch3d [49]. For simplification, we generate stereo views, a special case of novel view synthesis by translating poses \mathbf{P}_t of each video frame t to left view v_t^l and right view v_t^r . The generated feature

maps of v_t^l and v_t^r are projected from stylized point cloud set PC_t . Finally, a pretrained decoder is used to transform feature maps into RGB images.



3.5 Training

We conduct three type of losses, content loss L_c , style loss L_s and temporal consistency loss L_{temp} .

Content Loss To keep the structure and shape between input and output images. We define the content loss function below:

$$L_c = \|\Phi(\mathbf{I}_t) - \Phi(\mathbf{O}_t)\|^2, \quad (3.5)$$

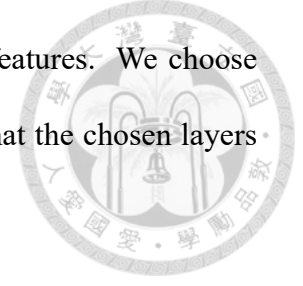
where Φ is the function that extracts the feature of VGG-19 for each input image and \mathbf{I}_t and \mathbf{O}_t are the input frame and the stylized frame at time t . In this work, content loss uses the output of layer *relu_4_1*.

Style Loss Style similarity can be evaluated by using Gram Matrix, a method that calculates the channel-wise correlation of the feature map. Hence, for each style image \mathbf{S} and stylized frame \mathbf{O}_t , style loss is computed as:

$$L_s = \sum_i \|\mathbf{G}(\Phi_i(\mathbf{S})) - \mathbf{G}(\Phi_i(\mathbf{O}_t))\|^2, i \in l_s \quad (3.6)$$

where Φ_i is the function that extracts the feature of layer i of VGG-19, \mathbf{G} computes

Gram matrix, and l_s is the set of layers where style loss extracts features. We choose $l_s = \{relu_1_1, relu_2_1, relu_3_1, relu_4_1\}$ in our work. Note that the chosen layers of both style loss and content loss are referred to [16].



Temporal Consistency Loss To prevent flickering effects, we want the transformation model to form similar results between consecutive point cloud sets. Thus, we propose a temporal consistency loss. We first introduce short-term consistency loss L_{short} at the following equation:

$$L_{short} = \sum_{t=0}^{T-1} \|\mathbf{O}_t - \mathbf{O}_{t+1 \rightarrow t}\|, \quad (3.7)$$

where \mathbf{O}_t is the stylize result projected from PC_t to pose P_t and $\mathbf{O}_{t+1 \rightarrow t}$ is the stylize result projected from PC_{t+1} to pose P_t . We don't use occlusion mask in [10] because there are common points in PC_t and PC_{t+1} . The advantage of this method is that we don't need to recalculate optical flows between frames and handle the disocclusion issue between 2D images. We also used the original warping loss proposed by [2] to train the stylization module and found that it learns similarly, which means our temporal consistency loss is enough.

Moreover, we still want the model to transfer similar results in the long-term temporal case. To solve this, we utilize long-term temporal loss L_{long} :

$$L_{long} = \sum_{t=2}^T \|\mathbf{O}_0 - \mathbf{O}_{t \rightarrow 0}\|. \quad (3.8)$$

We set $T = 6$ for the trade-off between computational space and duration to keep

consistent.

Overall temporal loss L_{temp} is the combination of L_{short} and L_{long} and total loss function L is shown as follows:



$$\begin{aligned} L &= L_c + \lambda_s L_s + \lambda_t L_{temp} \\ &= L_c + \lambda_s L_s + \lambda_t (L_{short} + L_{long}), \end{aligned} \tag{3.9}$$

where λ_s and λ_t are hyper-parameters to adjust the weight of style loss and temporal consistency loss, respectively.

3.6 Implementation Details

To show the generality of our method, we choose to train the modules on 7scenes dataset [42] and test on Tanks and Temples dataset [20]. The device we use is NVIDIA RTX3090. We train the model for 100000 iterations and it costs about 3 days. For hyperparameters, we set $\lambda_s = 0.02$, $\lambda_t = 30.0$ to balance between temporal consistency and stylized quality. The optimizer we use is Adam with default parameters of Pytorch. The learning rate is set to 0.0001 for the first 80000 iterations and is set to 0.00005 for the final 20000 iterations.



Chapter 4 Experiments

4.1 Qualitative Results

Image Stylization In this part, we show the stylized results of our method compared with FVMST [10], ReReVST [48], LSNV [16] and Chiang’s *et al.* [5] in Figure 4.1. For the video-based style transfer method, we warp original frames to novel views via 3D projection before stylization. We find that FVMST and ReReVST can keep the output image in good shape. However, they both have the disadvantage that the degree of conversion to the target style is relatively weak, even though some patterns from stylized images are present. Chiang *et al.* proposed NeRF-based stylization training, but the content of the results is lost obviously. Stylizing features on the point cloud can generate more delicate results. LSNV can stylize results with sharper contours and more colorful visual effects. The only point is that the output of LSNV does not resemble the specified style, and the color distribution is quite different from the original image. Our method can produce stylized images that are not only colorful but also close to style images.

Novel View Stylization In our work, we take stereo videos as an example. We care about the stylizing consistency as well as the illumination of reality. For the Deep3D-NVS+FVMST method, we first use Deep3D to generate novel view sequences of original

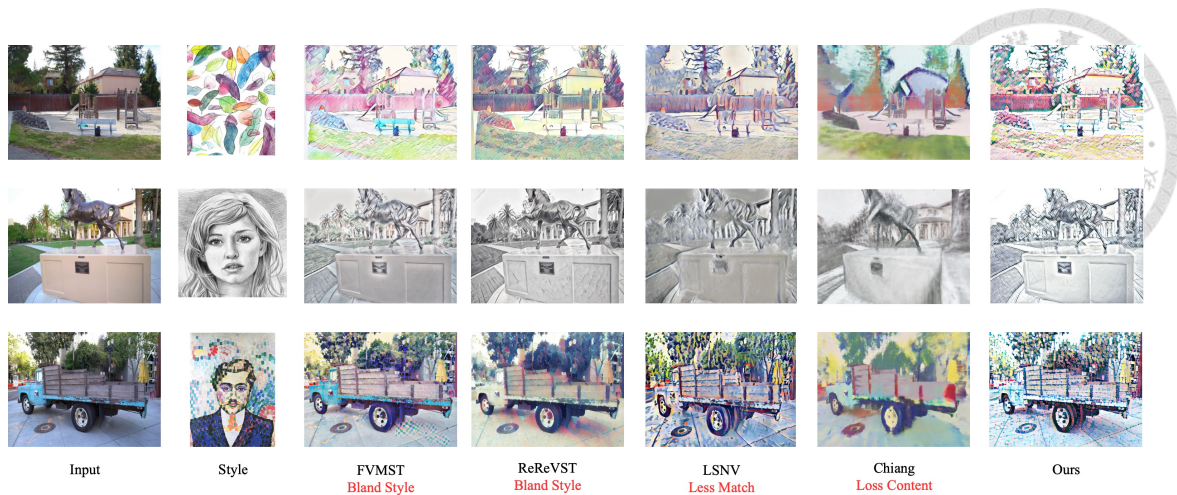


Figure 4.1: **Comparison of State of the Art** We compare each work with ours on the Tanks and Temples dataset. Our work can stylize closer to style images and preserve enough content.

videos and then apply FVMST to stylize them. In 4.2, we can discover that Deep3D-NVS+FVMST can keep consistency in most part of the video and it can take illumination into account, while the noise exists in some places, which make it uncomfortable for the eyes. LSNV has no this issue since the model transforms complete scenes in 3D place and it's less likely to generate noisy patterns due to point cloud aggression. This solution makes consistency better. Nevertheless, it cannot handle the issue of illumination. A 3D point may have different colors when viewed from different angles, but LSNV aggregates these to the same color. Our method can still address the illumination correctness as the illumination part of the point cloud sets constructed by our method are similar. Overall, we take the advantages that we generate few noisy patterns and that we do not neglect the illumination of the scene.

4.2 Quantitative Results

Temporal Consistency We list the consistency score between previous work and our method in table 4.1. We also mark the score with or without PointLSTM to show the

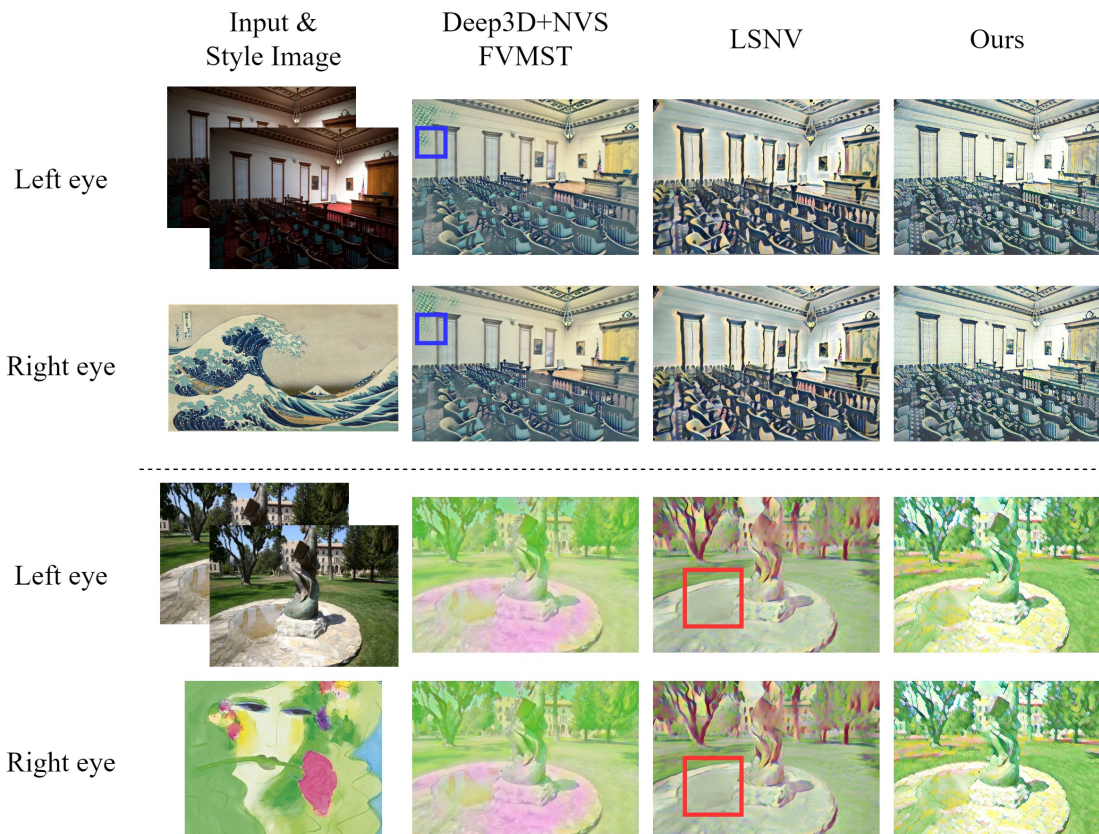


Figure 4.2: **Further Insight of Stylization** The blue box shows noisy patterns of style image and the red box displays the ignorance of the reflection part of the pool. Our work not only reduces unimportant noisy patterns but also maintains the content of the reflection part of the scene.

Table 4.1: Consistency Scores ($\times 10^{-3}$)

Method	Courtroom	Truck	Train	Meetingroom	Panther
Deep3D-NVS + FVMST	1.8961	1.2973	1.5634	1.8451	1.0565
LSNV	3.8639	2.3725	<u>2.1843</u>	<u>2.1197</u>	2.3950
Ours w/o PointLSTM	3.3646	3.2081	5.2779	3.8838	4.3307
Ours	<u>2.6209</u>	<u>2.2512</u>	2.9528	2.7886	<u>2.0083</u>

Temporal Consistency Loss Here we use the flow warping error proposed by [21] between neighboring frames as our evaluation metric. The best score is in **bold** and the second is underlined. Note that our method is tried to get close scores to LSNV since dynamic point cloud sets are unavoidable to face the stylized difference as the time distance stretches.

improvement of our work. First, there is indeed a significant improvement in adding PointLSTM when using sliding-window point cloud sets. Then, it's obvious that Deep3D-NVS+FVMST has the lowest error compared with other methods. On the other hand, our method is competitive with LSNV. Although the error of Deep3D-NVS+FVMST is the lowest, the visual quality isn't necessarily higher than any other method. Notably, our results are similar to LSNV, which means that our method can produce comparable output by using sliding-window point cloud sets constructed by a learning-based method, rather than using a huge point cloud reconstructed by traditional methods.

4.3 User Preference

In this part, we aim to find out the feeling of the user's experience. We recruited 15 participants for our study. Each participant watched 12 different types of stylized videos. Each type of stylized video includes our stylized result and that from one of the compared methods. Participants are asked which result is more consistent and which result has a more similar style to the referenced image. Figure 4.3 shows that our method has higher consistency than Deep3D-NVS+FVMST but is lower than LSNV. Such a finding is in our

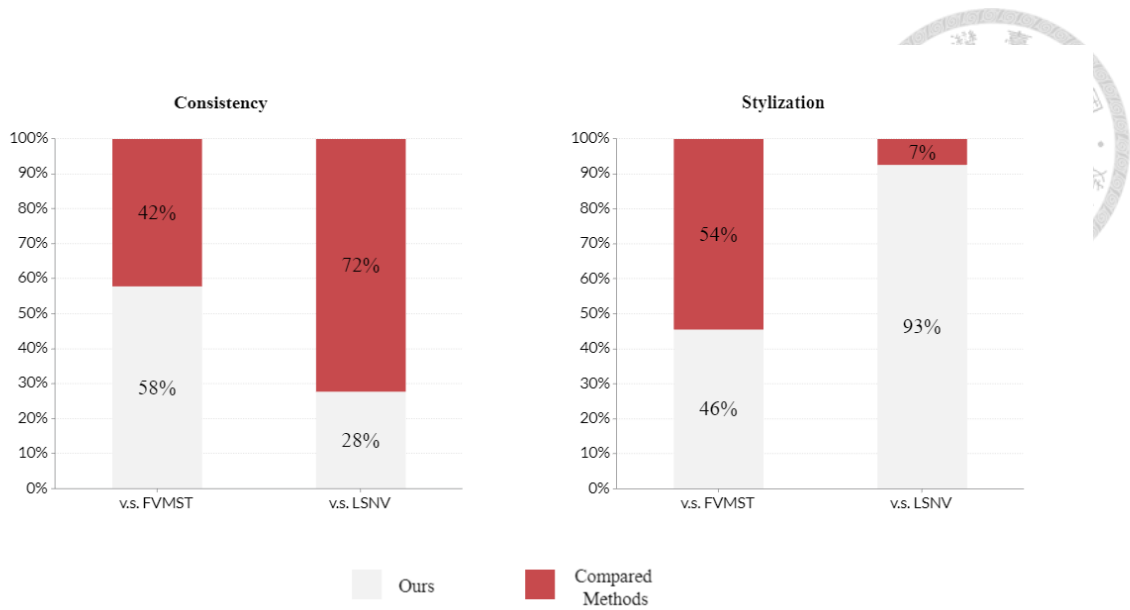


Figure 4.3: **User Preference** The bar chart shows the preference comparison of the two methods. We let users vote which video shows less flickering and which video transforms more similarly to the referenced style.

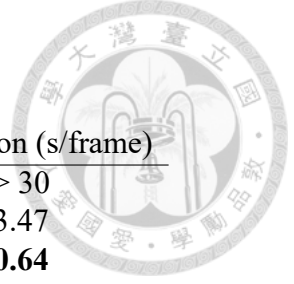
expectation because the 3D scene of LSNV is completely static and it's difficult to beat the consistency score using the dynamic point cloud set. However, we got a significantly higher preference for stylization compared with LSNV although we lose a little bit from Deep3D-NVS+FVMST. We can see that our approach allows for a trade-off between efficiency, consistency, and stylization. For efficiency, we will illustrate the experiment results in the next part.

4.4 Efficiency

We also show our execution efficiency in Table 4.2. For 3D construction, we test the execution time using a video with 500 frames and a resolution of 640*480. For stylization, we compute the total execution time of the entire video and divide it by the number of frames. Obviously, Deep3D performs much better than COLMAP as a 3D constructor since it's at least 10 times faster. In addition, we focus on the execution time among all the different methods. Chiang cost at least 30 seconds per frame, which is extremely slow.

Table 4.2: Execution Time

Method	3D Construction (min)	Stylization (s/frame)
Chiang	259 (COLMAP)	> 30
LSNV	259 (COLMAP)	3.47
Deep3D-NVS + FVMST	24 (Deep3D)	0.64
Ours	24 (Deep3D)	<u>0.72</u>



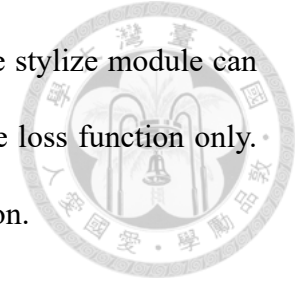
We analyze the runtime of both 3D construction and stylization. The best score is in **bold** and the second is underlined. Deep3D performs much more efficiently than COLMAP, which means SfM-learners are helpful for the acceleration of the stylization pipeline. Moreover, we modify the method of LSNV, yet we perform better than it, which means we bring some progress.

The efficiency problem is a big issue among recent implicit 3D stylization approaches. Compared to these approaches, stylizing 3D point clouds runs faster. However, it costs 3 seconds per frame using LSNV. Our method and Deep3D-NVS+FVMST can run for less than 1 second per frame. As a result, our work performs similarly to video-based methods and outperforms current 3D-based methods in efficiency. Note that the stylization time of Deep3D-NVS+FVMST does not include the novel view generation time from original videos, and our method may become even faster than Deep3D-NVS+FVMST if the runtime of the novel view generation is considered.

4.5 Discussion

The first approach we tried is adding a temporal consistency loss function to deal with the flickering issue. However, this approach is not sufficient to solve it since the transformation module cannot handle time-related problems. It deserves noticing that there are two kinds of techniques to cope with temporal information. One is the self-attention module to let the model transform overlapped parts similarly, and the other is to apply a recurrent layer to remember time-variant features as the hidden states. We choose the

latter one as our approach. We found that adding PointLSTM in the stylize module can successfully deal with the consistency issue compared to adding the loss function only. As the result, hidden states are needed to control temporal information.



Even though we solve the flickering issue by using the proposed pipeline, it's apparent that our consistency is still lower than video-based stylization after novel view synthesis. The main reason might be that after point aggression in PointNet++, the structure of aggregated points differs. We can't add the PointLSTM layer before point cloud aggregation since the capacity of the memory is not huge enough to find the previous point and extract features over 100M points. Therefore, we put it after the point aggregation module as a trade-off between consistency and memory space. Another reason may be that the correctness of the 3D constructor affects the training. In our experiment, we train the model using point cloud sets reconstructed by Deep3D in order to achieve an end-to-end pipeline. Note that the point cloud constructed by Deep3D is not as consistent as COLMAP, while Deep3D generates 3D results that are closer to other learning-based 3D predictors or even RGB-D datasets. Training on point cloud sets from Deep3D can apply to other methods than training on that from COLMAP. Both of the reasons can be investigated more in our future work.



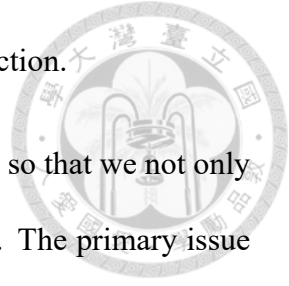
Chapter 5 Conclusion

In this work, we introduce an end-to-end pipeline for 3D scene style transfer. Our approach accomplishes the following improvements. First, we use the SfM-learner method instead of traditional 3D constructors to make the entire pipeline much faster. Second, we apply sliding-window back-projection as the pre-processing of the 3D point cloud, which is useful to reduce ghost effects. Finally, we use PointLSTM to control the temporal consistency of different point cloud sets. Overall, we save a lot of time compared with existing multi-image 3D scene style transfer approaches, while still producing favorable results. Besides, our method can flexibly choose different learning-based 3D reconstruction methods in our implementation, and can also be extended to using RGB-D images as inputs. One limitation is that our stylization module costs much memory and time in the training stage although the inference stage is faster.

We discuss some possible applications and extensions of our work in AR/VR. One of the applications that have been realized is that we can generate binocular stylized videos and put them into the VR showroom. This application allows users to experience a stereoscopic effect. Another potential application is that we can extend our approach based on a 360 surrounding video of a scene. After that, we do 3D stylization of the captured video instantly. Finally, we can add the stylized point clouds of keyframes to the AR system and users can watch novel views' style transfer directly. With these potential applications, our

work can serve as a useful tool with better efficiency in 3D reconstruction.


In the future, we will try to improve the diversity of novel views so that we not only use this work for stereo video generation but also arbitrary traveling. The primary issue is how to decide which point cloud sets should be projected to target views. Probably, several tools such as pose graphs can deal with the relationship between the current view and keyframes. If we make a breakthrough in these issues, it will be very helpful in VR/AR application mentioned above. In summary, our method has the opportunity to become more useful and it might become very promising.




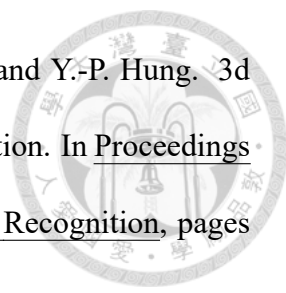


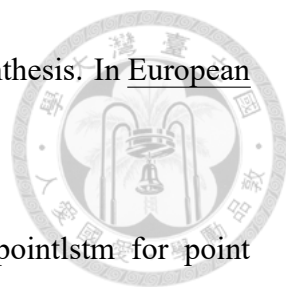
References

- [1] J. Bian, Z. Li, N. Wang, H. Zhan, C. Shen, M.-M. Cheng, and I. Reid. Unsupervised scale-consistent depth and ego-motion learning from monocular video. Advances in neural information processing systems, 32, 2019.
- [2] D. Chen, J. Liao, L. Yuan, N. Yu, and G. Hua. Coherent online video style transfer. In Proceedings of the IEEE International Conference on Computer Vision, pages 1105–1114, 2017.
- [3] D. Chen, L. Yuan, J. Liao, N. Yu, and G. Hua. Stereoscopic neural style transfer. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 6654–6663, 2018.
- [4] T. Q. Chen and M. Schmidt. Fast patch-based style transfer of arbitrary style. arXiv preprint arXiv:1612.04337, 2016.
- [5] P.-Z. Chiang, M.-S. Tsai, H.-Y. Tseng, W.-S. Lai, and W.-C. Chiu. Stylizing 3d scene via implicit representation and hypernetwork. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pages 1475–1484, 2022.
- [6] D. Crandall, A. Owens, N. Snavely, and D. Huttenlocher. Discrete-continuous optimization for large-scale structure from motion. In CVPR 2011, pages 3001–3008. IEEE, 2011.

- 
- [7] Y. Deng, F. Tang, W. Dong, H. Huang, C. Ma, and C. Xu. Arbitrary video style transfer via multi-channel correlation. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, pages 1210–1217, 2021.
- [8] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2625–2634, 2015.
- [9] H. Fan and Y. Yang. Pointrnn: Point recurrent neural network for moving point cloud processing. arXiv preprint arXiv:1910.08287, 2019.
- [10] W. Gao, Y. Li, Y. Yin, and M.-H. Yang. Fast video multi-style transfer. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pages 3222–3230, 2020.
- [11] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2414–2423, 2016.
- [12] S. Gu, C. Chen, J. Liao, and L. Yuan. Arbitrary style transfer with deep feature reshuffle. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 8222–8231, 2018.
- [13] R. I. Hartley and P. Sturm. Triangulation. Computer vision and image understanding, 68(2):146–157, 1997.
- [14] S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

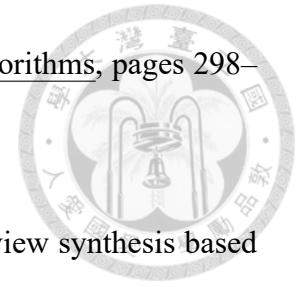
- 
- [15] H. Huang, H. Wang, W. Luo, L. Ma, W. Jiang, X. Zhu, Z. Li, and W. Liu. Real-time neural style transfer for videos. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 783–791, 2017.
- [16] H.-P. Huang, H.-Y. Tseng, S. Saini, M. Singh, and M.-H. Yang. Learning to stylize novel views. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 13869–13878, 2021.
- [17] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In Proceedings of the IEEE international conference on computer vision, pages 1501–1510, 2017.
- [18] Y.-H. Huang, Y. He, Y.-J. Yuan, Y.-K. Lai, and L. Gao. Stylizednerf: consistent 3d scene stylization as stylized nerf via 2d-3d mutual learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 18342–18352, 2022.
- [19] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In European conference on computer vision, pages 694–711. Springer, 2016.
- [20] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. ACM Transactions on Graphics (ToG), 36(4):1–13, 2017.
- [21] W.-S. Lai, J.-B. Huang, O. Wang, E. Shechtman, E. Yumer, and M.-H. Yang. Learning blind video temporal consistency. In Proceedings of the European conference on computer vision (ECCV), pages 170–185, 2018.

- 
- [22] Y.-C. Lee, K.-W. Tseng, Y.-T. Chen, C.-C. Chen, C.-S. Chen, and Y.-P. Hung. 3d video stabilization with depth estimation by cnn-based optimization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10621–10630, 2021.
- [23] X. Li, S. Liu, J. Kautz, and M.-H. Yang. Learning linear transformations for fast image and video style transfer. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 3809–3817, 2019.
- [24] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Universal style transfer via feature transforms. Advances in neural information processing systems, 30, 2017.
- [25] Y. Li, N. Wang, J. Liu, and X. Hou. Demystifying neural style transfer. arXiv preprint arXiv:1701.01036, 2017.
- [26] S. Liu, T. Lin, D. He, F. Li, M. Wang, X. Li, Z. Sun, Q. Li, and E. Ding. Adaattn: Revisit attention mechanism in arbitrary neural style transfer. In Proceedings of the IEEE/CVF international conference on computer vision, pages 6649–6658, 2021.
- [27] F. Luan, S. Paris, E. Shechtman, and K. Bala. Deep photo style transfer. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4990–4998, 2017.
- [28] R. Mahjourian, M. Wicke, and A. Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 5667–5675, 2018.
- [29] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng.

- 
- Nerf: Representing scenes as neural radiance fields for view synthesis. In European conference on computer vision, pages 405–421. Springer, 2020.
- [30] Y. Min, Y. Zhang, X. Chai, and X. Chen. An efficient pointlstm for point clouds based gesture recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 5761–5770, 2020.
- [31] F. Mu, J. Wang, Y. Wu, and Y. Li. 3d photo stylization: Learning to generate stylized novel views from a single image. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 16273–16282, 2022.
- [32] D. Y. Park and K. H. Lee. Arbitrary style transfer with style-attentional networks. In proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 5880–5888, 2019.
- [33] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Advances in neural information processing systems, 30, 2017.
- [34] G. Riegler and V. Koltun. Free view synthesis. In European Conference on Computer Vision, 2020.
- [35] G. Riegler and V. Koltun. Stable view synthesis. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 12216–12225, 2021.
- [36] E. Risser, P. Wilmot, and C. Barnes. Stable and controllable neural texture synthesis and style transfer using histogram losses. arXiv preprint arXiv:1701.08893, 2017.
- [37] C. Rockwell, D. F. Fouhey, and J. Johnson. Pixelsynth: Generating a 3d-consistent

- 
- experience from a single image. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 14104–14113, 2021.
- [38] M. Ruder, A. Dosovitskiy, and T. Brox. Artistic style transfer for videos. In German conference on pattern recognition, pages 26–36. Springer, 2016.
- [39] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. arXiv preprint arXiv:1701.05517, 2017.
- [40] J. L. Schonberger and J.-M. Frahm. Structure-from-motion revisited. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4104–4113, 2016.
- [41] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. Advances in neural information processing systems, 28, 2015.
- [42] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2930–2937, 2013.
- [43] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.
- [44] C. Sweeney. Theia multiview geometry library: Tutorial & reference, 2015.
- [45] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment

—a modern synthesis. In International workshop on vision algorithms, pages 298–372. Springer, 1999.



- [46] K.-W. Tseng, Y.-C. Lee, and C.-S. Chen. Artistic style novel view synthesis based on a single image. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, pages 2258–2262, June 2022.
- [47] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022, 2016.
- [48] W. Wang, S. Yang, J. Xu, and J. Liu. Consistent video style transfer via relaxation and regularization. IEEE Transactions on Image Processing, 29:9125–9139, 2020.
- [49] O. Wiles, G. Gkioxari, R. Szeliski, and J. Johnson. Synsin: End-to-end view synthesis from a single image. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 7467–7477, 2020.
- [50] C. Wu. Towards linear-time incremental structure from motion. In 2013 International Conference on 3D Vision-3DV 2013, pages 127–134. IEEE, 2013.
- [51] C. Zhang, M. Fiore, I. Murray, and P. Patras. Cloudlstm: A recurrent neural model for spatiotemporal point-cloud stream forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, pages 10851–10858, 2021.
- [52] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1851–1858, 2017.
- [53] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely. Stereo magnification: Learning view synthesis using multiplane images. arXiv preprint arXiv:1805.09817, 2018.