Foundations of 3D Vision with Emphasis on Neural Radiance Fields (NeRF)

Name: Saharsh Sandeep Barve
Netid: ssbarve2
Subject: CS597-DWH, Independent Study, UIUC, Spring 2024.

0) Acknowledgement

I would like to express my gratitude to Prof. Derek Hoiem and Yuqun Wu for their guidance and support throughout this study.

1) Introduction

The field of computer vision has seen tremendous progress in the domain of three dimensional (3D) scene representation, more commonly known as 3D vision. Our study focuses on a subset of this field, specifically on the task of converting a set of images, which represent a particular scene, into an accurate 3D representation like point clouds, meshes, voxel grids. The utility of 3D vision extends across several applications, such as augmented reality, robotics, medical imaging, industrial automation and construction management. These applications require both accurate geometry and realistic visualization of the scene, and may involve large scenes with sparse input views, which we further explore in our study.

At its core, estimating 3D geometry using images relies on fundamental camera, vision, and mathematical concepts. With the widespread adoption of deep learning models, even 3D vision has seen recent progress by leveraging neural networks, notably Neural Radiance Fields (NeRF), alongside these fundamental concepts.

The primary objective of this study is to explore techniques to refine the geometry of 3D scene representations from NeRF. Specifically, we explore the potential of Neural Kernel Surface Reconstruction (NKSR) as a complementary technique to improve NeRF results. We also discuss and share insights into the sensitivity of this approach to the presence of noise, highlighting the challenges encountered in this endeavor.

The rest of the report is structured as follows: To comprehensively understand the significance of our project, it is crucial to first delve into the fundamental concepts like Structure from Motion (SfM), Multi-View Stereo (MVS), and NeRF. These concepts, outlined in the background section

(2), serve as the pillars for our investigations into recent advancements in 3D scene representation, as detailed in section 3. In this section, we dive into specific models and techniques, including the Tiny NeRF Model (3.1), MonoPatch NeRF (3.2), and NKSR (3.3), which build upon the foundational principles.

2) Background

2.1 Structure From Motion

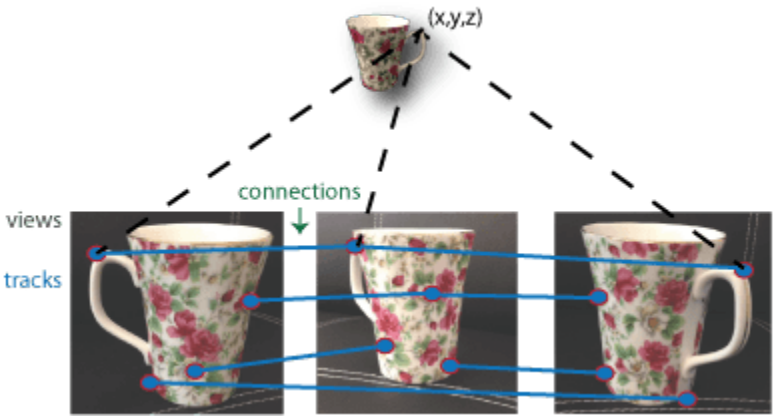


Figure 1: Structure from Motion
Image source: Mathworks [link]

The structure from motion (SfM) algorithm aims to reconstruct the sparse geometry of a scene (finding the 3D location of a point present in different images) and the motion (finding poses) of the camera simultaneously, using a set of images taken from different viewpoints as shown in Figure 1 [1]. It relies upon the epipolar constraint, which defines the relationship between two camera views of a scene (Figure 2). Specifically, the constraint helps reduce the search space because for a given 3D point X and its projections x and x' in two views, the projection x' lies on an epipolar line determined by the point x. Mathematically this constraint gets defined using an essential matrix (E), for calibrated camera case or fundamental matrix (F) for uncalibrated camera cases, as per the given formulas (1) and (2), which are estimated using a normalized 8 point algorithm [2].

$$x'^{T}Ex = 0 \quad (1)$$
$$x'^{T}Fx = 0 \quad (2)$$

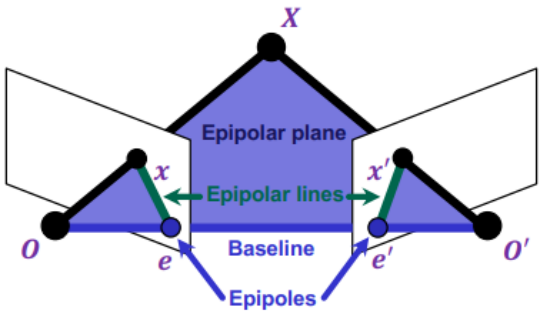


Figure 2: Epipolar geometry
Image source: Prof. Svetlana Lazebnik UIUC CS543 lecture 17 [3] [link]

Modern SfM pipelines, designed for large scale scenes, (also available in 3D vision software tools like COLMAP) often utilize an incremental SfM approach as shown in Figure 3. After extracting key features (using methods like SIFT or ORB detectors) across all the images the next step involves generating 2D tracks from these matches as per the epipolar constraint. Subsequently, both the structure and motion models are initialized using “good” image pair to begin with and further refined by incorporating additional views and refining the camera parameters and 3D points [4].

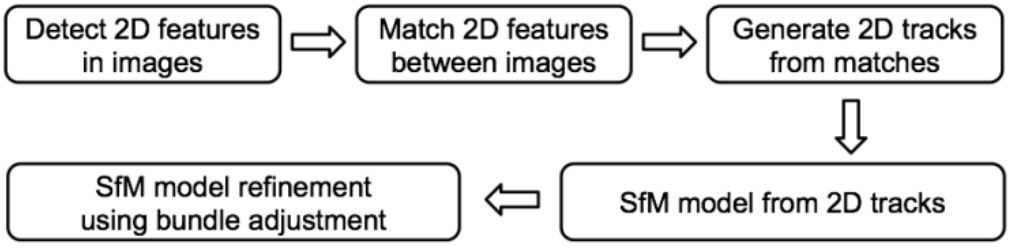


Figure 3: Modern SfM pipeline
Image source: MVS Tutorial by Prof. Yasutaka [link]

2.2 Multi View Stereo

Multi View Stereo (MVS) builds upon SfM pipeline to generate dense 3D scene representation using multiple images covering diverse viewpoints. It begins by computing correspondences (similar to how it's done in SfM) followed by plane sweep stereo to estimate depth of each pixel in the images. This step involves sweeping a set of depth planes through the scene and selecting the depth plane with the best match as shown in Figure 4.

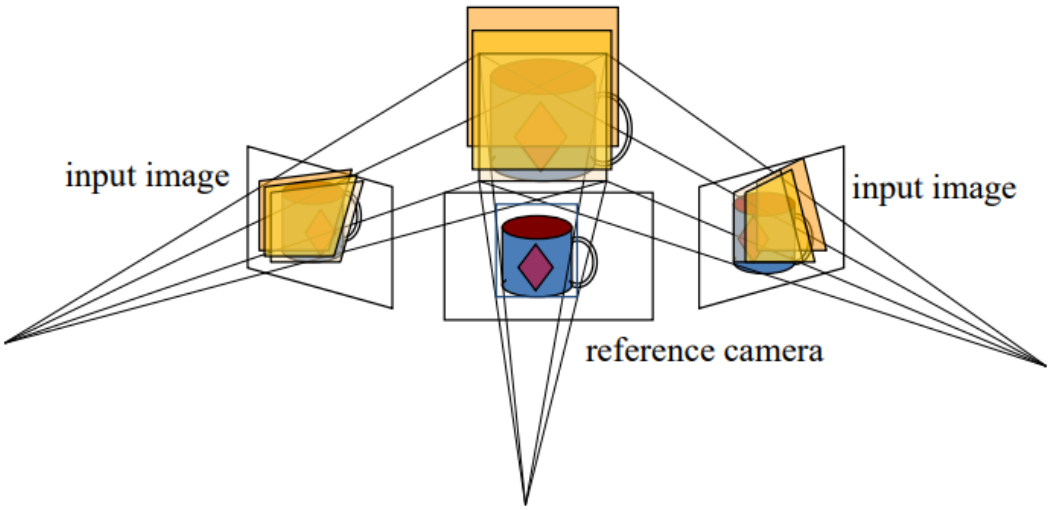


Figure 4: Plane sweep stereo intuition
Image source: Prof. Svetlana Lazebnik UIUC CS543 lecture 20 [link]

Additionally, similarity measures (like Sum of squared distances (SSD), Normalised Cross Correlation (NCC)) are also used to compute photo-consistency (*can refer to Chapter 2 MVS Tutorial* [2] *for an in-depth explanation*). Finally, the depth maps from each image are fused into a single coherent 3D representation like meshes or point clouds.

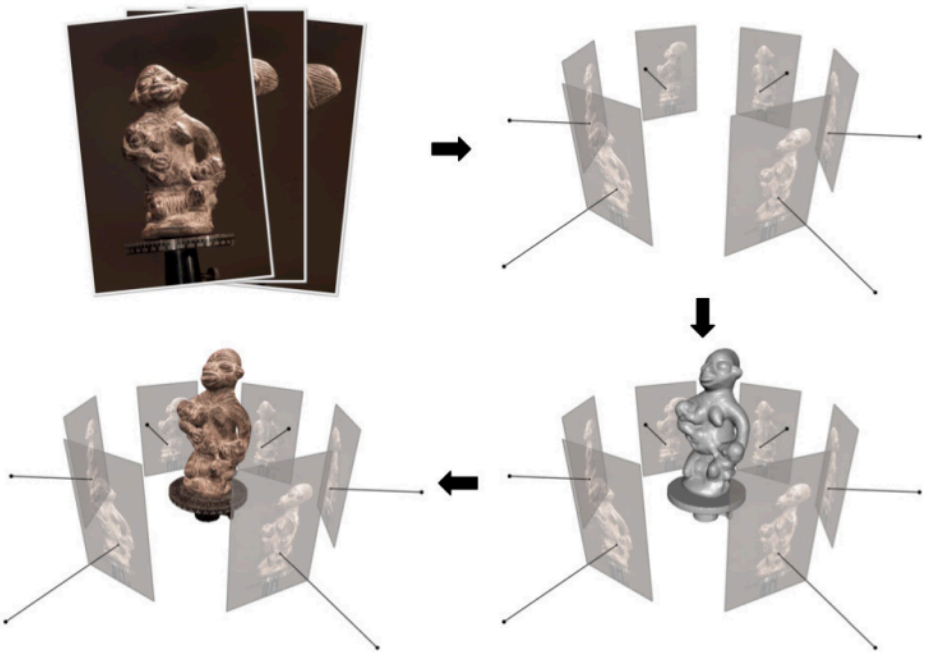


Figure 5: Multi View Stereo algorithm pipeline
Image source: MVS Tutorial by Prof. Yasutaka [link] (found from Prof. Derek's slides)

2.3 Neural Radiance Fields (NeRF)

Mildenhall et. al proposed an approach of leveraging multi-layer perceptron (MLP) to parameterize the radiance field function (f_θ) (3) and quantify the radiance (in simpler terms means intensity and color) of light rays traveling from a 3D point in space towards the camera. [5]. In essence, this function maps 3D location $x \in R^3$ and viewing direction $d \in S^2$ to volume density $\sigma \in [0, \infty)$ and color $c \in [0, 1]^3$ [6].

$$f_\theta : R^{L_x} X R^{L_d} \rightarrow [0, 1]^3 X [0, \infty) \quad (3)$$

$$\gamma(x),\ \gamma(d) \mapsto (c, \sigma) \quad (4)$$

In the given formulas (3) and (4), θ indicates network weights, γ defines positional encoding applied to input x and d.

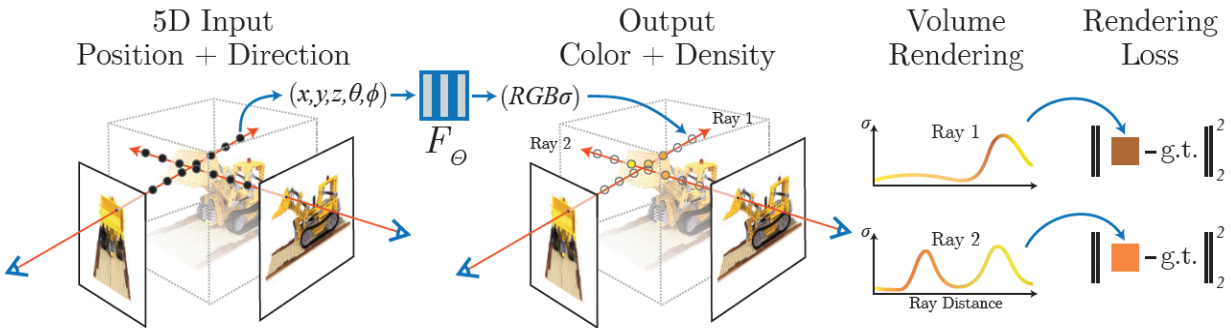


Figure 6: Neural Radiance field pipeline
Image source: Mildenhall et. al NeRF paper [link]

In the classical NeRF pipeline, illustrated in Figure 6, the process begins by sampling points along camera rays. Each point comprises 3D coordinates and 2 direction components. These inputs are then mapped to a higher-dimensional space using positional encoding. Next, these inputs are fed to a MLP (8 Fully-Connected (FC) layers with ReLU activation and 256 channels) which then produces volume density and color. As this model (rendering function) is naturally differentiable, gradient descent optimization is used to reduce the error between input images of the scenes and corresponding rendered views.

Volume Rendering:

Volume rendering is utilized to differentiably render out novel views using the radiance function described above. To render out a pixel using the parameterized radiance field function, a ray $r(t) = o + td$ is cast from camera center o through the pixel, with direction d. The pixel's color value $\hat{C}_\theta(r)$ is computed using alpha composting as per the equation below [5,6].

$$\hat{C}_\theta(r) = \int_{tn}^{t_f} T(t)\sigma_\theta(r(t))c_\theta(r(t), d)dt \text{ where } T(t) = exp(-\int_{tn}^{t}\sigma(r(s))ds) \quad (5)$$

In this equation of computing pixel color value, $T(t)$ denotes accumulated transmittance along the ray, (which in simpler terms means the probability the ray travels from t_n to t_f without hitting another particle) [5,6].

3) Investigations

3.1 Tiny NeRF Model

In this experiment, we go through the Tiny NeRF pytorch implementation and reproduce the results from [7] to gain a better understanding of the underlying concepts of neural radiance fields and their code level implementation. We utilize the lego toy dataset [8] comprising 106 inward facing images of a lego toy bulldozer with an image size of 100 x 100. The hyper parameters utilized are described in the Table 1 below.

Table1: Tiny NeRF hyperparameters

Hyper Parameter	Value
1. Encoding functions	6
2. Sampling rate along each ray	32
3. Optimizer params: (learning rate, iterations)	$(5e^{-3}, 1000)$

The NeRF model comprises 3 FC layers each followed by a ReLU activation function with feature size of 128 channels instead of the usual 256. Figure 7 visualizes the rendered outputs of the scene alongside corresponding PSNR (measures quality of image) values.

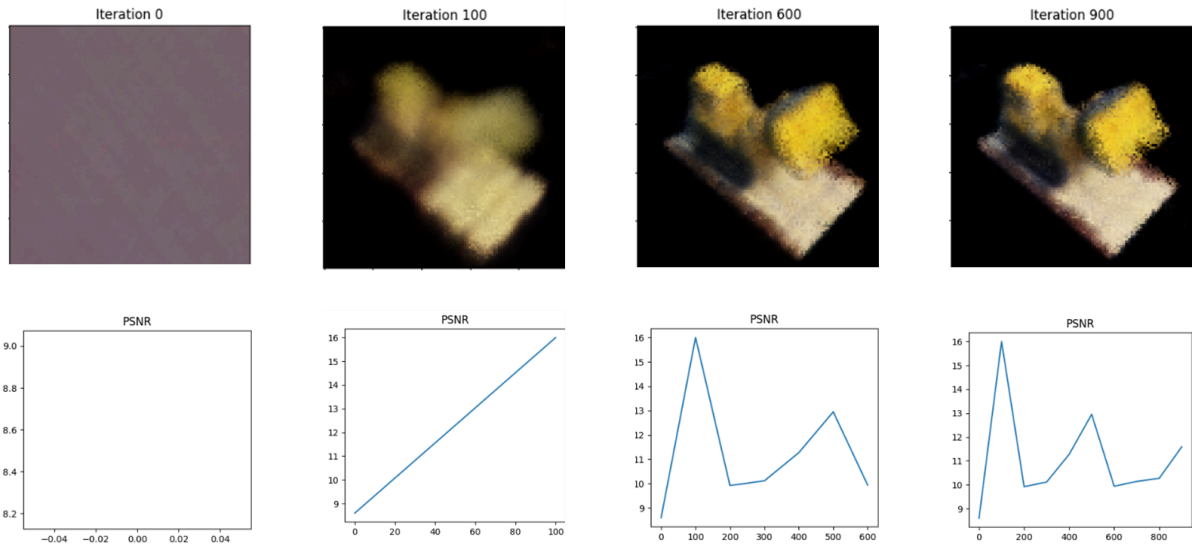


Figure 7: TinyNeRF results on Lego images

3.2 MonoPatch NeRF

In this experiment, we explore the implementation of the MonoPatchNeRF model proposed by Yuqun et al. MonoPatchNeRF is a novel approach aimed at constructing 3D scene representations with precise geometry and enabling view synthesis, drawing inspiration from the Neural Radiance Fields (NeRF) and MVS. It employs a patch-based methodology, utilizing monocular surface normals and monocular relative depths to achieve its objectives [9].

The patch-based approach ensures a more effective integration of monocular information, including depth and normals, than ray-based sampling. Using patches is effective because monocular geometry is more accurate locally up to a translation and scale. Moreover, this approach is also suitable for encoding large scenes using density models. Finally it also allows utilization of structural losses such as Normalized Cross-Correlation (NCC) and Structural Similarity Index Measure (SSIM) which provide robustness to lightning conditions and appearance regularization. For generating normals, a normal-rendering-MLP head is attached onto the NeRF generating surface normals as negative direction gradient of optical density.

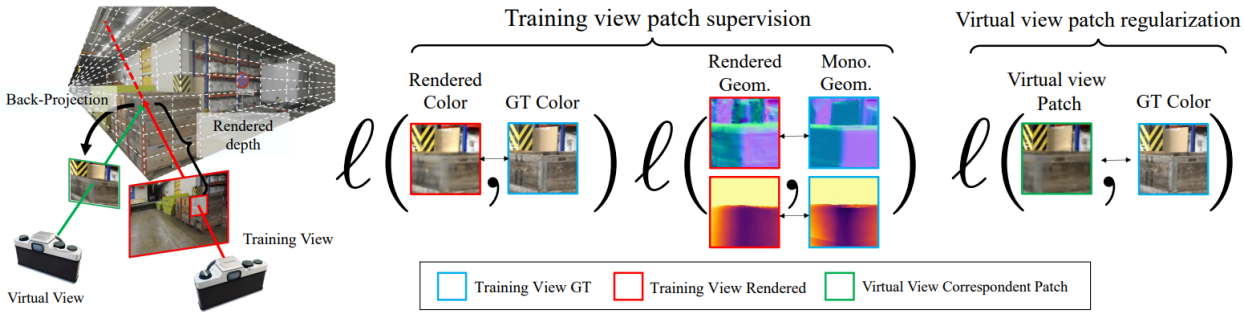


Figure 8: MonoPatchNeRF framework
Image source: monoPatchNeRF paper [9] [link]

MonoPatchNeRF employs a combination of losses for model optimization, as shown in Figure 8, each serving an important purpose in achieving better results. Firstly, the color supervision of RGB utilizes pixel level Huber loss to supervise the rendered colors. Secondly, the geometric supervision of depth and normals is used to ensure better geometry of the 3D scene rendering. Lastly, the virtual view patch regularization is used, computing loss between a randomly sampled patch and the ground truth color using structural losses discussed earlier.

For our experiments, we utilize Tanks&Temples (TnT) dataset [10], which is a widely used benchmark dataset in the domain of 3D vision. Specifically, for this report, we present our findings on the Barn scene of the TnT dataset, which comprises 410 images of a barn housing structure.

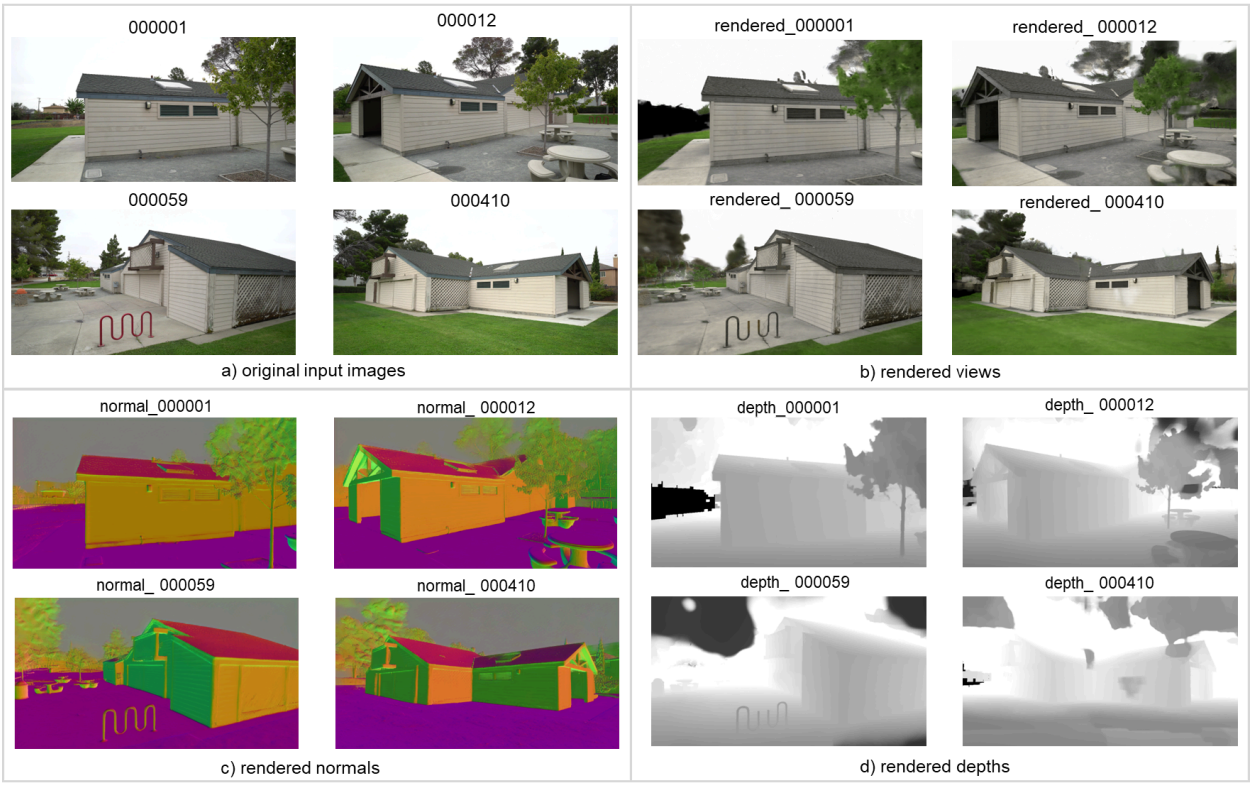


Figure 9: MonoPatchNeRF results on Barn dataset

In Figure 9, we present the results from fitting MonoPatchNeRF onto the barn dataset. 9(a) shows 4 input images of the barn, taken from various angles. 9(b) displays the rendered outputs of the barn from corresponding viewpoints. In 9(c), the scene visualizes view normals generated using the normal-rendered MLP head of MonoPatchNeRF. Finally, 9(d) presents a depth map of the corresponding scenes.

3.3 Neural Kernel Surface Reconstruction (NKSR)

Neural Kernel Surface Reconstruction (NKSR) provides an approach to map a large scale, spare, noisy point cloud into a 3D surface (like a mesh) [11]. It serves as the next iteration of Neural Kernel Fields (NKF) [12] and overcomes its limitation of working only on a relatively small scale scene (~10k points) and its susceptibility to noisy input. In its working, it learns a data dependent kernel and predicts a continuous occupancy field as a linear combination of this kernel supported on input points [11]. This ensures better generalizability. Figure 10, further expands the NKSR pipeline wherein we feed input points and normals to a sparse convolutional neural network (CNN) and it predicts voxel hierarchies which are then used to extract a resultant mesh.

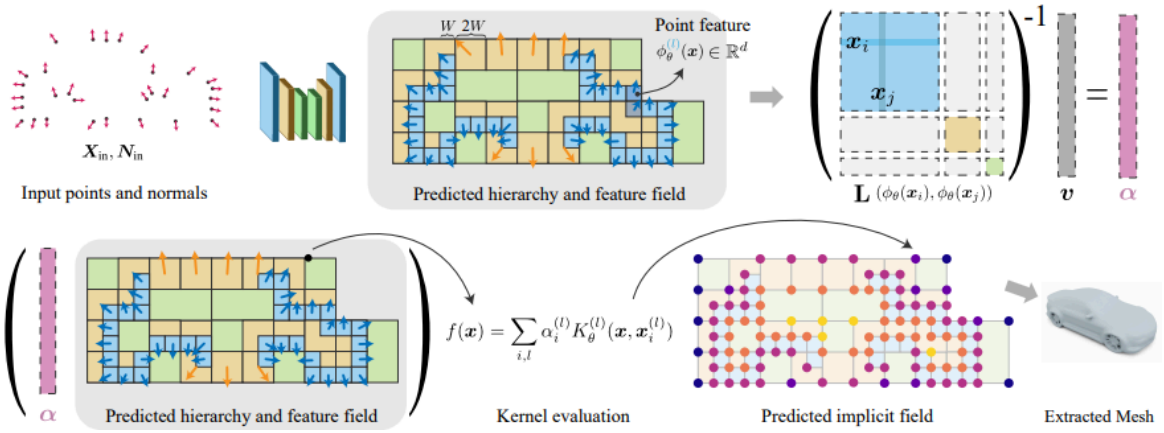


Figure 10: NKSR pipeline
Image source: NKSR paper [link]

In Figure 11, we present the complete pipeline outlining the steps involved in the 3D reconstruction process using MonoPatchNeRF, ColmapFusion and NKSR. The process begins by using the MonoPatchNeRF model to generate depths and normals as discussed previously, followed by the ColmapFusion module. In this step, we establish image pairs with overlapping points, convert coordinates to world coordinates, filter out unwanted points, determine points to fuse based on calculated disparity and threshold, and finally apply fusion between image pairs while retaining the back projection of fused points. Thus, we end up with a 3D point cloud and point normals which serve as our input to NKSR as discussed above. Finally, we utilize NKSRs default setup (while downsampling our points and normals by a factor of 2 and sampling 10 million points to meet our hardware constraints) to generate the mesh result.

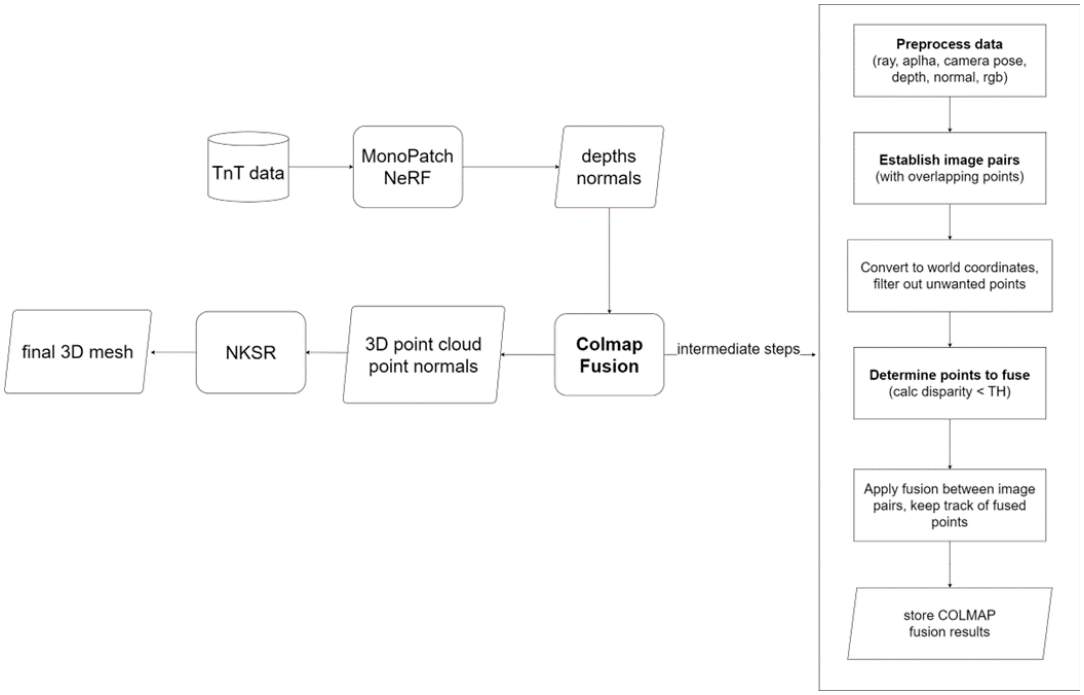
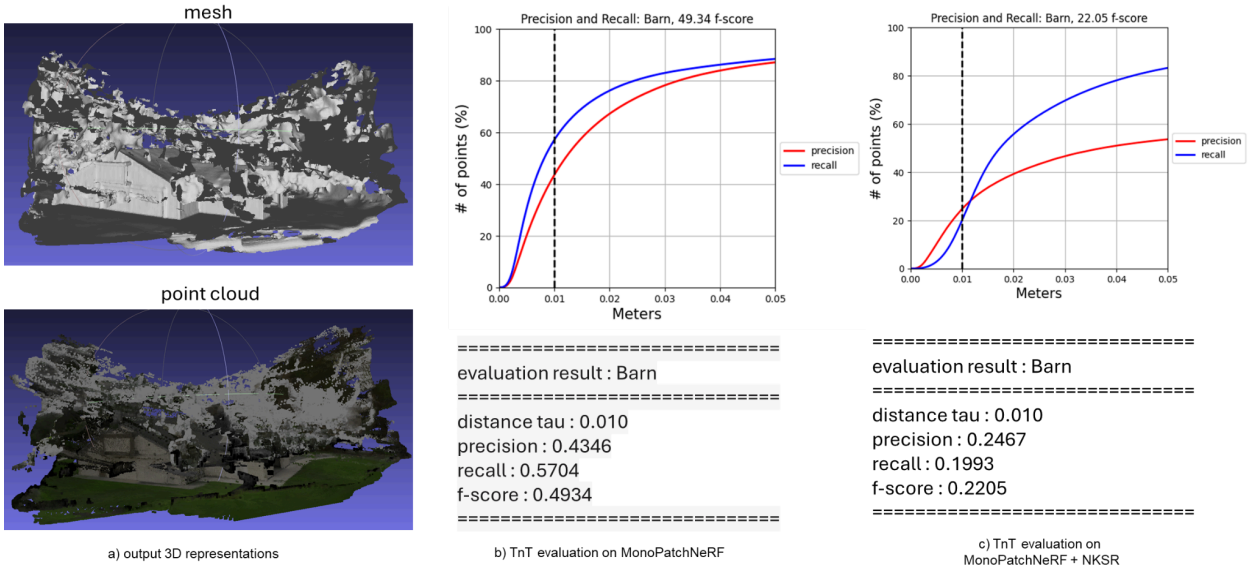


Figure 11: Complete workflow of our (MonopatchNeRF + NKSR mesh generation) investigation

Figure 12 illustrates the results of our 3D reconstruction using the MonoPatchNeRF + NKSR pipeline on TnT evaluation. In 12(a), we display the resultant output mesh from NKSR and the point cloud generated from the ColmapFusion step. Figure 12(b) presents the results of utilizing only MonoPatchNeRF, while Figure 12(c) displays the results when MonoPatchNeRF is combined with NKSR. These results suggest that incorporating NKSR leads to poorer performance compared to the original pipeline. The drop in precision could be attributed to noisy sampling from the resultant mesh.



a) output 3D representations

b) TnT evaluation on MonoPatchNeRF

c) TnT evaluation on MonoPatchNeRF + NKSR

Figure 12: NKSR results on Barn dataset

Even qualitatively, the final mesh generated using NKSR does show signs of being sensitive to noise, indicated by the presence of artifact regions in the output. We further investigated this sensitivity through supplementary experiments, as demonstrated in Figure 13. In Figure 13(a), we attempted to address this limitation by incorporating voxel-based radius nearest neighbor noise removal, which resulted in sparse point clouds (as seen by the empty areas on the ground of the scene) and also led to a deterioration in our F1-score on the TnT evaluation. In comparison to Poisson surface reconstruction [15], our approach appears to perform relatively better in this regard (Figure 13(b)).

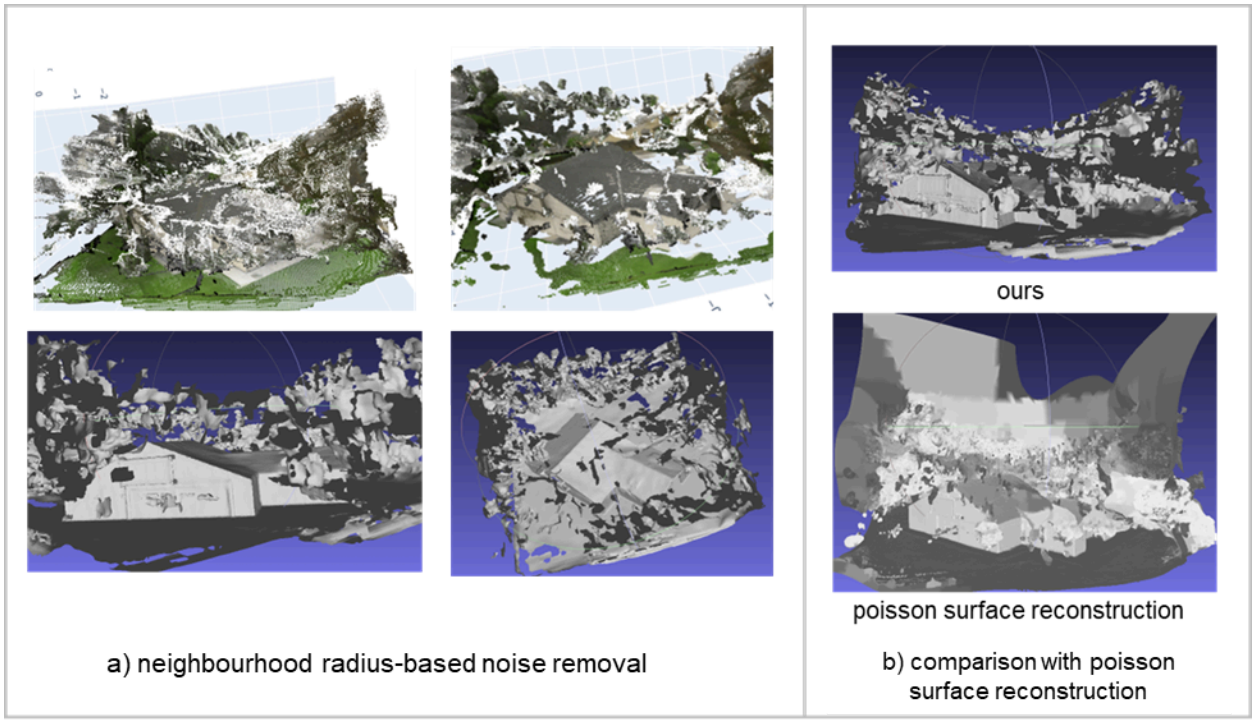


Figure 13 illustrates the sensitivity of our pipeline to noise, showcasing the resulting outcomes.

As a further extension of this study, with a goal of improving 3D scene geometry, we are currently exploring how feeding novel views generated by NeRFs, along with original images, into MVS would bring improvements to the multi-view stereo results.

4) Additional Learning Outcomes

Additional learning outcomes from the study covered a wide array of topics and experiences, including practical skills like utilizing Python debugger (for lots and lots of debugging), configuring NVIDIA CUDA toolkit for GPU acceleration, and running training on GPU clusters. Moreover, the study also helped me delve into fundamental principles of low level system design and coding. The discussions led by Prof. Derek, Prof. Shenlong, PhD. Chuhang and my mentor Yuqun helped me understand how concepts get dissected and scrutinized, and decisions assessed for their applicability in various scenarios while exploring research areas. I also got to see first hand how experiment planning and logging happens. A significant takeaway from the discussions for me was the treatment of ML models and pillar research-works as unit level abstractions. This perspective helped me understand that I must continue working in this domain to attain a deeper level of understanding and knowledge.

Additionally, the following literature review was performed at the beginning of the study to create a foundational understanding of relevant concepts:

1. SfM & MVS: PhotoTourism, MVS - Tutorial
2. NeRF: NeRF, mip-NeRF, Reg-NeRF, FreeNeRF, MonoPatchNeRF
3. SDF: MonoSDF
4. Others: Neuralangelo, NKSR

5) Conclusion

In conclusion, this study has provided valuable insights and first hand experience in the field of 3D vision. The foundational concepts learnt in MVS, SfM still prove to be crucial in 3D scene representation and these concepts which rely on epipolar and camera geometry and photo-consistency form the backbone of modern approaches.

The integration of deep learning based approaches as demonstrated through NeRF and our Tiny NeRF experiments showcase the use of traditional techniques integrated with modern deep learning based implementations.

Our exploration of the MonoPatchNeRF + NKSR pipeline (Figure 11) further highlights the relevance of these foundational concepts and provides decent results while combining traditional concepts alongside the more recent, deep learning based, NeRF implementations.

6) References

[1] Noah Snavely, Steven M. Seitz, and Richard Szeliski. 2006. Photo tourism: exploring photo collections in 3D. ACM Trans. Graph. 25, 3 (July 2006), 835–846. https://doi.org/10.1145/1141911.1141964

[2] Y. Furukawa and C. Hernández . Multi-View Stereo: A Tutorial. Foundations and TrendsR in Computer Graphics and Vision, vol. 9, no. 1-2, pp. 1–148, 2013.

[3] Forsyth, David. "UIUC CS543 Lectures." University of Illinois at Urbana-Champaign, Department of Computer Science, Fall 2023.

[4] Lazebnik, Svetlana. "UIUC CS543 Slides." University of Illinois at Urbana-Champaign, Department of Computer Science, Fall 2022.

[5] Ben Mildenhall and Pratul P. Srinivasan and Matthew Tancik and Jonathan T. Barron and Ravi Ramamoorthi and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. ECCV, 2020

[6] Michael Niemeyer, Jonathan T. Barron , Ben Mildenhall , Mehdi S. M. Sajjadi , Andreas Geiger, Noha Radwan. RegNeRF: Regularizing Neural Radiance Fields for View Synthesis from Sparse Inputs, CoRR, abs/2112.00724, 2021, https://arxiv.org/abs/2112.00724

[7] Murthy, Krishna. "Tiny NeRF PyTorch Implementation." GitHub repository, https://github.com/krrish94/nerf-pytorch.

[8] Tiny NeRF data, Lego Toy dataset,
http://cseweb.ucsd.edu/~viscomp/projects/LF/papers/ECCV20/nerf/tiny_nerf_data.npz

[9] Yuqun Wu and Jae Yong Lee and Chuhang Zou and Shenlong Wang and Derek Hoiem. MonoPatchNeRF: Improving Neural Radiance Fields with Patch-based Monocular Guidance, arXiv, 2024.
[10] Tanks & Temples dataset. Available at: https://tanksandtemples.org/
[11] J. Huang, Z. Gojcic, M. Atzmon, O. Litany, S. Fidler, and F. Williams. "Neural Kernel Surface Reconstruction." in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 4369–4379. https://research.nvidia.com/labs/toronto-ai/NKSR/
[12] F. Williams, Z. Gojcic, S. Khamis, D. Zorin, J. Bruna, S. Fidler, and O. Litany. "Neural Fields as Learnable Kernels for 3D Reconstruction." arXiv preprint arXiv:2111.13674, 2021. https://research.nvidia.com/labs/toronto-ai/nkf/
[13] Hoiem, Derek. “UIUC CS598 - 3D Vision Slides & CS441 Lectures.” University of Illinois at Urbana-Champaign, Department of Computer Science, Fall 2021, Spring 2024.
[14] T. Müller, A. Evans, C. Schied, and A. Keller. InstantNGP: "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding." ACM Trans. Graph., vol. 41, no. 4, Jul. 2022, pp. 102:1–102:15. [Online]. Available: https://doi.org/10.1145/3528223.3530127
[15] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. 2006. Poisson surface reconstruction. In Proceedings of the fourth Eurographics symposium on Geometry processing (SGP '06). Eurographics Association, Goslar, DEU, 61–70.