International Joint Conference on Neural Networks (IJCNN), Padova, Italy, July 2022.

NeuralMVS: Bridging Multi-View Stereo and Novel View Synthesis

Radu Alexandru Rosu
Autonomous Intelligent Systems
University of Bonn
Bonn, Germany
rosu@ais.uni-bonn.de

Sven Behnke
Autonomous Intelligent Systems
University of Bonn
Bonn, Germany
behnke@ais.uni-bonn.de

***Abstract*—Multi-View Stereo (MVS) is a core task in 3D computer vision. With the surge of novel deep learning methods, learned MVS achieves more complete depth maps than classical approaches, but still relies on building a memory intensive dense cost volume. Novel View Synthesis (NVS) is a parallel line of research and has recently seen an increase in popularity with Neural Radiance Field (NeRF) models, which optimize a per scene radiance field. However, NeRF methods do not generalize to novel scenes and are slow to train and test. We propose to bridge the gap between these two methodologies with a novel network that can recover 3D scene geometry as a distance function, together with high-resolution color images. Our method uses only a sparse set of images as input and can generalize well to novel scenes. Additionally, we propose a coarse-to-fine sphere tracing approach in order to significantly increase speed. We show on various datasets that our method reaches comparable accuracy to per-scene optimized methods while being able to generalize and running significantly faster.**

***Index Terms*—image-based, novel-view, multi-view**

I. Introduction

Multi-view Stereo (MVS) recovers depth and geometry from multiple images with known camera poses. This is usually done with classical methods like COLMAP [1], Gipuma [2], or MVE [3] by searching for correspondences along epipolar lines. These algorithms lack learned components and cannot cope with challenging conditions like imperfect calibration, blurry or incomplete images, and heavy occlusion [4], [5].

Recent Novel View Synthesis (NVS) methods like NeRF [6] recover geometry as a byproduct of view synthesis. Geometry is represented as a radiance field which is multi-view consistent between all images. This has the advantage of recovering more complete depth while being more robust to imperfect images than classical methods. Main disadvantages of NVS methods are the large processing time and a lack of generalization. They require per-scene training which can take up to several days. Additionally, inference speed is also limited—often requiring multiple minutes to synthesize a full image together with the corresponding depth.

More recent NVS approaches like pixelNeRF [7] and IBRNet [8] improve the reconstruction speed by having a training phase which allows the model to generalize to novel scenes and thus require only little per-scene fine-tuning. Inference is still slow as the network needs to query the radiance field multiple times during synthesis and requires a very dense sampling of the view frustum.

In our approach, we leverage ideas from MVS and NVS and combine them in a new learning-based method that generalizes well to novel scenes and reaches comparable reconstructions to per-scene optimized methods while requiring only a fraction of the time.

First, our method accelerates the ray marching step by differentiable sphere tracing. While ray marching requires hundreds of samples per ray in order to achieve good accuracy, sphere tracing can reach the object surface with only a few iterations by predicting for each ray sample its jump towards the next one.

Second, instead of tracing one ray per pixel, our approach starts by tracing on a coarse image which is iteratively refined until we reach the full resolution. This coarse-to-fine method further alleviates the labor-intensive step of finding the 3D surface of the object.

Third, we propose a new scheme for the selection of conditioning views based on a Delaunay triangulation of the input views. We find that this is more temporally stable than methods based on proximity of viewing direction.

Fourth, we introduce a loss that encourages the network to output a confidence map for the novel RGB-D view. These confidence values align well with parts of the image that are undersampled or occluded and may be used to inform further reconstruction or refinement methods.

In summary, our contributions are:

- a new learning-based novel view synthesis method which generalizes to unseen scenes,
- an efficient coarse-to-fine approach based on differentiable sphere tracing to recover depth with few samples conditioned on a set of input views, and
- a loss that encourages the network to output a confidence map for each novel view produced.

The general pipeline of our method is shown in Fig. 1 and the core components are detailed in Fig. 2.

This work has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2070 - 390732324.

Fig. 1. NeuralMVS processes multiple input views to synthesize a novel colored view with corresponding depth and gives an estimate for the output confidence.

II. Related Work

Several methods for learned MVS have gained popularity lately. MVSNet [5] proposes an end-to-end differentiable model to learn depth inference from unstructured stereo. Features are extracted from images and a dense cost volume is built using samples at regular intervals. The volume is regularized using 3D convolutions and then used to regress a depth map. In contrast in our approach, we do not define the depth samples a priori, but rather let the network learn where to sample using a differentiable sphere tracer.

The work of Darmon *et al*. [9] further builds on MVSNet and shows that depth can be recovered by using a color reconstruction loss. Similarly, we only employ an RGB loss, and do not supervise the depth map as in many settings an accurate ground-truth may not be available.

Recently, NeRF [6] has gained popularity for synthesizing highly-detailed novel views. NeRFs represent the scene as a radiance field and optimize it using differentiable volume rendering. The volume rendering step is computationally expensive since it samples the 3D space densely at regular intervals. One main contribution of NeRF is the introduction of the positional encoding in the context of NVS that enables the model to learn high-frequency details. In order to recover the 3D surface precisely, Mildenhall et al. propose a hierarchical sampling strategy that optimizes two NeRF models: one for coarse samples and one for fine samples closer to the surface. In our work, we leverage the positional encoding for our ray marching step and propose sphere tracing as a method to alleviate the regular sampling of NeRF.

Scene Representation Networks (SRN) [10] is another approach which uses sphere tracing for traversing the rays in 3D space. However, this method is only able to recover low-frequency detail of the scene while we recover fine details by directly conditioning our model on the image features.

Other methods like FaDIV-Syn [11] propose to recover novel views of the scene without reconstructing depth by warping input views into the target frame at a series of predefined depth planes and letting the network learn how to best render the novel view. In contrast, we infer both the depth and the novel view jointly by explicitly letting the network modify the depth planes which are used to project input views.

MVSNerf [12] is a general network that can recover radiance fields conditioned on input views. They explicitly construct a cost volume using sweeping planes. The cost volume is processed by 3D convolutions into a neural encoding volume and the local voxels features are used to output density and color along the ray. A limitation of their work is that the neural encoding volume is represented in the frustum of a reference view. As a result, only the contents of the scene that are visible from the reference view can be fine-tuned and rendered in high detail. In contrast, our approach doesn't define a fixed cost volume and rather aggregates image features from nearby view onto the casted rays. This allows us to model arbitrary scenes while dynamically changing the input views.

PixelNeRF [7] achieves generalization by aggregating features from nearby images onto the ray samples. The aggregated image features together with positional encodings are passed to a NeRF network that output final radiance and color. However, the ray sampling strategy is the same as the original NeRF, requiring hundreds of samples and slowing down inference. In contrast, we let the network learn the spacing between samples, greatly reducing their number and achieving higher rendering speed.

Similarly to PixelNeRF, IBRNet [8] proposes to aggregate image features onto the ray. Additionally, they use a ray transformer that enables ray samples to attend to each other and better reason about occlusions. In our method, the samples along the ray can communicate front-to-back through the usage of an LSTM that dynamically predicts the jump between samples.

Local Light Field Fusion [13] recover novel-view by promoting input views to a local light field representation and blending them at the target view location. However, their method is only demonstrated on front facing scenes due to the multi-plane approach. Our method recovers both depth and color and can render views from arbitrary scenes.

III. Method

Given a set of source views, our method synthesizes depth and color image for a novel target view pose. The core idea is to recover a depth map for the target view such that warping source views onto it results in an image that matches the target as close as possible. Our method can thus be viewed as two jointly trained networks where the first one recovers the geometry of the scene and a second network which uses that geometry as a proxy onto which the source views are projected to recover the novel view.

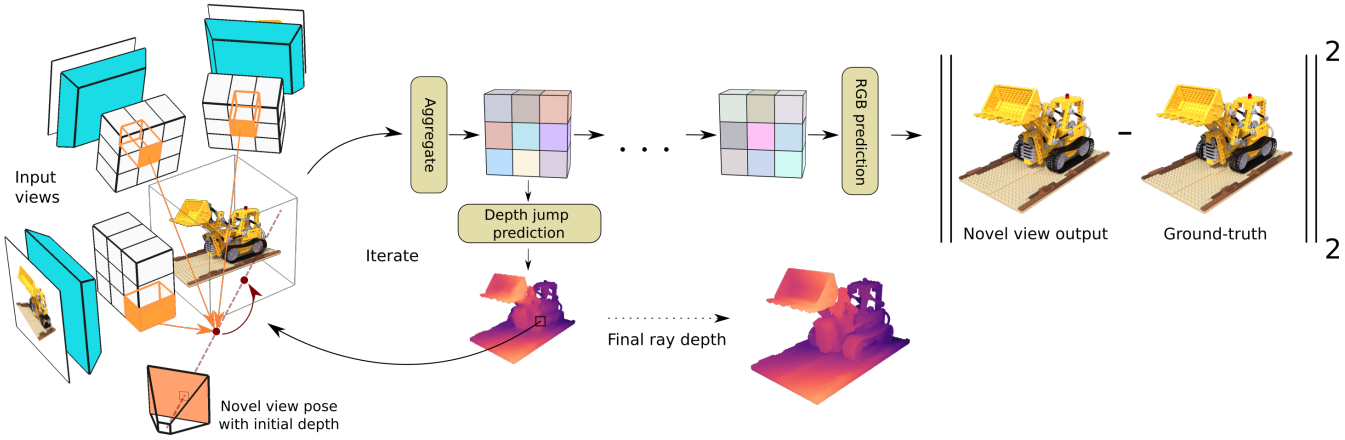

Fig. 2. High-level features from the input view images are aggregated onto each ray sample. For each ray, a recursive network predicts the jump towards the next sample (red arrow). This process is repeated for a fixed number of iterations. At the last ray iteration, the final aggregated features are passed through a rendering network in order to predict the novel view RGB map. The network is only supervised with an RGB loss.

A. *View Selection Strategy*

In order to select the best suited source views to create the target view, various schemes have been proposed. Most of them are based on spatial proximity and view direction [1], [3]. However, we observe that these methods tend to fail choosing the most informative views when images are taken with non-uniform spacing. As shown in Fig. 3, we may not be able to reconstruct the whole novel view for the target position (yellow dot) depending on scene geometry when choosing the right three views (blue dots) in case of large occlusions. Choosing based on proximity can force the network to extrapolate data from the source views while—ideally—we would want to network to interpolate the data in order to achieve smooth transitions and handle occlusions.

Therefore, we propose to choose the "working set" based on the Delaunay triangulation of the view positions. This ensures both a better coverage of the nearby view space and allows for an easy way to compute weightings for the views by using barycentric coordinates. Since we construct the triangulation in 2D while the camera positions are in 3D, we first need to determine which view configuration is present in the scene. We distinguish between two types: hemisphere sampling in which the views are placed in the upper hemisphere around the scene and fronto-parallel sampling where they are mostly planar in front of the scene.

For the case of hemisphere sampling, we stereographically project the camera positions onto the 2D plane where we perform the triangulation and then lift the result back to 3D. For fronto-parallel sampling, we orthographically project the camera positions onto the common plane defined by all views.

After triangulation, the closest triangle to the target view position is selected and the corresponding images form the working set. These three images are also assigned a weight b_i which corresponds to the barycentric coordinate of the target view w.r.t. the triangle. Fig. 3 (right) shows selected view positions with our approach on the previous example.

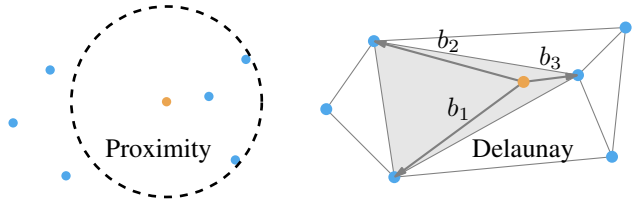

Fig. 3. Given a novel view (orange) we need to choose a working set from all the views in the dataset (blue). Proximity-based view selection can lead to significant occlusion as the working set only views the scene from the right. Our Delaunay-based approach selects the views that belong to the closest triangle, ensuring scene viewing from different sides.

Finally, we use a shared U-Net model to extract feature maps $\mathbf{F}_i \in \mathbb{R}^{H_i \times W_i \times d}$ for each image $\mathbf{I}_i$ in the working set.

To be noted that we only consider a working set of three images. A generalization to more images would require changing the view selection strategy and is left for future work.

B. *Geometry*

To recover the scene geometry, we shoot rays from each pixel of the target view and find the intersection of the ray with the scene surface. NeRF-like models accomplish this by densely sampling the ray at predefined intervals. Hence, most samples lay in empty space, slowing down processing. In contrast, we draw inspiration from Scene Representation Networks (SRN) [10] and propose to use a differentiable sphere tracer which predicts for each sample on the ray a jump towards the next sample. This effectively enables the network to learn and adapt the step-size, which greatly improves the sample efficiency and speed.

We parametrize each ray from the target view as follows:

$$r(t) = \mathbf{o} + t\mathbf{d}, \tag{1}$$

where t is the distance along the ray, $\mathbf{o}$ is the origin of the camera in world coordinates, and $\mathbf{d}$ is the normalized direction of the ray. We initialize t to be a small value such that ray marching starts close to the camera. At each ray marching

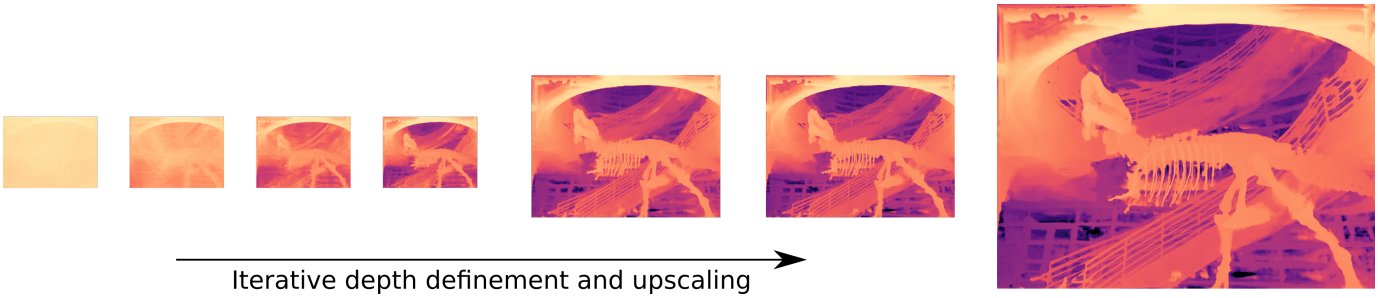

Fig. 4. Depth estimation for the novel view starts on a coarse scale and is initialized to a constant value close to the camera. The network iteratively refines the current estimate using input view features and upscales the depth until the full resolution is reached.

step, the position of the sample $\mathbf{x} = r(t)$ is obtained in world coordinates. The ray sample is projected into each source view $\mathbf{I}_i$ from where local features $\mathbf{f}_i \in \mathbb{R}^d$ are extracted using bilinear interpolation. The local features $\mathbf{f}_i$ from the source images are aggregated into a final feature by computing their weighted mean $\boldsymbol{\mu}$ and variance $\mathbf{v}$ using the barycentric weights.

The aggregated features are also concatenated with the positional encoding of the ray samples which helps the network to recover high-frequency depth. Hence, the aggregated feature for each ray is defined as:

$$\mathbf{g} = [\boldsymbol{\mu}, \mathbf{v}, \gamma(\mathbf{x})], \tag{2}$$

where $\gamma(.)$ is a positional encoding mapping the position into a higher-dimensional space [6].

Before computing the jump towards the next sample, an important consideration is that a single point sample does not contain sufficient information for an accurate jump prediction. Many real-world scenarios have objects with poor texture or ambiguous depth. If each ray sample independently predicts its own jumps, the final depth map will end up being noisy. We argue here that having knowledge of how the neighbouring rays behave is crucial for resolving ambiguities. Therefore, we add a series of 3×3 convolutions after the feature aggregation step in order to better constrain the features of each ray. To be noted that the per-pixel embeddings from U-Net capture only information from one particular view while convolving on the ray features also reasons about the multi-view features from the working set of images.

Finally, the ray features are passed through an LSTM that predicts the displacement δ along the ray which is used to update our depth $t_{i+1} = t_i + \delta$. This process is iterated a fixed number of times (we use 18 in our experiments) and the final ray sample is considered to be on the surface of the object. This is in stark contrast to NeRF-like models which require samples in the order of hundreds.

Since time consumption increases with the number of ray marching iterations and the number of rays we traverse, we propose to alleviate this problem by employing a coarse-to-fine scheme. Instead of creating rays for each pixel of the target view of size $H \times W$, we first ray march from a downsampled version at quarter resolution. After several ray marching steps, the computed depth map is upsampled

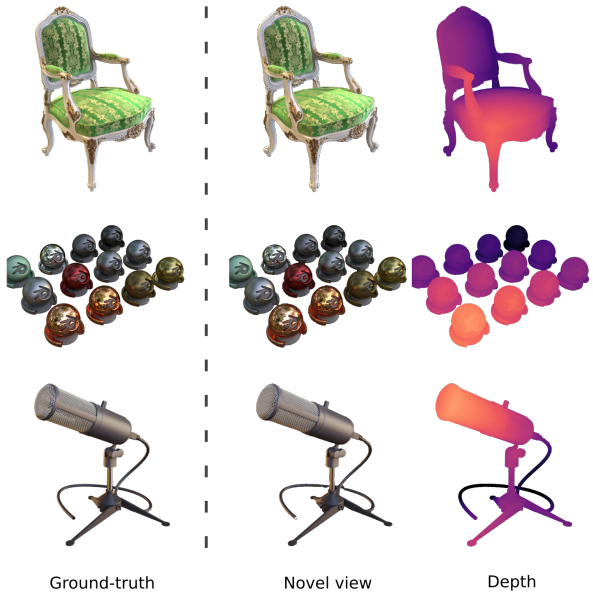

Fig. 5. Results from the Realistic Synthetic dataset [6]. Our network captures both high-frequency detail and view-dependent effects like specular reflections.

bilinearly to half resolution and the ray marching continues. This process, shown in Fig. 4, iterates until the final full resolution is reached. We observe that this scheme works well since locally-close pixels tend to march together and therefore their depth can be recovered by marching them as a whole. We use three levels of hierarchical depth, each with 10, 5, and 3 ray march steps, respectively.

C. Color

We obtain per-pixel color by projecting the final ray-marched surface into the three input views and bilinearly sampling both color $\mathbf{c}_i$ and local features $\mathbf{f}_i$ which are concatenated together in $\mathbf{k}_i = [\mathbf{c}_i, \mathbf{f}_i]$. Instead of aggregating $\mathbf{k}_i$ using the barycentric weights, we observed that it is beneficial to allow the network to predict the weights. Therefore, similar to IBRNet [8], we first compute $\boldsymbol{\mu}$ and $\mathbf{v}$ using the barycentric weights in order to capture global information. Afterwards, we concatenate these aggregated features with each per-frame feature vector $\mathbf{k}_i$. Each concatenated feature is fed into a small

Fig. 6. The confidence predicted by the network is low near depth discontinuities where occlusion occur and therefore errors are likely.

Fig. 7. Comparison of novel views on the test set of DTU. The model didn't use any per-scene optimization and was trained only on the training set of DTU showing that it can generalize to novel views and novel objects.

MLP to integrate both local and global information and predict multi-view aware feature $\mathbf{k}_i'$ and blending weights $w_i \in [0, 1]$. We pool $\mathbf{k}_i'$ into mean and variance by using the weights w_i and map the resulting vector to RGB color using another MLP. We denote the final RGB image with $\tilde{\mathbf{I}}$.

The color loss is computed as the ℓ_1-loss between the recovered RGB and the ground-truth color. This loss implicitly biases the geometry to lie on the true scene surface since the correct depth produces consistent input view features and the color prediction becomes possible. This allows the network to learn unsupervised depth and be applicable to datasets with only RGB images.

The reader should further note that we output a full RGB map in one pass of our network. In contrast, NeRF-like methods output a limited number of pixels at a time since their ray-marching step is more expensive and therefore requires to run the network multiple times to complete the full image. This allows our method to use more complex losses like perceptual losses which need to operate on the full image.

D. Loss with Confidence Estimation

Apart from predicting a correct novel view, it is also valuable to predict a confidence for each pixel. This allows to reason about possible occlusions or regions which are outside the frustum of the input views. In essence, the network can hallucinate detail when needed but it should be aware of this hallucination. The confidence is not used directly in our network but it is a useful output for downstream tasks like depth fusion.

In order to predict a confidence map, we draw inspiration from the work of Wagner *et al*. [14] which attempt to recover fine-grained explanations from classification networks. The input to their classification network is a pixel-wise blend between the image and a zero image. The loss function attempts to set as many pixels as possible to zero without affecting the classification accuracy. Hence, non-zero pixels are the ones that the network deems important for classification.

In our approach, we choose a similar scheme by defining our loss as a blend between the predicted $\tilde{\mathbf{I}}$ and the ground-truth image $\mathbf{I}$. The blend uses the confidence map $\mathbf{Q}$ which encourages it to be as close as possible to 1 such that most of the pixels are chosen from the predicted image. Then our image loss with confidence estimation is defined as:

$$L = \left\| \mathbf{I} - \left(\tilde{\mathbf{I}} \cdot \mathbf{Q} + \mathbf{I} \cdot (1 - \mathbf{Q}) \right) \right\|_1 + \lambda \left\| 1 - \mathbf{Q} \right\|_2 . \quad (3)$$

IV. Results

A. Datasets

We evaluate our method on three datasets. DTU [15] contains real images of various objects and is targeted towards evaluation of MVS methods. We use the train and test splits as defined by PixelNeRF [7]: 88 scenes for training and 15 for testing at a resolution of 400×300. We use this dataset to test the generalization capabilities of our method. The objects in the test set are different from the ones in the training set, so if the network is able to recover novel views of these novel objects, we can conclude that it learned a general reconstruction method. Results of the generalization to novel objects and novel views can be seen in Fig. 7.

Realistic Synthetic $360°$ [6] contains synthetic images of objects from the upper hemisphere. The dataset contains eight scenes with images at 800×800 resolution. The objects exhibit several view-dependent effects like specular reflections which must be captured correctly by the network for properly rendering the target view. Results of our network's prediction on this dataset can be seen in Fig. 5.

Real Forward-Facing [16] consists of real images of large scenes scanned with a camera in a forward-facing manner. The dataset contains eight scenes with image size of 1008×756. We use the train and test split as defined by [8]: every 8th image is selected for testing.

B. Evaluation

We train our method on the three datasets and distinguish between with and without per-scene optimization. In the case of no scene optimization, we train a generalizable model on the DTU dataset [15] and evaluate on the synthetic [6], a real dataset [16], and the novel scenes from DTU. Tab. I shows that our network generalizes to the novel views despite the drastic change in scale and object types. We also train our model with per-scene images similar to NeRF and show that it performs comparable to other generalizable models like MVSNeRF [12] while being significantly faster.

In Fig. 8 we compare our per-scene optimized model with the other baselines. We observe that our model can recover more detail especially in highly specular areas. However, our method also exhibits more errors near occlusion boundaries. This is to be expected as our method is imaged-based and therefore areas which are occluded in all source images cannot be reliably reconstructed.

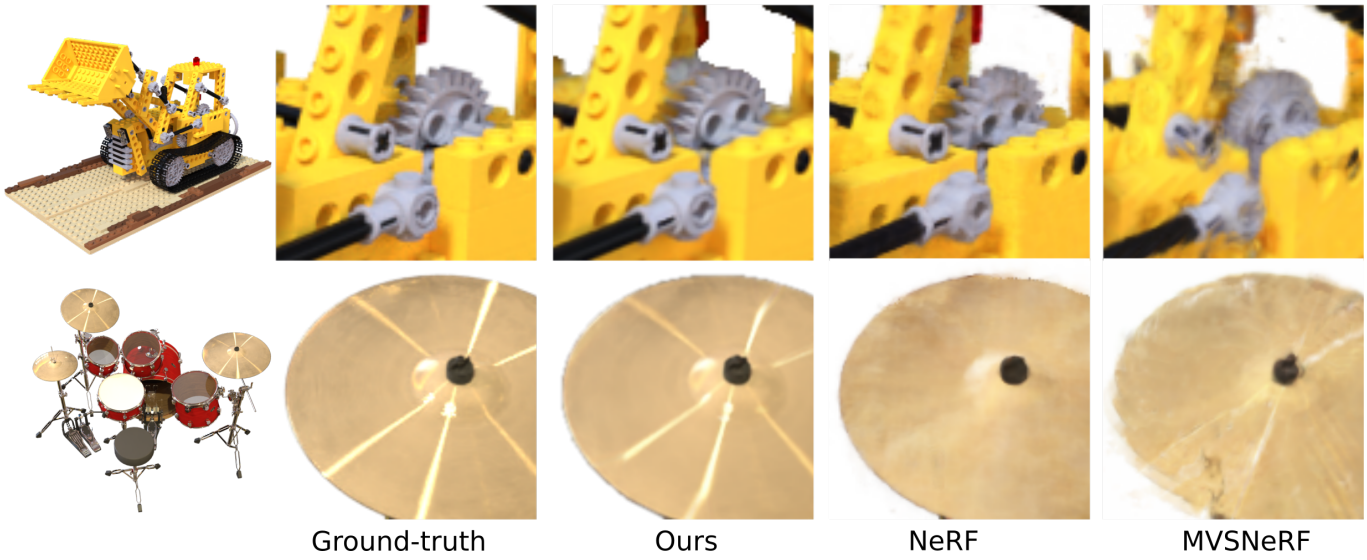

Fig. 8. Comparison of test-set views of the synthetic dataset. Our method can recover sharp detail together with view-dependent effects. However, because our method is image-based, it struggles with occlusions and thin objects like the drum stands.

TABLE I
COMPARISON OF DIFFERENT METHODS ON MULTIPLE REAL AND SYNTHETIC DATASETS. RESULTS WITH † CORRESPOND TO NERF MODEL TRAINED FOR 9.5H AS EVALUATED BY MVSNERF [12].

Method	Setting	DTU [15]			Realistic Synthetic 360° [6]			Real Forward-Facing [16]		
		PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
pixelNeRF [7]	No per-scene optimization	24.14	0.887	0.224	4.36	0.46	0.44	11.266	0.388	0.757
IBRNet [8]		25.84	0.902	0.213	19.43	0.841	0.231	16.70	0.566	0.498
MVSNeRF [12]		25.17	**0.911**	0.185	**22.67**	**0.90**	**0.21**	17.56	**0.691**	0.381
Ours		**26.376**	0.896	**0.184**	20.070	0.687	0.242	**18.909**	0.643	**0.372**
NeRF [6]	Per-scene optimization	23.70†	0.893†	0.247†	**31.01**	0.947	0.081	**26.50**	0.811	0.250
MVSNeRF [12]		**29.30**	**0.959**	**0.101**	27.21	0.945	0.227	26.25	**0.907**	**0.139**
Ours		28.093	0.913	0.165	28.425	**0.952**	**0.070**	25.206	0.803	0.218

C. Performance

Previous methods are unable to process the full image at once due to the high computational demand per ray and thus need to run several times with different ray batches to complete the image. We set the ray batches to the maximum size that fits in the memory of an NVIDIA GeForce RTX 3090 and measure the time for rendering a novel view with a resolution of 800×800 pixels. Tab. II shows that our method renders the full image in one forward pass and requires significantly less time than all previous approaches.

D. Ablation Study

We perform an ablation study of the different components of our network. We train on the Lego scene from the synthetic dataset and observe how the network performance is affected.

We first remove the positional encoding from the ray marching. This decreases the depth quality significantly as the network is unable to recover high-frequency details.

Disabling Delaunay for view selection and using a proximity-based method that takes the three closest frames as the working set shows also a slight decrease in performance.

TABLE II
COMPUTATION TIME AND MAXIMUM RAYS PER BATCH FOR A 800×800 IMAGE.

Method	Time	Rays
MVSNeRF [12]	5.2 s	110 k
NeRF [6]	6.4 s	120 k
IBRNet [8]	31 s	8 k
pixelNeRF [7]	164 s	300 k
Ours	**0.16 s**	**full(640 k)**

TABLE III
ABLATION STUDY

Setting	PSNR↑	SSIM↑	LPIPS↓
No position encoding	25.692	0.899	0.082
No Delaunay	26.764	0.919	0.075
Fewer ray marches	27.522	0.933	0.058
Only 1×1 convolutions	26.224	0.912	0.075
Complete model	**27.918**	**0.937**	**0.056**

Fig. 9. Failure cases: Our method fails in the case of occlusion like the ceiling detail or the image borders.

By default, we use three levels of depth refinement, each with 10, 5, and 3 ray march steps, respectively. We reduce the number of ray march steps to 5, 3, and 1 and observe a slight decrease in performance. The required number of ray marching steps is heavily dependent on the scene complexity with simple scene requiring less steps.

Finally, we change the 3×3 convolutions in the ray marcher to 1×1 in order to simulate propagating each ray independently with no spatial awareness of the neighbouring rays, similar to other NVS methods. We observe a significant decrease in accuracy, as the rays can no longer leverage spatial information to resolve ambiguities.

E. Implementation Details

The feature extraction network is a U-Net model [17] which outputs per-pixel a 64 dimensional vector. The features from the three images are aggregated and then passed through two convolutional layers of 3×3 which output 64 channels. Finally, the LSTM that predicts the sample jump has a hidden size of 32. The color estimation is implemented as an MLP with 3 layers and a hidden size of 64. The network is optimized using Adam [18] with a learning rate of 1×10^{-4}.

F. Limitations

One limitation of our method is that it is based on ray-marching instead of volumetric rendering and therefore cannot model transparent objects. A switch to a front-to-back additive blending of radiance could alleviate this issue.

Another limitation is that our method is image-based and therefore cannot recover detail in occluded regions as seen in Fig. 9.

Finally, the depth can be ambiguous in the case of no texture since the network can recover correct color even if the depth is noisy. This could be alleviated by using more input views or with stronger priors.

V. Conclusion

We proposed a network that jointly resolves scene geometry and novel view synthesis from multi-view datasets and is supervised only by image reconstruction loss. We represent the scene geometry as a distance function which we ray march using sphere tracing. Sphere tracing alleviates the memory constraints faced by other methods and allows us to render high resolution images in one forward pass and is thus much faster than previous methods. We further improve the speed by proposing a hierarchical depth refinement which estimates depth in a coarse-to-fine manner.

Finally, we show the generalization capabilities of our network by evaluating on datasets with different scale and object configurations for which we obtain competitive results but with significantly higher frame rates.

References

[1] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, "Pixelwise view selection for unstructured multi-view stereo," in *European Conference on Computer Vision (ECCV)*, 2016.

[2] S. Galliani, K. Lasinger, and K. Schindler, "Massively parallel multiview stereopsis by surface normal diffusion," in *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[3] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz, "Multi-view stereo for community photo collections," in *IEEE 11th International Conference on Computer Vision (ICCV)*, 2007.

[4] H. Laga, "A survey on deep learning architectures for image-based depth reconstruction," *arXiv preprint arXiv:1906.06113*, 2019.

[5] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan, "MVSNet: Depth inference for unstructured multi-view stereo," in *European Conference on Computer Vision (ECCV)*, 2018, pp. 767–783.

[6] 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 (ECCV)*, 2020.

[7] A. Yu, V. Ye, M. Tancik, and A. Kanazawa, "pixelNeRF: Neural radiance fields from one or few images," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[8] Q. Wang, Z. Wang, K. Genova, P. Srinivasan, H. Zhou, J. T. Barron, R. Martin-Brualla, N. Snavely, and T. Funkhouser, "IBRNet: Learning multi-view image-based rendering," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[9] F. Darmon, B. Bascle, J. Devaux, P. Monasse, and M. Aubry, "Deep multi-view stereo gone wild," *International Conference on 3D Vision*, 2021.

[10] V. Sitzmann, M. Zollhöfer, and G. Wetzstein, "Scene representation networks: Continuous 3D-structure-aware neural scene representations," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[11] A. Rochow, M. Schwarz, M. Weinmann, and S. Behnke, "FaDIV-Syn: Fast depth-independent view synthesis," in *Proceedings of Robotics: Science and Systems (RSS)*, 2022.

[12] A. Chen, Z. Xu, F. Zhao, X. Zhang, F. Xiang, J. Yu, and H. Su, "MVSNeRF: Fast generalizable radiance field reconstruction from multi-view stereo," *arXiv:2103.15595*, 2021.

[13] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar, "Local light field fusion: Practical view synthesis with prescriptive sampling guidelines," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–14, 2019.

[14] J. Wagner, J. M. Köhler, T. Gindele, L. Hetzel, J. T. Wiedemer, and S. Behnke, "Interpretable and fine-grained visual explanations for convolutional neural networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9097–9107.

[15] H. Aanæs, R. R. Jensen, G. Vogiatzis, E. Tola, and A. B. Dahl, "Large-scale data for multiple-view stereopsis," *International Journal of Computer Vision*, vol. 120, pp. 153–168, 2016.

[16] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar, "Local light field fusion: Practical view synthesis with prescriptive sampling guidelines," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 29:1–29:14, 2019.

[17] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.