Novel View Synthesis from Casually Recorded Videos

by

Eric Ding Qian

B.S. Computer Science and Electrical Engineering
Massachusetts Institute of Technology, 2021

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2021

Author .....
Department of Electrical Engineering and Computer Science
August 16, 2021

Certified by .....
William T. Freeman
Thomas and Gerd Perkins Professor of EECS
Thesis Supervisor

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Novel View Synthesis from Casually Recorded Videos

by

Eric Ding Qian

Submitted to the Department of Electrical Engineering and Computer Science
on August 16, 2021, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Generating new, photorealistic views of a scene given only a single video is a difficult task that computer vision researchers have worked on for decades. This problem has recently seen a resurgence in interest due to its potential application in areas such as virtual reality. However, current novel view synthesis techniques are not suitable for the short, casual videos that people typically record. Such videos deviate from the setups that these approaches typically use, where there are dense, high-resolution images of the scene. In this paper, we propose a method for refining an initial, coarse scene geometry which we then use for novel view synthesis on short video sequences. The core of our method is a geometry refinement step where we project the geometry to source views to remove inconsistent points. This refined geometry provides important shape and appearance information in data poor regions that would otherwise be difficult to accurately render. We evaluate our approach on the RealEstate10K dataset and demonstrate that compared to prior work, we synthesize views that are more temporally consistent.

Thesis Supervisor: William T. Freeman
Title: Thomas and Gerd Perkins Professor of EECS

Acknowledgments

I would first like to thank Professor Freeman for all his guidance throughout my time in his lab. I remember being fascinated by your lectures when I first took the computer vision class. It was what inspired me to pursue research in the first place.

I'm extremely grateful to Xiuming Zhang and Zhoutong Zhang for all their advice, encouragement, and time. Thank you for not only sharing your invaluable expertise but for teaching me how to conduct academic research and how to best communicate incredibly technical concepts.

Additionally, thank you to my friends and communities who have made my time here at MIT an unforgettable one.

Next, I would like to express my gratitude to my girlfriend, Jessica Xu, for supporting all my endeavors and for encouraging me to be bold in my pursuits.

Finally, I want to thank my parents, Hong Ding and Min Qian, and my brother, Allen Ding, for all their lifelong love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Imagine watching a video you took during your favorite vacation. No matter how many times you watch it, the trajectory through the scene will remain the same. If you want to see what the scene would look like from different viewpoints, you would have to return to where you recorded the video. But what if you could generate new views to explore the scene freely from the comfort of your own home?

To generate views of a scene in the traditional graphics pipeline, a skilled computer graphics designer needs to painstakingly create a high-quality 3D model, manually specify photorealistic materials, and run computation-intensive light simulations. This process is not only very tedious but also unforgiving, as any imperfection will result in unpleasant artifacts in the final renders.

To alleviate this problem, a new subfield, neural rendering, emerges at the intersection of computer vision and computer graphics, where researchers exploit the strong representation powers of deep neural networks to learn parts or the entirety of the rendering process. These machine learning techniques avoid the current bottleneck of manual work by learning directly from only videos or photos.

However, existing works on scene neural rendering typically use dedicated computer vision datasets that contain high resolution videos which were deliberately taken to capture what the scene looks like from many different positions and angles. Models learn from the dense views to implicitly reason about geometry or utilize the high quality images to explicitly reconstruct an accurate geometry which is used for

rendering.

These prior works' reliance on high resolution, comprehensive inputs limits their applicability since most casually recorded videos do not share those qualities. Even current state of the art techniques experience a considerable decrease in performance and produce temporally inconsistent views given these unstructured videos. This project focuses on novel view synthesis from a single, fly-through, monocular RGB video. Because this type of video is so ubiquitous, improving view synthesis on them would enable greater accessibility to creating and disseminating experiences.

In this paper, we propose a method for novel view synthesis on short video sequences that enables greater movement and produces fewer temporal inconsistencies than existing approaches. We begin by joining per-frame depth predictions to generate a fused mesh. Then, we refine the fused mesh's appearance and shape by differentiably rasterizing the mesh to source views and optimizing 2D losses. For each point on the refined mesh, we assign a set of learnable features which encode information about the scene. These features are jointly optimized with a rendering network which produces the final output.

We evaluate our method on scenes from the RealEstate10K [26] dataset. The quick fly-through style of the videos in the RealEstate10K dataset resembles that of the videos which people casually record and is also challenging for novel view synthesis since objects are seen in few frames and are viewed from few angles. We show that we qualitatively outperform existing works on novel extrapolated views and are quantitatively competitive on holdout views.

Chapter 2

Related Work

Our work's primary goal is to improve novel view synthesis results. However, we also address the challenge of producing 3D geometries that are well-suited for view synthesis tasks. In this chapter, we review related works in both 3D reconstruction and neural rendering.

2.1 3D Reconstruction

3D reconstruction is the problem of generating a scene's 3D geometry given only images of the scene. Popular representations of 3D geometries include point clouds and meshes.

2.1.1 Analytical Reconstruction

When the cameras' poses are known, multi-view stereo (MVS) can be used to generate a dense geometry. MVS finds correspondences [7] between images and uses geometric constraints to triangulate [5] the location of the points in the world. If camera poses are not known, structure from motion (SfM) techniques can solve for them up to a scale. These poses can be further refined using bundle adjustment and can be used in the MVS pipeline. COLMAP [18] and ORB-SLAM2 [10] are examples of systems that combine SfM and MVS to solve the 3D reconstruction problem. High quality

correspondences are necessary for MVS to produce good results. Consequently, when input images are of low quality or distorted, or when the scene contains flat, textureless, or reflective surfaces, MVS can produces noisy reconstructions. As a result, MVS cannot be assumed to work well on general videos. Indeed, Zhou et al. [26] find that COLMAP is prone to failure on scenes in the RealEstate10K dataset.

2.1.2 Learning-Based Reconstruction

Naively, one can use monocular depth prediction models [11] [3] to estimate geometries per frame and combine them. However, depth scales need to be calibrated and inconsistencies in depth predictions between views often result in layered surfaces in the fused reconstruction. Luo et al. [8] use optical flow between pairs of images to improve depth consistency across frames but doesn't completely solve the problem, especially for regions far from the camera and for views with little overlap.

Other methods directly optimize a single, explicit underlying geometry. These approaches achieve better consistency across multiple views but their formulations often limit their accuracy. Discrete, occupancy-based approaches [2] struggle with scaling to larger scenes while maintaining quality. Others volumetric representations [4] have difficulty with modeling thin objects and other fine details.

Recently, neural implicit representations have been used to solve the 3D reconstruction problem [19]. Such methods do not explicitly predict occupancy but rather optimize a signed distance function. These techniques can yield good results but require dense views of the scene.

We use the pipeline proposed by Luo et al. [8] to initialize geometries since it produces dense geometries for a wide array of scenes and videos. To alleviate its global consistency limitations, we use the scene's appearance signals to improve the underlying geometry.

2.2 Neural Rendering

Neural rendering is the class of deep learning techniques that enable the generation of new views of a scene under user-controlled parameters, like lighting, scene appearance, and camera pose. This project focuses on the popular problem of novel view synthesis given user-specified camera poses.

2.2.1 Geometry-Based Rendering

Many rendering approaches use some form of a scene's geometry to assist with view synthesis. Aliev et al. [1] use point clouds, which they obtain by capturing a scene with an RGB-D camera, and associate each point with learned features that can be projected to desired views and inpainted using a rendering network. Other techniques [14] [15] operate on estimated meshes and encode features on the mesh surface which is then projected to the novel view and passed to a decoder network for rendering. Wiles et al. [22] predict novel views using only a single image by predicting depth and features for every pixel, which are differentiably rasterized to the target view and refined with a GAN.

While geometries can provide great scaffolding, when they are incorrect, downstream rendering quality can suffer dramatically because the models expect the geometry to provide an accurate representation of the scene's shape. Examples of failure modes include 1) reconstruction is incomplete with holes in the surface, so points from behind can bleed to the front 2) incorrect geometries occlude important objects. The bleeding problem is especially prevalent when using point clouds since even slight movement can cause the camera ray to slip between two points representing a surface, leading to temporal inconsistencies.

Our method is similar to the one proposed by Aliev et al. [1] since we also assign learnable features to our geometries. We also draw inspiration from the differentiable rasterizer proposed by Wiles et al [22]. Importantly, we differ from existing works in that we do not require a high fidelity geometry initialization.

2.2.2 Geometry-Free Rendering

Another class of image-based rendering techniques do not need a scene's geometry.

Zhou et al. [26] create multi-plane images (MPIs), which are RGBA images at different depths, and then warps and renders them from back to front in order to create new images. Other approaches [16] use plane sweep volumes which are similar to MPIs except the planes are defined fronto-parallel to the target camera view. Such approaches have limited camera movement and are not well suited for our goal of rendering camera paths with substantial departure from source views.

One of the most recently proposed ideas for novel view synthesis is the use neural radiance fields (NeRF) [9]. NeRF forgoes using geometries and even convolutional layers, instead optimizing a 5D function conditioned on viewing location and direction. The original NeRF produces results that are reasonably consistent across views but has difficulty rendering views whose camera poses are far from the training poses. Additional research has extended NeRF to work with sparser views [23] and have made NeRF more generalizable [21] by using features from source frames. These models enable rendering novel views farther from training poses but their reliance on neighboring frames results in artifacts when the set of keyframes changes. Additionally, they necessitate a robust keyframe selection strategy. NeRF-based models are also slow to both train and perform inference with since every camera ray must be sampled at many locations [13].

Because geometry-free rendering approaches have no geometric priors, they typically requires many more source images to achieve the same quality as their geometry-based counterparts and offer less camera movement before rendering quality deteriorates.

Chapter 3

Approach

Our system pipeline is divided into three main components: geometry initialization, geometry refinement, and rendering. In this chapter, we first share relevant background on the video depth estimation method we use. We then describe each step of our method in detail.

3.1 Consistent Video Depth Estimation

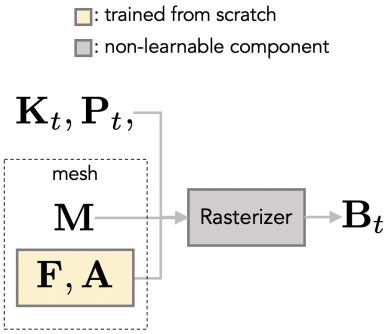
We use the pipeline proposed by Luo et al. [8] to initialize geometries. Their method fine-tunes independent monocular depth estimates using dense optical flow fields. Optical flow is predicted between pairs of images to establish pixel correspondences. Intuitively, points in one image should match the optical flow when projected to the second image (minimize spatial loss), and the same point in two images should project to the same point in world space (minimize disparity loss). These flow-based geometric losses are back propogated to make the depth predictions more consistent across frames.

Since the flows are used to calculate the spatial and disparity loss that refine depth predictions, errors in initial flow prediction can substantially decrease the quality of the final geometries. The authors acknowledge that frames with little overlap generate poor flow estimations and use a forward-backward consistency check to determine an image pair's suitability. In practice, we find that some of the optical flow predictions

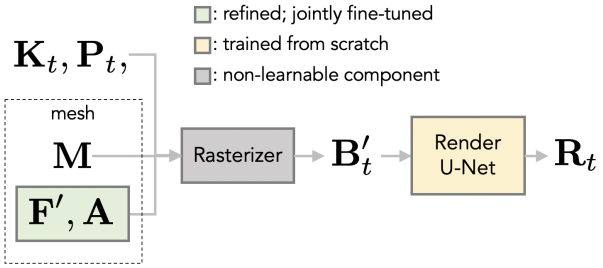
in the preprocessing step are incorrect even when the frames are close together and pass the consistency check, as can be seen in Figure 3-4.

Additionally, the videos used in the original paper keep an object at the center of focus while moving around it, meaning most pairs of images have sufficient overlap. Videos in the RealEstate10K dataset, however, quickly move across the scene, so objects do not stay in frame for long. Consequently, we can only construct optical flow pairs between frames that are temporally close together since those far apart temporally are also very distant spatially and have little overlap. While we hoped that information to improve consistency could be implicitly propagated by the chain of flows, we find that even when a region is seen in both the beginning and the end of a video, the depth predictions are quite different.

As a result, in order to use these geometries for novel view synthesis, we first must refine them in order to remove the incorrect points.



(a) The geometry refining setup. A fused mesh composed of the vertices and triangles $\mathbf{M}$, vertex features $\mathbf{F}$, and triangle alphas $\mathbf{A}$ are rasterized to a view specified by camera parameters $\mathbf{K}_t$, $\mathbf{P}_t$ to produce buffer $\mathbf{B}_t$.



(b) The rendering setup is similar to the geometry refining one. The refined mesh features are augmented to produce $\mathbf{F}'$. The augmented features $\mathbf{F}'$, refined alphas $\mathbf{A}$, and geometry definition $\mathbf{M}$ are rasterized to buffer $\mathbf{B}'_t$ which is rendered by a rendering network to produce the final output $\mathbf{R}_t$.

Figure 3-1: Diagrams showing our method setup for the geometry refining step and rendering step.

3.2 Geometry Initialization

For a given input video consisting of n source images that are each $W \times H$, let $\mathbf{I}_i \in [0,1]^{H \times W \times 3}$ be the i-th image, and let $\mathbf{K}_i \in \mathbb{R}^{3\times3}$ and $\mathbf{P}_i \in \mathbb{R}^{3\times4}$ be the corresponding camera's intrinsics and extrinsics respectively. We use the aforementioned consistent video depth network to generate a dense depth estimate $\mathbf{D}_i$ for every image.

3.3 Geometry Refining

From depth estimate $\mathbf{D}_i$, we can generate mesh $\mathbf{M}_i$ with $2(H-1)(W-1)$ triangles by connecting every set of three adjacent pixels (such that a 2 x 2 square of pixels forms 2 triangles that share an edge). We back project this mesh into world space using the predicted depth. Every vertex $\mathbf{x} \in \mathbb{R}^3$ on the mesh is assigned a feature vector which is a concatenation of its RGB value according to its color in $\mathbf{I}_i$ and its depth prediction in $\mathbf{D}_i$. Additionally, each triangle is initialized with an alpha value of 0.5. Let $\mathbf{F}_i$ be the set of vertex RGB-D features describing the mesh, and let $\mathbf{A}_i$ be the set of triangle alpha values.

We use a smooth piece-wise function $c(x)$ to clamp features and alphas in the range $[0,1]$ so that RGBA values are valid even after back propagation updates. Let $t(x) = \frac{1}{2}\tanh(7(x-\frac{1}{2})) + \frac{1}{2}$. Then we have

$$c(x) = \begin{cases} t(x) + 0.001 - t(0.001) & x < 0.001 \\ x & 0.001 \leq x \leq 0.999 \\ t(x) + 0.999 - t(0.999) & x > 0.999 \end{cases} \tag{3.1}$$

We join meshes from K different views to form fused mesh $\{(\mathbf{M}_k, \mathbf{F}_k, \mathbf{A}_k)\}_{k=1}^K$. We use a differentiable rasterizer from Pytorch3D [12] and a custom alpha compositing blending function to rasterize the fused mesh to a buffer $\mathbf{B}_t$ in target view t, which is specified by a camera $\mathbf{K}_t$ and pose $\mathbf{P}_t$. Formally, $\mathbf{B}_t = R(\mathbf{K}_t, \mathbf{P}_t, \{(\mathbf{M}_k, c(\mathbf{F}_k), c(\mathbf{A}_k))\}_{k=1}^K)$. Importantly, we omit the depth channel from being clamped. $\mathbf{B}_t \in \mathbb{R}^{H\times W\times4}$, with

the top three channels $\mathbf{T}_t$ being the rendered RGB and the last channel $\mathbf{D}'_t$ being the rendered depth. Blending and alpha compositing are discussed in further detail in Section 3.5.

We minimize the difference between the top three channel buffer and the target source view. We also impose anisotropic total variation loss TV on the rendered depth D as a form of regularization since we expect surfaces to be flat.

$$TV(D) = \sum_{i,j} |D_{i,j} - D_{i,j-1}| + |D_{i,j} - D_{i-1,j}| \tag{3.2}$$

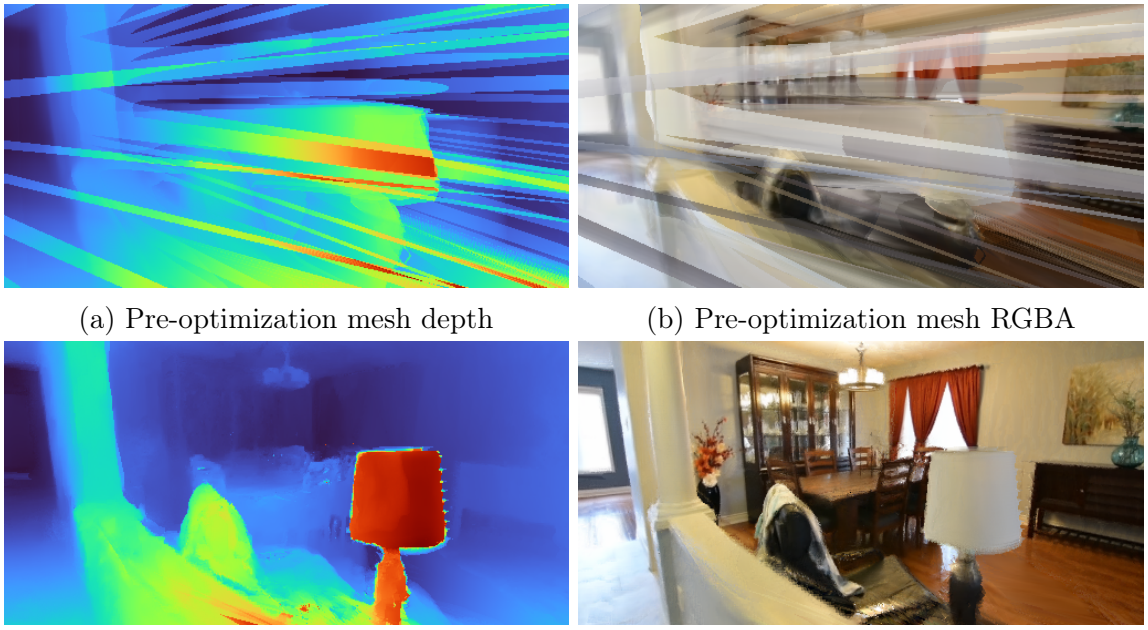
Thus the whole loss function to refine the geometries is

$$\mathcal{L} = \|\mathbf{T}_t - \mathbf{I}_t\|_2^2 + \lambda_{TV} TV(\mathbf{D}'_t) \tag{3.3}$$

where λ_{TV} is the weight of the total variation loss.

With this method, a point that is observed as having different colors when viewed from different angles can have its colors adjusted accordingly to be the average. However, a point which is spatially inconsistent cannot be fixed by changing its color alone since it will appear inconsistent in some source view. Because we accumulate many points in the z-buffer and initialize alphas at 0.5, incorrect geometries' alphas will have large gradients pushing them to 0 while the correct geometries behind it will change very little. Since points with $\alpha = 0$ effectively have no impact on the rendered buffer, incorrect geometries can be eliminated in this fashion.

Thus, we are able to directly and jointly optimize the geometry's shape and appearance, using only source images as supervision. An example of the geometry optimization can be seen in Figure 3-2.



(a) Pre-optimization mesh depth

(b) Pre-optimization mesh RGBA

(c) Post-optimization mesh depth

(d) Post-optimization mesh RGBA

Figure 3-2: Mesh RGBA optimization results projected to a source view. Many of the floating artifacts are removed in the optimization process. Furthermore, the incorrectly distorted regions on the curtain and painting are refined away.

3.4 Rendering

Once the fused geometry has been refined, it can be used for rendering. The optimized RGBA $\{(\mathbf{F}_k, \mathbf{A}_k)\}_{k=1}^K$ from the refinement step are used during rendering. In addition to the optimized RGB values, each vertex is also associated with 28 additional features. Let $\{(\mathbf{F}'_k)\}_{k=1}^K$ be this augmented set of features.

Similar to in the geometry refinement step, we differentiably rasterize the mesh to a feature buffer $\mathbf{B}'_t = R(\mathbf{K}_t, \mathbf{P}_t, \{(\mathbf{M}_k, c(\mathbf{F}'_k), c(\mathbf{A}_k))\}_{k=1}^K)$. This feature buffer is then processed by the rendering network to produce the final result $\mathbf{R}_t$.

The rendering network is a U-Net [17] with 2 downsampling layers, 2 skip connections, and 2 upsampling layers. Every downsampling layer contains 2 convolution layers, each followed by ReLU. The final convolution in the downsampling layers has stride 2 to accomplish the downsampling.

We minimize the L_2 error between the final rendered image and the target source view in addition to the losses in the geometry refining step. The loss for training the

(a) First 3 channels of $\mathbf{B}'_t$ (b) Rendered image (c) Target source image

Figure 3-3: Example of a intermediate buffer and resulting rendered view

rendering network and fine tuning the geometries is

$$\mathcal{L} = \|\mathbf{R}_t - \mathbf{I}_t\|_2^2 + \|\mathbf{T}_t - \mathbf{I}_t\|_2^2 + \lambda_{TV} TV(\mathbf{D}'_t) \tag{3.4}$$

3.5 Alpha Compositing

To perform alpha compositing, we first project the mesh to the target view in screen space. Then, for every pixel, we accumulate the 40 nearest points on the mesh in the z-direction in a z-buffer in increasing order. Given its original alpha α_i, the i-th point's effective alpha α'_i is determined front to back $\alpha'_i = \alpha_i \prod_j^{i-1}(1 - \alpha'_j)$. Additionally, every point is assigned a probability w_i of affecting the pixel in question based on its depth and proximity to the center of the camera ray. A pixel's final value v is determined by compositing the features of the points in the buffer $v = \sum_j \alpha'_j w_j v_j$. This process is repeated for every channel. A triangle's features are determined by interpolating its vertices' features. Our alpha compositing is implemented to work with the existing Pytorch3D framework.

3.6 Implementation Details

Optical Flow Preprocessing

Optical flows are generated between views that are 1, 2, 3, and 4 frames apart using the method proposed by Teed et al. [20]. Views farther then this distance have con-

siderably less accurate flow predictions and decrease depth estimate quality.

Consistent Video Depth Network

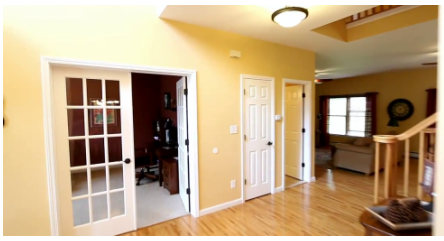
Our depth network backbone is MiDaS [11]. The depth network is trained for a total of 40 epochs. During the first 10 epochs, we only optimize the embedding layer to correct the scale while freezing MiDaS. In the final 30 epochs, we jointly train the depth prediction network and the embedding layer. The network is trained using the ADAM [6] optimizer with a learning rate 10^{-5}, $\beta_1 = 0.5$, and $\beta_2 = 0.9$.

Geometry Refining

We select the first frame, last frame, and 7 uniformly sampled intermediate frames to fuse. We use a subset of the frames due to reduce memory usage and to speed up the process. Geometry is refined for 30 epochs. For vertex RGBs, we use ADAM with a learning rate 0.001, $\beta_{1_{RGB}} = 0.5$, $\beta_{2_{RGB}} = 0.9$. For triangle alphas, we use ADAM with a learning rate 0.01, $\beta_{1_{alpha}} = 0.5$, $\beta_{2_{alpha}} = 0.9$. Additionally, we set $\lambda_{TV} = 10^{-5}$.

Rendering Network

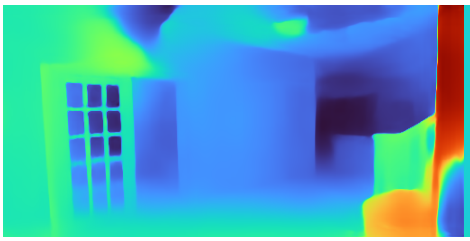
We train the painting network for 40 epochs with ADAM with learning rate 0.005, $\beta_{1_{paint}} = 0.5$, $\beta_{2_{paint}} = 0.9$. For geometry tuning, we use the same parameters as in the geometry refining step.



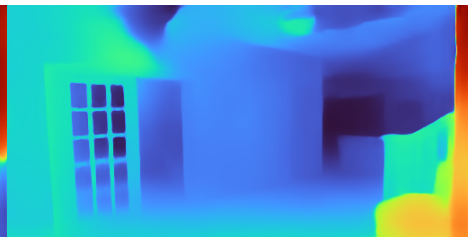
(a) Image 1

(b) Image 2

(c) Estimated optical flow from pixels in image 1 to image 2



(d) Depth prediction for Image 1



(e) Depth prediction for Image 2

Figure 3-4: Figures showing where consistent video depth fails. In 3-4c, each pixel's color encodes the flow's direction of movement, and the color's saturation is directly proportional to the magnitude of the 2D displacement. Neighboring points on wall surfaces undergo similar movement, but the sharp color boundaries on the walls between the white and pink regions incorrectly suggest that portions of the wall are stationary while adjacent parts move significantly. Consequently, the depth prediction is incorrect in those areas.

Chapter 4

Results

We compare our method with state of the art novel view synthesis methods on 8 scenes from the challenging RealEstate10K dataset. Additionally, we justify our design choice in our experiment with different model designs. We demonstrate that our approach qualitatively outperforms current state of the art methods on novel extrapolated views and are quantitatively competitive with existing works on held out source views.

4.1 Dataset

The RealEstate10K dataset is a large dataset generated from YouTube videos. Because the videos are from YouTube, no ground truth information about camera parameters or scaling is known. However, the dataset contains camera poses which were estimated by SLAM and refined using bundle adjustment. On average, each scene we use has 200 source views. We resize all the images to be 512×256 pixels and hold out 10% of the source images which are used for testing.

In addition to the holdout views from the input video, we also generate novel poses for every scene by interpolating between the first and last source poses. Novel pose rotation is interpolated using spherical linear interpolation. Because there is no ground truth for the custom novel poses we generate, we evaluate these novel views qualitatively on how photorealistic they are.

4.2 Comparisons

We compare our approach with leading geometry-based and geometry-free novel view synthesis methods. All models are trained using the same views except for Stable View Synthesis which comes pretrained and can be used on previously unseen environments.

NeRF

NeRF optimizes a 5D function that predicts the color and volume density given a point's 3D location and viewing direction. It determines a pixel's color by sampling many different points along the camera ray. We train NeRF for 200000 iterations on every scene. We sample rays linearly in inverse depth and without normalized device coordinates.

IBRNet

IBRNet uses a lot of the same principles as NeRF, except it extracts features from neighboring source views instead of querying a 5D function. These features are processed by a ray transformer to produce the final color and density prediction for the target pixel. We fine-tune IBRNet for 60000 iterations on every scene. We use the 9 closest source poses during test time. To determine neighboring keyframes, we calculate source pose similarity as the weighted sum of the difference in the source pose rotation and translation from the target's.

Stable View Synthesis (SVS)

SVS encodes features from every source image and back projects them onto a 3D mesh that was generated using COLMAP MVS and Delaunay reconstruction. During rendering, for every pixel in the target view, a neural network generates an aggregated feature vector based on the source features on the mesh and the viewing direction. These aggregated features are then rendered by a rendering network. SVS can be used out of the box, so we use their pretrained network.

(a) Ours

(b) NeRF

(c) IBRNet

(d) SVS

(e) Ground truth image

Figure 4-1: Holdout view comparisons. NeRF produces blurry renderings on the left part of the image. The other methods produce good results.

(a) Ours

(b) NeRF

(c) IBRNET

(d) SVS

(e) Sample source image

(f) Sample source image

Figure 4-2: Novel view comparisons with a large camera baseline. We correctly render large portions of the scene's geometry whereas other methods struggle with this view. SVS suffers from an occlusion due to incorrect reconstruction. IBRNet has substantial line artifacts. NeRF renders part of the fridge and the light but otherwise fails. We provide examples of source frames to show that most of the regions in the target view are visible during training. Thus, these failures are not because those regions were occluded in training images.

(a) Ours (b) NeRF

(c) IBRNet (d) SVS

(e) Sample source image (f) Sample source image

Figure 4-3: Novel view comparisons with a small camera baseline. Our method produces sharper results and has fewer artifacts. NeRF has a lot of blurring and even some incorrect colors covering large portions of the wall and table. IBRNet produce many wave-like artifacts on both the wall and the table, distorting the scene's geometry. SVS has some geometric artifacts at the far right tip of the table which is duplicated, and the segment of the wall on the right by the white door. Additionally, SVS has incorrect geometries for the light on the top left and the boundary between the ceiling and wall on the left. Our geometry is correct except for the slight curve in the wall. We exhibit slightly less detail on the table compared to SVS.

		Scene 1	Scene 2	Scene 3	Scene 4*	Scene 5	Scene 6	Scene 7*	Scene 8	Mean
PSNR ↑	SVS	29.69	26.85	28.44	-	30.59	26.03	-	30.92	28.70
	NeRF	28.82	26.25	26.46	24.46	29.12	25.49	27.33	27.24	26.93
	IBRNet	**31.42**	**31.82**	**34.50**	**29.06**	**31.02**	31.11	31.15	**35.86**	**32.03**
	Ours	**31.35**	**31.82**	33.46	28.11	**30.80**	**32.77**	**32.39**	34.72	**31.92**
SSIM ↑	SVS	**0.959**	0.921	**0.957**	-	**0.968**	0.918	-	0.9672	**0.948**
	NeRF	0.883	0.774	0.782	0.795	0.853	0.786	0.836	0.811	0.817
	IBRNet	0.949	**0.949**	**0.957**	**0.916**	0.894	**0.951**	**0.938**	**0.973**	**0.943**
	Ours	0.940	0.929	0.940	0.904	0.885	**0.950**	**0.938**	0.959	0.932
LPIPS ↓	SVS	**0.031**	0.053	**0.036**	-	**0.028**	0.051	-	**0.020**	**0.037**
	NeRF	0.125	0.254	0.371	0.202	0.184	0.204	0.192	0.224	0.218
	IBRNet	0.050	**0.042**	0.043	**0.055**	0.075	0.035	0.055	0.025	0.047
	Ours	0.044	0.050	0.049	0.059	0.096	**0.020**	**0.049**	0.021	0.048

Table 4.1: Quantitative comparison of our approach and state of the art methods on holdout views. COLMAP MVS failed on Scene 4 and Scene 7, so the required mesh could not be generated for SVS. We evaluate performance on three metrics: peak signal to noise ratio (PSNR), structural similarity index measure (SSIM), and perceptual metric (LPIPS) [25]. For PSNR and SSIM, higher scores are better, whereas for LPIPS, lower scores are better. Numbers in bold are within 1% of the best.

4.3 Discussion

On the holdout views, NeRF produces temporally consistent but blurry renderings, which is reflected in its low scores across all metrics. IBRNet and SVS excel on the holdout views. Since adjacent frames in the video are spatially very close to each other and both approaches leverage the source images to directly generate features for rendering, their strong performance here is expected. IBRNet, however, produces many unnatural flickering artifacts between frames. This is likely because IBRNet does not optimize features in canonical space which can result in different per-frame estimates for the same region in space, causing the temporal inconsistencies. SVS also introduces some flickering but to a lesser degree. While we do not report the best scores, our results are more temporally consistent - something which cannot be measured by quantitative metrics. We include a link to the video results in the Appendix so the reader can see our method's improved temporal consistency.

NeRF and IBRNet do not perform well on the custom novel views. Both of these methods rely on learned estimates of volume density and color. When training poses all view the same point in space from similar angles, it is understandable that the network has difficulty generalizing to points on the surface that were not directly

		Scene 1	Scene 2	Scene 3	Scene 4	Scene 5	Scene 6	Scene 7	Scene 8	Mean
PSNR ↑	U-Net	**31.35**	**31.82**	**33.46**	**28.11**	**30.80**	**32.77**	**32.39**	**34.72**	**31.929**
	U-Net + Gate	30.63	30.96	32.19	26.39	29.42	31.45	29.78	33.78	30.57
	U-Net + No Ft	29.92	29.65	31.16	26.81	29.91	29.87	31.07	32.62	30.12
SSIM ↑	U-Net	**0.940**	**0.929**	**0.940**	**0.904**	**0.885**	**0.950**	**0.938**	**0.959**	**0.932**
	U-Net + Gate	0.927	0.914	0.930	0.878	0.867	0.939	0.906	**0.952**	0.915
	U-Net + No Ft	0.919	0.893	0.917	0.883	0.865	0.917	0.920	0.936	0.908
LPIPS ↓	U-Net	**0.044**	**0.050**	**0.049**	**0.059**	**0.096**	**0.020**	**0.049**	**0.021**	**0.048**
	U-Net + Gate	0.059	0.066	0.063	0.081	0.143	0.031	0.083	0.025	0.067
	U-Net + No Ft	0.066	0.086	0.085	0.077	0.131	0.039	0.068	0.039	0.073

Table 4.2: Results of different model designs. Numbers in bold are within 1% of the best.

sampled on any training camera ray. Indeed, in poorly rendered regions, the disparity maps produced by these methods are incorrect. NeRF degeneracy with limited source views supports the findings by Yu et al. [23]. On custom novel views, SVS produces artifacts on some of the same regions that look good in the holdout views. This disparity is likely due to poorly reconstructed geometries. Occluding geometries have the occluded objects' features projected onto them in the SVS feature aggregation step. Because holdout views are so close to the source views, there are relevant features to use when rendering holdout views. However, on our custom novel poses, there could be potentially no relevant features of the newly occluded regions on the occluding surface, causing this drop in performance. Because we remove many inconsistent points during our geometry refinement step, we don't suffer from this issue as much.

It's important to note that NeRF and IBRNet produce remarkable results on datasets where there are dense source views and limited camera movement. Similarly, SVS does well on scenes where MVS produces high quality reconstructions and there are sufficient source views to help compensate for imperfect geometries. These methods are well-suited to scenes that satisfy the assumptions of their setups. Novel view synthesis from casually recorded videos is a difficult task because there are virtually no assumptions that can be made about the data setting or quality.

4.4 Model Experiments

We compare our final model with one that uses gated convolutions and one that does not augment points with learnable features in Table 4.2.

Many novel view synthesis methods report better results using gated convolutions [24] when performing inpainting; however, we do not see such improvements. Instead, we find that gated convolutions result in lower PSNR and SSIM scores, although they're within 5% of the non-gated version. LPIPS increases by a relative 39%, which shows that using gated convolutions definitively decreases performance in our case. We also find that when we do not add learnable features to the geometry, performance decreases even further, although this is expected since material information besides color can no longer be represented.

Chapter 5

Conclusion

We presented a method for using a scene's appearance information in source images to improve an initial geometry prediction and enable better view synthesis on short videos sequences. Starting with a rough estimate of the scene geometry, our method improves geometry appearance and shape by differentiably rasterizing the mesh to source views and optimizing 2D losses. We assign learnable features to points on the refined mesh which are jointly trained with a rendering network. We demonstrate that scene geometry can be improved using only source images as supervision, which can benefit other geometry-based rendering methods. Our method surpasses state of the art neural rendering techniques on short videos, where we produce more temporally consistent extrapolated views and achieve similar quality on holdout views.

There are a few areas to explore to improve the accuracy and efficiency of our method. While we show success in improving scene geometries and appearance by optimizing RGBA values, the initialized depth predictions need to be reasonable since we only remove points but don't have a strategy for adding new ones. Additionally, points that minimally affect the rasterizations because the points in front of them have high alphas are still included in the forward pass. Because the same point in 3D space is often visible in multiple frames, there can be many unnecessary points. Removing these points can considerably improve rasterizing speed. Additionally, similar to most recent works in neural rendering, ours is currently limited to static scenes, so adapting this method to work on dynamic scenes would be a very exciting future direction.

Appendix A

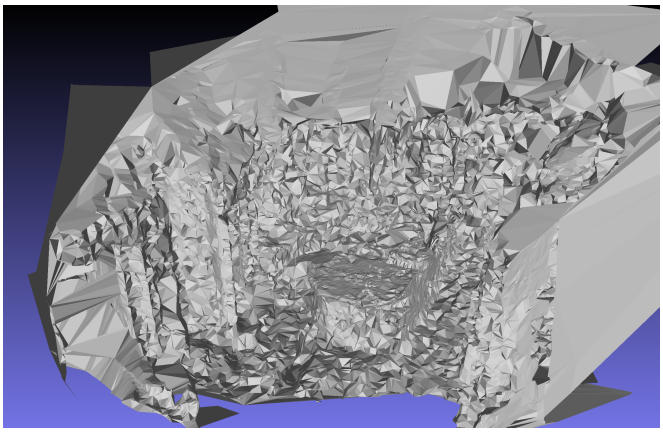
Supplementary Material

A.1 Video Results

Link to the supplemental video comparing our method to state of the art approaches on the source poses and novel poses.

A.2 Additional Figures

(a) COLMAP's dense point cloud



(b) COLMAP's Delaunay mesh

(c) Our fine-tuned mesh

Figure A-1: Comparison of our reconstruction with COLMAP's. Notice how the appearance of our mesh is much clearer than COLMAP point cloud, especially on the floor. Additionally, much of the noise which is present in the COLMAP reconstructions is not present in ours. Because we use per frame depth estimates, we also recover geometries that are seen in few frames, like the floor and the window in the foreground, which COLMAP fails to do.

(a) Ours

(b) NeRF

(c) IBRNet

(d) SVS (Reconstruction fails)

(e) Sample source image

(f) Sample source image

Figure A-2: More novel view comparisons with a small camera baseline. Our method produces sharper results and suffers from fewer artifacts. NeRF cannot render the details on the tile flooring and is overall quite blurry. IBRNet has a lot of artifacts like on the tiling which is wavy, and on the column which is heavily distorted. Our method produces a slight distortion on the top of the column.

Bibliography

[1] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics, 2020.

[2] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent, 2019.

[3] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel Brostow. Digging into self-supervised monocular depth estimation, 2019.

[4] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. Atlasnet: A papier-mâché approach to learning 3d surface generation, 2018.

[5] Richard I. Hartley and Peter F. Sturm. Triangulation. *Comput. Vis. Image Underst.*, 68(2):146–157, 1997.

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

[7] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[8] Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen, and Johannes Kopf. Consistent video depth estimation, 2020.

[9] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020.

[10] Raul Mur-Artal and Juan D. Tardos. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, Oct 2017.

[11] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer, 2020.

[12] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020.

[13] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps, 2021.

[14] Gernot Riegler and Vladlen Koltun. Free view synthesis. In *European Conference on Computer Vision*, 2020.

[15] Gernot Riegler and Vladlen Koltun. Stable view synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[16] Andre Rochow, Max Schwarz, Michael Weinmann, and Sven Behnke. Fadiv-syn: Fast depth-independent view synthesis, 2021.

[17] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[18] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[19] Jiaming Sun, Yiming Xie, Linghao Chen, Xiaowei Zhou, and Hujun Bao. Neuralrecon: Real-time coherent 3d reconstruction from monocular video, 2021.

[20] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow, 2020.

[21] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering, 2021.

[22] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image, 2020.

[23] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images, 2021.

[24] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas Huang. Free-form image inpainting with gated convolution, 2019.

[25] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018.

[26] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images, 2018.