MONOCULAR RECONSTRUCTION DURING COLONOSCOPY AND ENGINE INSPECTION

A Dissertation
Presented to
The Academic Faculty

By

Ishan Chadha

In Partial Fulfillment
of the Requirements for the Degree
Masters of Science in the
Georgia Institute of Technology
College of Computing

Georgia Institute of Technology

July 2024

MONOCULAR RECONSTRUCTION DURING COLONOSCOPY AND ENGINE INSPECTION

Thesis committee:

Dr. Yue Chen
Biomedical Engineering
Georgia Institute of Technology

Dr. Alexis Noel
Georgia Tech Research Institute
Georgia Institute of Technology

Dr. Seth Hutchinson
College of Computing
Georgia Institute of Technology

Date approved: July, 2024

ACKNOWLEDGMENTS

I would like to profusely thank the members of my thesis committee for guiding me through this work. Dr. Yue Chen pushed me to find the intersection of my interests with challenging research questions, and Dr. Seth Hutchinson has helped me ponder the more philosophical side of these ideas. Dr. Alexis Noel asked me to think more broadly in parallel research areas, and Dr. James Hays gave me advice pertaining to how to formulate a meaningful research problem in the field of computer vision - a field which I am new to. Dr. David Hu oversaw much of my work and gave me direction in the realm of borescope inspection.

I would especially like to thank the PhD students at Georgia Tech's Biomedical Mechatronics Lab who have also given me guidance, including but not limited to Yifan Wang, Jia Shen, and Milad Azizkhani. The resources that the lab has provided me have allowed me to complete a research project with this level of impact.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

3DGS 3D Gaussian Splatting

4DGS 4D Gaussian Splatting

BRDF Bidirectional Reflectance Distribution Function

FF-MLP Fully-fused MLPs

GS-SLAM Simultaneous Localization and Mapping using Gassian Splatting

I-NGP Instant Neural Graphics Primitives

LE-GS Light Encoded Gaussian Splatting

LE-NGP Light Encoded Neural Graphics Primitives

LightNeuS Neural Implicit Surfaces using Illumination Decline

MLP Multi-layer Peceptron

NeRF Neural Radiance Fields

NeuS Neural Implicit Surfaces

PLS Point Light Source

SDF Signed-Distance Function

SfM Structure from Motion

SLAM Simultaneous Localization and Mapping

SUMMARY

Monocular reconstruction of surfaces has become possible in recent years with the advent of efficient and accurate algorithms like Neural Radiance Fields (NeRF) and 3D Gaussian Splatting (3DGS), but reconstruction can be difficult to perform in real-time depending on the nature of the environment and the precision required. Alongside this, acquiring ground-truth data in hard-to-reach environments poses additional challenges for validating these algorithms. Prior work has tackled this problem using generated datasets; this paper proposes validating this in silico and experimentally. This is accomplished by testing and optimizing state-of-the-art monocular reconstruction algorithms for endoscopy, specifically targeting colorectal endoscopy with the auxiliary application of borescope engine inspection. Proposed algorithms build off of 3DGS and its dynamic extension, EndoGaussian, for storing temporal deformation. Temporal and spatial features are stored in a compressed yet explainable representation called a HexPlane. The introduced opimization also computes an illumination context using the BRDF and light source location, which is particularly applicable to dim, watertight settings where the lighting model of the endoscope or borescope can be of great utility.

The proposed method, Light Encoded Gaussian Splatting (LE-GS), demonstrated an average PSNR of 27.29 on the C3VD dataset, compared to an average PSNR of 25.49 using EndoGaussian. The Simultaneous Localization and Mapping using Gassian Splatting (GS-SLAM) algorithm tested on a 3D printed pipe demonstrated a PSNR of 4.59, demonstrating the potential efficacy of our context-based optimization to reconstruction in medical settings.

CHAPTER 1
INTRODUCTION

Scene reconstruction from monocular image sequences has significantly advanced in the past few years, with neural methods dominating this leap. In particular, NeRF and 3DGS have paved the way for novel view synthesis: NeRF brought precise scene representation via a Multi-layer Peceptron (MLP) model conditioned on position and viewing angle, and 3DGS vastly reduced the runtime for reconstruction with the use of gaussians for scene representation and tile-base splatting in place of the costly ray marching used in NeRF [1, 2].

In addition to neural methods, optimizations such as MLP models ”fully-fused” on GPUs, hash maps, and numerical gradients have rapidly decreased the runtime and increased the throughput of NeRF [3, 4, 5]. This has opened the doors to real-time applications of reconstruction methods, where previously, scene reconstruction from monocular images would have required many hours of processing on a powerful GPU.

NeRF and its variants, along with 3DGS, are widely applicable, but endoscopy, engine inspection, and other similar settings pose a host of challenges. These settings share the characteristics of being too narrow to fit an accurate depth camera and presenting poor features for matching. Although optimizations have been made to both NeRF and 3DGS, there is a lack of explainability that is inherent to universal function approximators, which limits NeRF optimizations in clarity and safety. Despite the more explicit representation of scenes with gaussians in 3DGS, depth information is assumed to be precomputed via a Structure from Motion (SfM) tool such as COLMAP to initialize the gaussians [6, 7]. The absence of texture in endoscopic images leaves SfM struggling to recover information, as well as demanding extensive memory and runtime when sparse point cloud recovery is possible. Narrow environments only permit monocular camera usage as opposed to stereo cameras, which can compute depth via epipolar geometry. While NeRF and 3DGS aim for generalizability to diverse scenarios, medical and aerospace applications require meticulous precision and thus motivate task-specific learning. Additionally, the environment in endoscopy is constantly deforming while the motion of endoscopes can often be considered quasistatic.

Some works have explored this setting, with Neural Implicit Surfaces using Illumination Decline (LightNeuS) expanding on Neural Implicit Surfaces (NeuS) by adding in an illumination context based off of the Bidirectional Reflectance Distribution Function (BRDF); however, the reliance on deep MLP models for learning view-dependent features detracts from the clarity of the approach [8, 9]. For tracking deformation, 4D Gaussian Splatting (4DGS) makes the use of 6 feature maps which can be qualitatively examined. EndoGaussian employs 4DGS to keep track of feature maps spatially and temporally and achieves the best performance of current methods, but it fails to incorporate information about illumination of the scene and uses DepthAnything for Gaussian initialization, which is a generic monocular depth estimation framework [10].

This paper proposes a more explainable approach for reconstructing a surface during endoscopy, as well as validating this approach qualitatively by examining model features and quantitatively by testing on image datasets and on 3D printed parts. The contributions in this paper are as follows:

- An illumination-based approach to 3D reconstruction in colonoscopy as well as engine inspection.
- Analysis of various reconstruction techniques in conjunction with the illumination optimization, with focus on implicit neural and explicit gaussian representations.
- Experimental validation of methods in simulation and on a 3D printed phantom model.

CHAPTER 2
BACKGROUND AND RELATED WORK

Real-time reconstruction of three-dimensional environments is most accurately achieved using depth cameras, whether this means a stereo camera, time-of-flight camera, or structured light camera. However, when reconstruction needs to be performed with a monocular camera due to environmental constraints, options for reconstruction are limited. Broadly speaking, the problem can be split into two components: depth estimation and motion estimation. This work examines the problem of depth estimation and introduces motion estimation subsequently.

2.1 Depth Estimation

Assuming a static or quasi-static environment, recovering information for 3D reconstruction requires depth estimation, which is not absolute unless there is some measure of scale provided, such as via calibration. Prior work has shown that monocular estimation is possible with the use of deep neural networks, as well as by taking advantage of a known lighting model.

DepthAnything has shown the most promise for general-purpose, zero-shot 3D reconstruction [10]. Based on MiDaS and DINOv2, DepthAnything uses a teacher-student paradigm for taking advantage of labeled and unlabeled data, harnessing the power of several terabytes of data overall [11, 12]. This immense body of pre-training data explains the superior accuracy of this model compared to MiDaS and ZoeDepth [13]. In terms of the model specifics, the DINOv2 encoder is used for feature extraction, and the DPT decoder from MiDaS is used to regress depth values from the model's latent space. This encoder-decoder model is trained in a supervised manner for 20 epochs, followed by a single epoch of training on unlabeled data with a ViT-L ”teacher” encoder for supervision.

Although DepthAnything generalizes well, task-specific learning is crucial for maximizing extracted features. The most prevalent method of accomplishing this in medical settings is by formulating an illumination model of the endoscope [14]. Modrzejewski et al. set the stage for incorporation of photometry in medical assistance, developing an illumination model to aid with laparoscopy and tackling questions regarding lighting model complexity and varying calibration approaches [15]. Batlle et al. were able to create a 3D reconstruction of the colon in vivo with only 3mm error, which was later generalized to neural methods as well [9, 14].

The lighting system for an endoscope can be modeled as a Point Light Source (PLS) due to its negligible size, so incident irradiance from the light source P to a point on the surface x given principal direction $\vec{d}$ can be modeled as follows:

$$\begin{aligned} L_i(x) &= cos^k(\phi) * \frac{\sigma}{||(x-P)||^2} \\ \phi &= < x - P, \vec{d} > \end{aligned} \tag{2.1}$$

The formulation above takes into account cosine radial decay due to optical fiber transmission effects and inverse square decay from the light source. Here, σ is the maximum emitted radiance, and $k = 4$ was found to be the best term for modeling radial decay modulatio [14]. In order to model exitant radiance, i.e. reflected radiance back to the camera, the surface is assumed to be opaque; thus, L_o can be modeled using the BRDF:

$$\begin{aligned} L_o(x, \omega_i, \omega_o) &= L_i(x) * BRDF(\omega_i, \omega_o) cos(\theta) \\ \theta &= < \omega_i, \vec{n} > \end{aligned} \tag{2.2}$$

Incident angle ω_i and exitant angle ω_o are measured with respect to the surface normal $\vec{n}$, and the cosine term accounts for the solid angle of the ray. Vignetting can be empirically approximated with $V(x) = cos^{k'}(\alpha)$, and auto-gain can be approximated with a multiplier g_t and a power γ.

$$L_o(x) = (\frac{cos^k(\phi)\sigma}{||x-P||^2} * BRDF(\omega_i, \omega_o) * cos(\theta) * V(x) * g_t)^{\frac{1}{\gamma}} \tag{2.3}$$

A further simplifying assumption that the light source is collocated with the camera location can be made, which results in the following formulation [14]:

$$cos^k(\phi)V(x) = cos^k(\phi)cos^{k'}(\alpha) \approx cos^k(\alpha) \tag{2.4}$$

$$L_o(x) = (\frac{cos^k(\alpha)\sigma}{||x-P||^2} * BRDF(\theta) * cos(\theta) * g_t)^{\frac{1}{\gamma}} \tag{2.5}$$

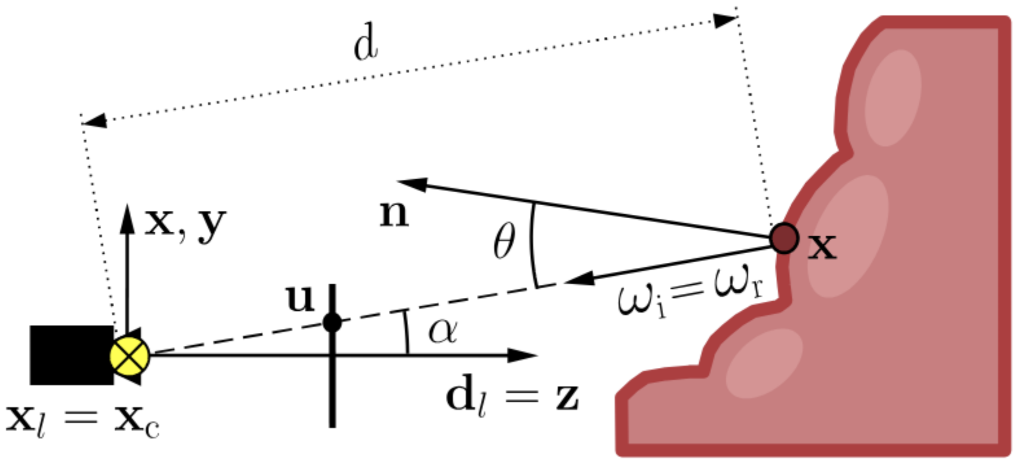


Figure 2.1: Photometric model after simplifications [14].

2.2 Neural Implicit Functions

2.2.1 NeRF

NeRF is a 3D reconstruction technique which employs deep learning to create an implicit, continuous neural representation of a scene from 2D images [1]. The deep network at the core of this approach takes in a 5D representation including position and viewing angle of the camera, and it outputs RGB colors and volume densities for the reproduced image.

To synthesize images from the volumetric representation, NeRF employs volume rendering techniques. The color of a pixel is determined by integrating the radiance along the camera ray passing through the pixel:

$$C(r) = \int_{t_n}^{t_f} T(t)\sigma(\vec{r}(t))c(\vec{r}(t), \vec{d})dt \tag{2.6}$$

Here, the ray $\vec{r}$ is parametrized be t and integrated from the near bound bound to the far bound to obtain the color. This volumetric equation takes into account the volume density, σ, and the accumulated transmittance, defined as follows:

$$T(t) = exp(-\int_{t_n}^{t} \sigma(\vec{r}(s))ds) \tag{2.7}$$

To efficiently compute the volume rendering integral, NeRF adopts a hierarchical volume sampling strategy. Initially, a coarse network is used to sample points along the ray, capturing the broad structure of the scene. A finer network then refines these samples, focusing on areas with high variance. This multi-resolution approach ensures both computational efficiency and high-quality renderings.

A critical aspect of NeRF is its use of positional encoding to handle high-frequency details. Directly inputting 3D coordinates into a neural network often results in poor performance due to limited representation capacity. Instead, NeRF maps input coordinates to a higher-dimensional space using functions like spherical harmonics or additional neural networks. A spherical harmonic encoding of degree L is constructed as follows:

$$\gamma(\vec{x}) = (sin(2^0\pi\vec{x}), cos(2^0\pi\vec{x}), ..., sin(2^{L-1}\pi\vec{x}), cos(2^{L-1}\pi\vec{x}) \tag{2.8}$$

2.2.2 I-NGP

A major limitation of the use of NeRF in SLAM settings is the immense training time required for each model. Each novel scene requires a completely new NeRF model to be trained. Instant Neural Graphics Primitives (I-NGP) is an extension of NeRF which vastly speeds up computation by augmenting a ”fully-fused neural” MLP with a trainable, multiresolution hash encoding [4, 3].

Fully-fused MLPs (FF-MLP) take advantage of the inherent parallelism of neural network operations [3]. At their core, neural networks consist of repeating layers of matrix multiplications of feature vectors with weight matrices followed by a nonlinearity. GPU computation exploits embarrassingly parallel applications such as this, so FF-MLP is implemented to run quickly on CUDA kernels specifically. Input features are split into batches of size 128 that are processed on their own thread block. This batch size, in conjunction with the relatively narrow layer height of 64, allows all of the weights to fit into GPU registers and all of the 64x128 activations to fit into GPU shared memory, avoiding expensive trips to global memory or deeper caches. Within each thread block, the hidden layer being operated on is further split into 16x16 blocks and performs row-wise multiplication with the corresponding weight matrix. This means that a 16x64 row of the 64x64 weight matrix is stored directly in registers and only loaded from global memory once - the key reason for FF-MLP operating so quickly.

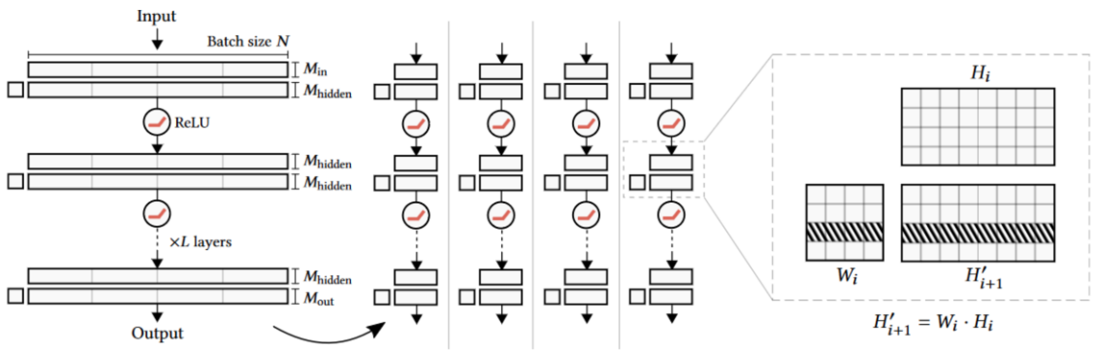


Figure 2.2: Batched MLP evaluation, distribution of batched operations over threads, and block row-wise multiplication [3].

The other component of I-NGP, a multiresolution hash encoding, takes into account features on both coarse and fine scales while retaining the trainability of feature vectors. The use of a dense grid in this scenario would be wasteful since the same number of features are used to represent empty space as feature-rich space. The intuition behind storing features in hash tables like this is that coarse features will map one-to-one to entries in the grid, while fine features will map many-to-one, and only the most important fine features will be kept over many iterations. The implication of this is that there is no collision resolution strategy as the network is trained keep the most important features when hash collisions occur. Backpropagation with a multiresolution hash encoding scheme is also significantly faster since weight updates only need to occur from the surrounding eight cells of the 3D grid via trilinear interpolation.

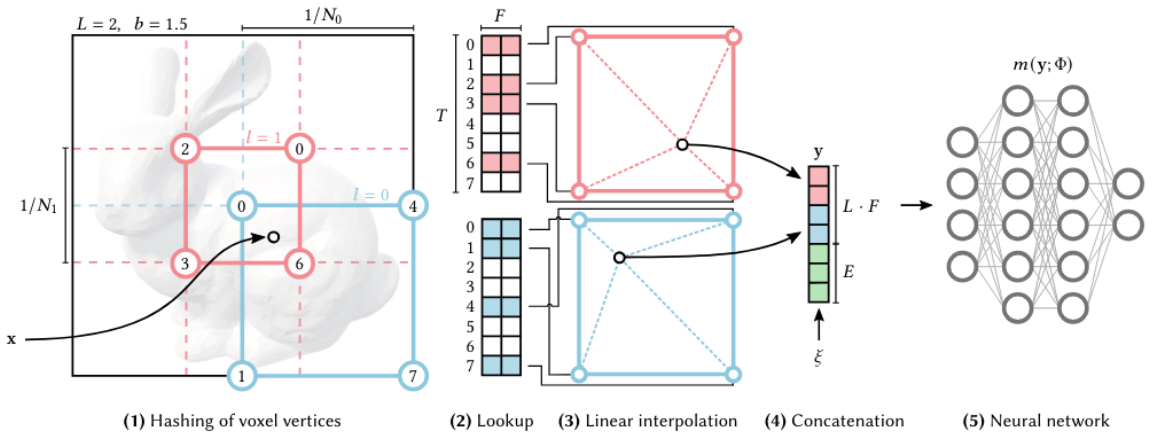


Figure 2.3: Hash value retrieval at multiple resolutions [4].

Following the hashing and retrieval process in more detail, for an input $\vec{x}$, a hash value is computed, and voxels at each resolution are retrieved. For every corner index of each voxel, a feature vector is retrieved from each hash table and use this to linearly interpolate where the location of this feature is within each voxel. These interpolated positions are concatenated and passed into an FF-MLP. Then, gradients are computed, backpropagating from the FF-MLP through concatenation, interpolation, and the hash tables. The hash function itself is the bitwise XOR of the product of each the i-th dimension of each feature and a large prime number. For example, for coordinate (x,y,z) and large generated prime numbers (p1, p2, p3) with hash table size n, the hash would be as follows:

$$hash(x, y, z) = ((x * p1)\ \textbf{XOR}\ (y * p2)\ \textbf{XOR}\ (z * p3))\ \text{mod(n)}$$

2.2.3 Neuralangelo

Neuralangelo further extends the efficacy of NeRF by adding more functionality to I-NGP in the form of higher-order derivatives with numerical gradients. A coarse-to-fine optimization not only occurs for hash resolutions, but also for gradient step sizes. If computed numerical gradients have a step size larger than the size of the grid cell, then a smoothing operation is performed; if it is smaller, then this is equivalent to just computing the analytical gradient. This helps in preventing local extrema in the gradient that are not representative of global behavior. Additionally, mean curvature regularization of the SDF results in a much smoother overall geometry - a favorable property in numerous medical settings but not in engine inspections. An appearance embedding is also included for exposure, lighting, weather-related effects, and other environmental factors.

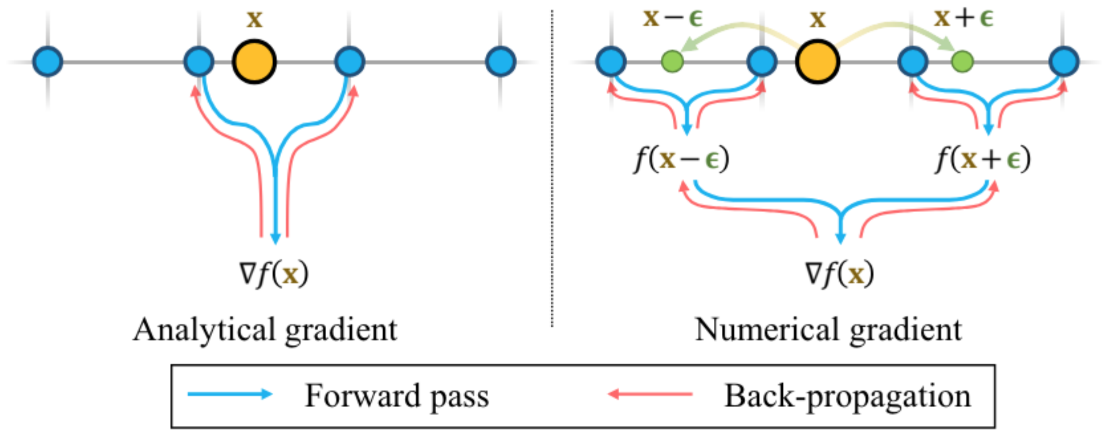


Figure 2.4: Numerical gradient computation smoothing analytical gradients [5].

2.2.4 NeuS

NeuS proposes the idea of presenting the zero-level set of the Signed-Distance Function (SDF) of a scene using a neural representation similar to that of NeRF [8]. The concept of reconstructing just a surface removes geometric errors by assuming watertightness, a useful characteristic in medical settings. A volume rendering scheme is defined, where the output color for a pixel is computed by accumulating colors and various points $p(\vec{t})$ along ray direction $\vec{v}$:

$$C(\vec{v}) = \int_0^{\infty} w(t)c(\vec{p}(t)), \vec{v})dt \tag{2.9}$$

The notable characteristic here is the use of a weight function $w(t)$, which must be unbiased so that the locally maximal value is in the zero-level set of the SDF as well as occlusion-aware so that two points with the same signed distance are weighted such that the nearer point contributes more to the color.

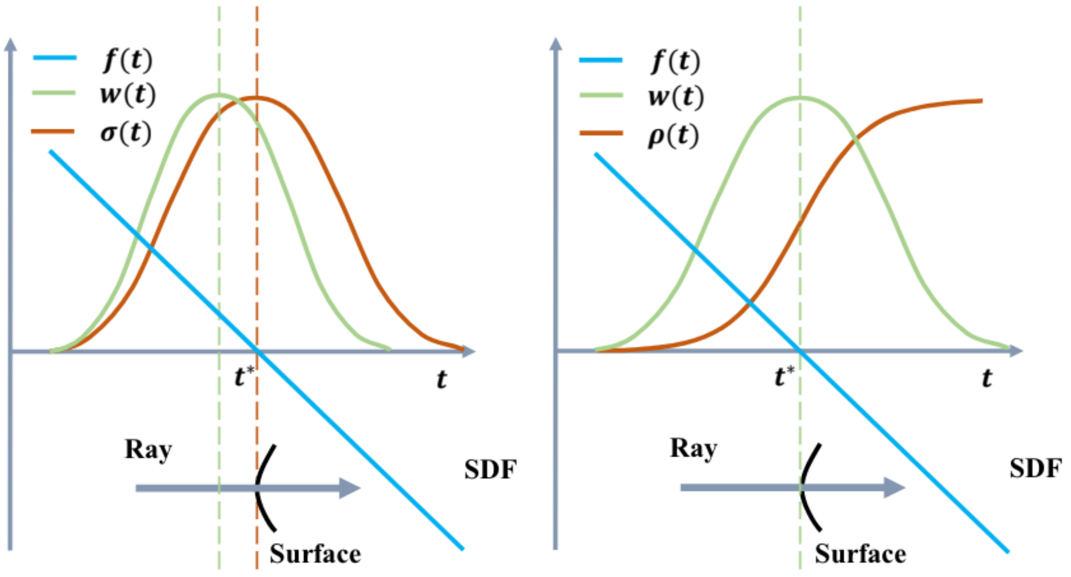


Figure 2.5: A biased SDF weighting (left) versus an unbiased one (right) [8].

The weight function is defined using logistic density function $\phi_s(x) = se^{-sx}/(1 + e^{-sx})^2$ due to its easy of differentiability and unimodal distribution with a zero center:

$$w(t) = \frac{\phi_s(f(\vec{p}(t)))}{\int_0^\infty \phi_s(f(\vec{p}(u)))du} \tag{2.10}$$

This formulation for weight, however, is not occlusion-aware, since if two points are both intersected by the corresponding ray that this weighting is occurring upon, both points will be considered zero points on the ray and the final color will just be an average of the two. NeuS takes a different approach, splitting the weight into accumulated transmittance $T(t)$ and opaque density $\rho(t)$ (instead of volume density $\sigma(t)$ like and then deriving the opaque density:

$$\begin{aligned} w(t) &= T(t)\rho(t) \\ T(t) &= exp(-\int_0^t \rho(u)du) \end{aligned} \tag{2.11}$$

$$\rho(t) = max(\frac{-\frac{d\phi_s}{dt}(f(\vec{p}(t)))}{\phi_s(f(\vec{p}(t)))}, 0) \tag{2.12}$$

An in-depth derivation of the opaque density can be found in the supplementary material of the paper by Wang et al [8].

2.2.5 LightNeuS

LightNeuS works to combine the photometric approach to 3D reconstruction with NeuS, making a couple of modifications to the NeuS formulation [9]. Recall the formulation for exitant radiance from section 2.1:

$$L_o(\vec{x}) = (\frac{L_e}{t^2} BRDF(\vec{x}) * cos(\theta) * g_t)^{1/\gamma} \tag{2.13}$$

NeuS, however, does not take advantage of the light source being collocated with the camera, and thus struggles to learn the $1/t^2$ dependency. To rectify this, the color $C(r)$, computed by NeRF as shown in section 2.2.1, can be reformulated as follows:

$$C(r) = \int_{t_n}^{t_f} T(t)\sigma(\vec{r}(t))\frac{c(\vec{r}(t), \vec{d})}{t^2}dt \tag{2.14}$$

To account for the additional factors of emitted radiance, auto-gain, and gamma correction, photometric loss is computed using normalized images as opposed to the original images:

$$I' = (\frac{I^\gamma}{L_e g})^{1/\gamma} \tag{2.15}$$

With these simple modifications, LightNeuS has paved the way for exploitation of environmental factors for adapting neural 3D reconstruction to settings which have been historically difficult to gather data from.

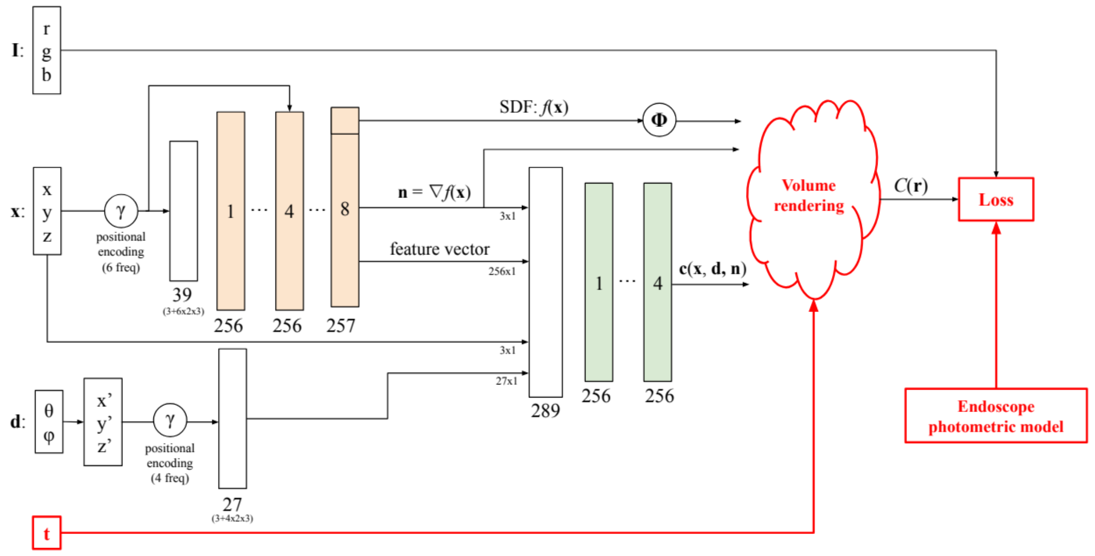


Figure 2.6: The original NeuS architecture with photometric modifications outlined in red [9].

2.3 Gaussian Splatting

I-NGP attemps to create a more efficient representation of empty space by using multiresolution hashmaps as opposed to a dense grid, but 3DGS takes this a step further by only representing occupied space as anisotropic 3D gaussians [2].

2.3.1 3D Gaussian Splatting

3D gaussians are ”splatted,” or projected, onto a 2D image plane, where alpha blending combines them - a much more computationally efficient process than repeated hierarchical sampling along rays. Alongside this, the approach is significantly more explainable due to the explicit representation of the scene as gaussians as opposed to neural networks. The tile-based rasterization uses fast GPU sorting for its forward pass and is differentiable during its backward pass, which is the key to the success of this method over neural rendering approaches.

Gaussians are defined with a full 3D covariance matrix, where the covariance matrix can be decomposed into a scaling matrix S and a rotation matrix R:

$$G(x) = e^{\frac{1}{2}x^T(RSS^TR^T)x} \tag{2.16}$$

Based on the hyperparameters of the approach, gaussians are inserted or pruned after each iteration, with large gaussians being split and gaussians in sparse areas being cloned.

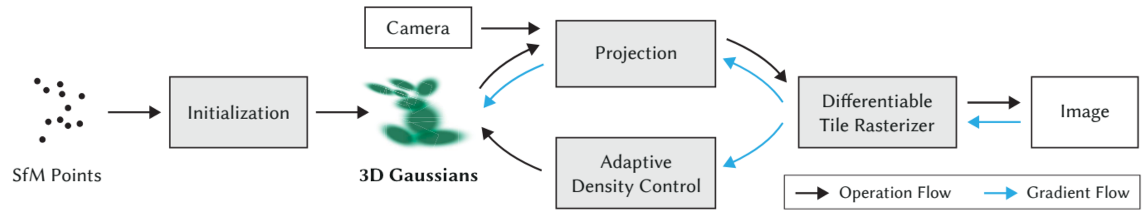


Figure 2.7: The Gaussian Splatting process [2].

2.3.2 Deforming Gaussians

In intraoperative medical settings such as colonoscopy, the environment moderately deforms between frames and moreso with robot contact. Tracking new gaussians for every timestep would be extremely inefficient, so a more compact representation is proposed by Liu et al. and Wu et al [16, 17].

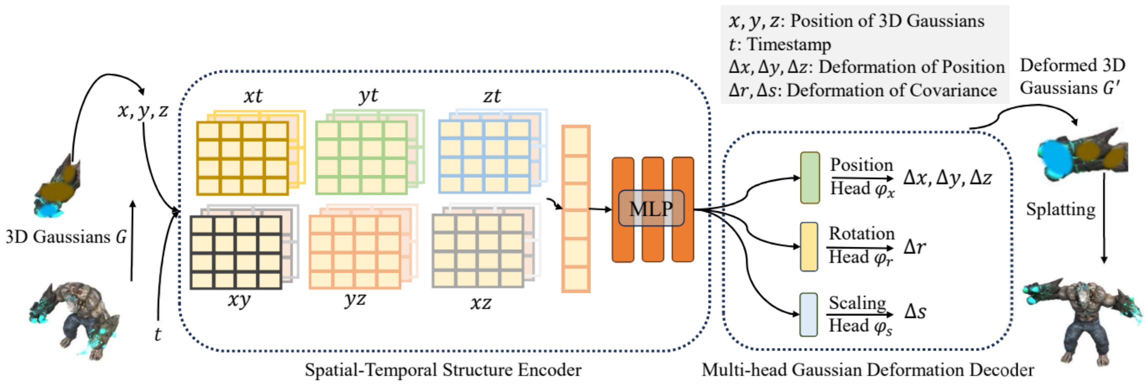

x, y, z: Position of 3D Gaussians
t: Timestamp
$\Delta x, \Delta y, \Delta z$: Deformation of Position
$\Delta r, \Delta s$: Deformation of Covariance
Deformed 3D Gaussians G'
x, y, z
xt
yt
zt
xy
yz
xz
3D Gaussians G
t
MLP
Position Head φ_x
$\Delta x, \Delta y, \Delta z$
Rotation Head φ_r
Δr
Scaling Head φ_s
Δs
Splatting
Spatial-Temporal Structure Encoder
Multi-head Gaussian Deformation Decoder

Figure 2.8: How HexPlane forms a compact representation of deformation. [17].

In the EndoGaussian system, 3D gaussians are projected into space and time planes to encode their motion in both planes before being passed through a small MLP. This voxel encoding approach is called HexPlane, and it is very powerful for computing features that are both representative and explainable, avoiding the need for a deep neural network [18]. HexPlane can be represented via 6 concatenated learned feature planes of form $M_r^{AB} \in \mathbb{R}$ and feature vectors v_r^i:

$$D = \sum_{r=1}^{R_1} M_r^{XY} \otimes M_r^{ZT} \otimes v_r^1 + \sum_{r=1}^{R_2} M_r^{YZ} \otimes M_r^{XT} \otimes v_r^2 + \sum_{r=1}^{R_3} M_r^{XZ} \otimes M_r^{YT} \otimes v_r^3 \quad (2.17)$$

Here, $\otimes$ is the outer product. As one can see, querying such a plane is much faster than using a deeper neural network, as only six bilinear interpolations and one vector-matrix product need to be performed.

2.3.3 SLAM with Gaussian Splatting

Simultaneous Localization and Mapping (SLAM) aims to solve the problems of pose estimation and reconstruction concurrently, performing a time update based on the current motion estimates and a measurement update based on feature matches that can impose constraints on the model. In order to utilize gaussian splatting in a real-time reconstruction context, a combination of the two, GS-SLAM, was proposed by Yan et al [19].

For the pose estimation portion, referred to as the tracking thread, a photometric residual is optimized, comparing rendered image $I(G)$ with gaussians G from pose T_{CW} with the observed image $\bar{I}$:

$$E_{pho} = ||I(G, T_{CW}) - \bar{I}||_1 \tag{2.18}$$

Maintaining gaussians for the whole scene would be very computationally expensive, so GS-SLAM adopts a keyframing approach, where a window of gaussians is maintained based on the proportion of gaussians that are covisible between frames. This goes hand-in-hand with the existing gaussian insertion and pruning scheme of 3DGS.

The mapping thread optimizes over the photometric error as well as an isotropic error – a regularization term for mitigating gaussians with extreme scales – preventing unwanted artifacts:

$$\min_{T_{CW}^k \in SE(3), G, \forall k \in W} \sum_{\forall k \in W} E_{pho}^k + \lambda_{iso} E_{iso} \tag{2.19}$$

Here, W is the window of keyframes being operated over and λ_{iso} is the weighting for the isotropic regularization term.

CHAPTER 3
METHODOLOGY

This project focused on bringing illumination-based optimizations to the realm of 3D reconstruction, just as LightNeuS was able to. First, the photometric approach for 3D reconstruction detailed in Batlle et al was recreated, as it is not open source: Link to Github [14]. Notably, both multithreaded Python and C++ implementations were developed. In order to accomplish the optimization detailed in the paper, the Ceres optimization library was used:

$$k, g_t, \gamma | \forall t* = \arg\min_{k, g_t, \gamma} \sum_{j,t} \rho(I_{jt} - L_o(x_j, k, g_t, \gamma)) \tag{3.1}$$

Here, ρ is a robust cost function, j is the index for sample points, and t is the frame at timestep t.

After this, LightNeuS also had to be developed from scratch, as the open-source implementation for this is also not available: Link to Github [9].

After this, two proposed systems were implemented: Light Encoded Neural Graphics Primitives (LE-NGP) and LE-GS. LE-NGP extends Neuralangelo very similarly to LightNeuS, modifying the images for emitted radiance, gamma correction, and gain for the loss function and adding the t^2 dependency. Neuralangelo is a good choice to build upon since it incorporates neural SDF computation as opposed to just volume rendering, maintaining the element of watertightness present in LightNeuS.

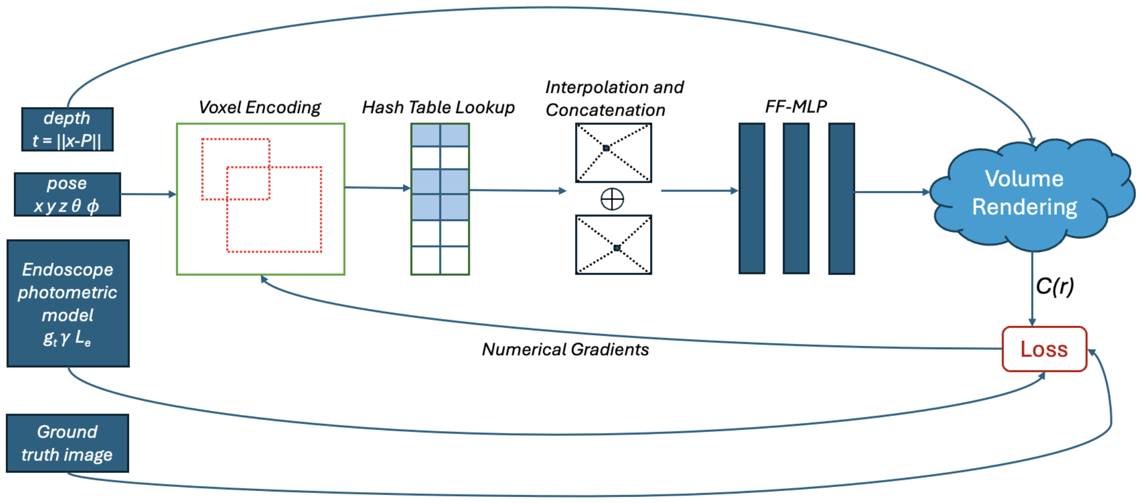


Figure 3.1: Architecture of LE-NGP model.

LE-GS addresses the same issue by extending EndoGaussian, once again modifying images for the loss function as well as adding the t^2 dependency to the α-blending step for determining the color C of a pixel, where α represents opacity and there are N ordered points:

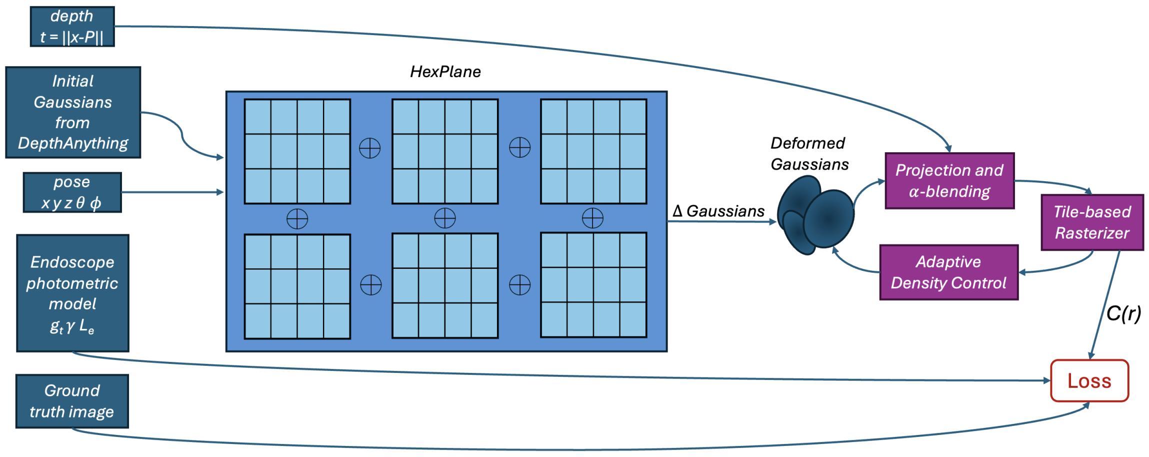


Figure 3.2: Architecture of LE-GS model.

$$C = \sum_{i \in N} \frac{c_i}{t_i^2} \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) \tag{3.2}$$

Here, z_i is the distance to the i-th gaussian.

For the hardware setup, a phantom was first created of the Cecum T1 A simulation video from the C3VD Dataset. The mesh was gathered for Cecum T1 A and it was placed inside a 3D printed pipe. Then, a borescope was inserted into the pipe, maintaining an approximately linear path in and out. In future experiments, a more precisely controlled continuum robot will be actuated to ensure precise camera motion. The deformation and illumination modifications were made to the 3DGS element of GS-SLAM, essentially combining LE-GS with GS-SLAM for the algorithm tested on hardware.

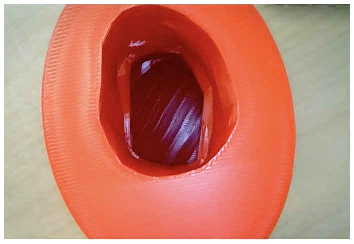


Figure 3.3: Snapshot of the phantom captured by borescope.

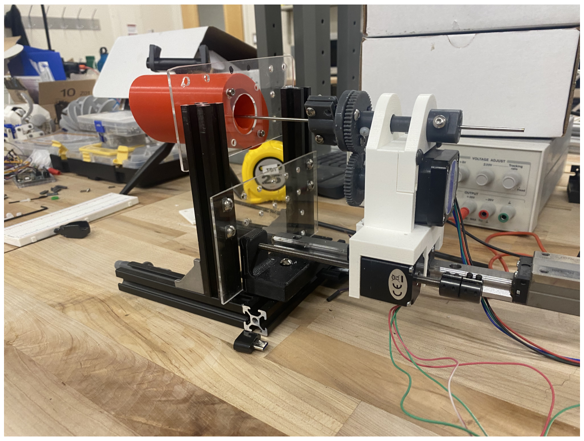


Figure 3.4: Continuum robot setup with an endoscope and phantom for future experiments.

CHAPTER 4
RESULTS

The baselines of LightNeuS and EndoGaussian were compared to LE-NGP and LE-GS on the C3VD dataset [20]. LE-NGP demonstrated poor performance, most likely due to the decreased computational capacity of the model with the use of a smaller MLP and hash encodings for the representation. Furthermore, LE-NGP required significant GPU memory to run, so some of the subsets of the C3VD dataset could not run on the Georgia Tech PACE cluster, as indicated by the blank entries in Table 4.1. LE-GS, however, outperformed EndoGaussian in most simulated settings, displaying the efficacy of incorporating photometric information.

Table 4.1: C3VD PSNR with A100 GPU

	LE-NGP	**LE-GS**	**EndoGaussian**	**LightNeus**
Cecum T1 A	11.01	30.54	26.24	31.19
Cecum T1 B	5.21	30.26	25.93	15.04
Cecum T2 A	11.25	34.81	32.80	27.43
Cecum T2 C	5.52	27.54	24.51	23.02
Cecum T3 A	7.28	27.01	25.26	18.73
Cecum T4 A	5.90	33.07	29.88	15.39
Cecum T4 B	6.01	28.46	27.97	15.76
Descending Colon T4 A	–	27.36	20.90	12.09
Sigmoid Colon T1 A	–	27.03	26.77	31.24
Sigmoid Colon T2 A	–	33.14	32.04	15.67
Sigmoid Colon T3 A	–	29.69	28.28	22.06
Sigmoid Colon T3 B	–	23.75	24.71	15.51
Transcending Colon T2 A	4.52	30.98	28.32	23.10
Transcending Colon T2 B	4.83	23.39	21.05	13.79
Transcending Colon T2 C	3.97	29.03	33.18	11.76
Transcending Colon T3 A	4.35	19.86	18.01	11.78
Transcending Colon T3 B	4.41	24.18	21.10	11.82
Transcending Colon T4 A	4.34	18.39	16.34	15.92
Transcending Colon T4 B	7.28	19.99	21.02	22.22

In terms of hardware testing, the PSNR values are less meaningful in comparison to the qualitative results. For the engine inspection task, fine details of the engine were not recovered due to the lack of meaningful pose information. A key ablation involved separately testing COLMAP SFM and DiffPoseNet to recover pose information for aiding reconstruction [21]. Reconstruction was also attempted using pixel-wise depths from a DepthAnything depth estimate. An open-source implementation of DiffPoseNet was created, as it was not previously available: Link to Github. DiffPoseNet demonstrated faster and more effective pose estimation, but could only achieve the quality of what might be produced by noisy IMU readings. The ground truth trajectory was a linear path in and out of the pipe, as can be seen by the hardware setup in the methodology section.

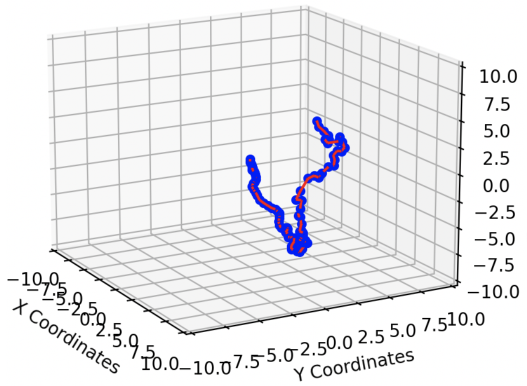


Figure 4.1: Estimated path from DiffPoseNet.

A larger engine reconstruction attempt was made - a Concord jet engine at the Georgia Tech Aerospace building - which resulted in poor reconstruction as well.

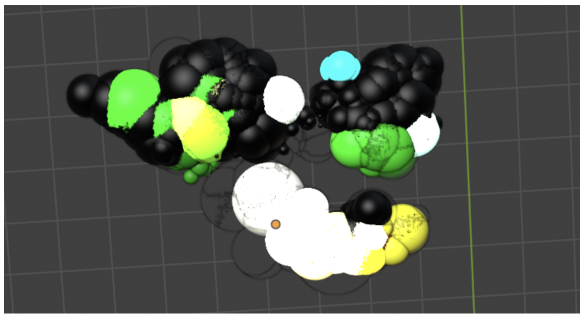
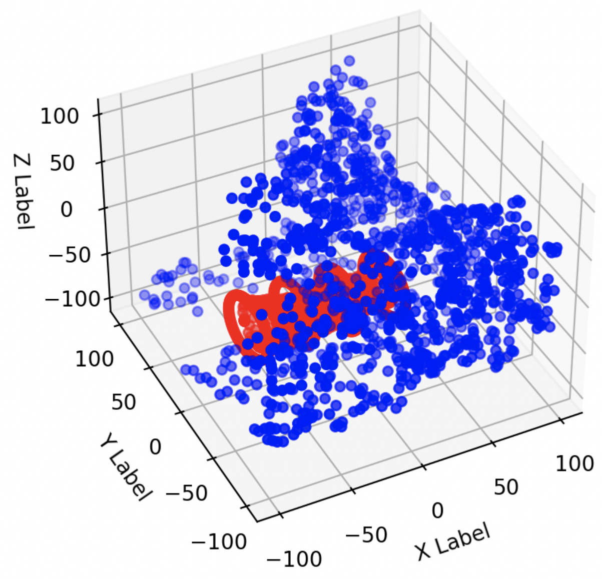


Figure 4.2: Gaussians from Concord engine reconstruction in Blender.

The output of GS-SLAM on the colon phantom pipe shows greater potential, achieving a rough estimate of colon walls and haustral folds while not producing accurate estimates.



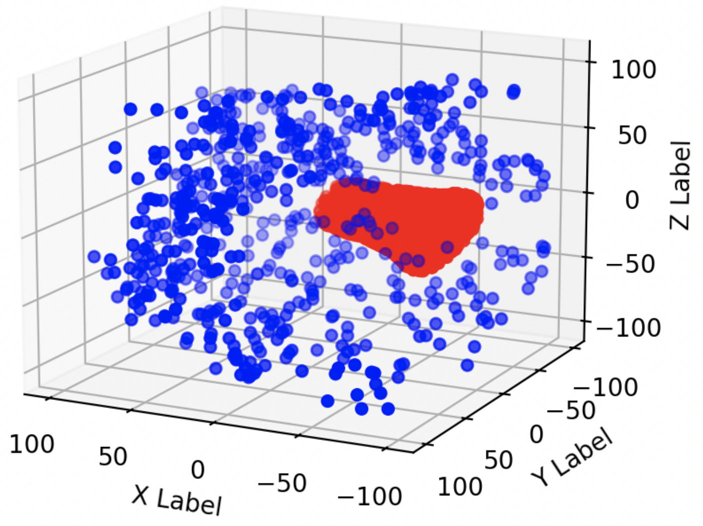


Figure 4.3: Actual pipe CAD (red) compared to LE-GS estimate (blue).

CHAPTER 5
CONCLUSION AND DISCUSSION

This work introduces a novel approach for monocular surface reconstruction in feature-poor settings such as in endoscopy and borescope engine inspection by building upon EndoGaussian. The proposed method leverages a fast illumination-based initialization for gaussians and integrates maps of illumination at each timestep, enhancing the spatial and temporal feature representation. Through simulation and experimental validation, this project demonstrated the efficacy of the approach in both simulated environments and physical hardware tests. It highlighted improvements in precision and real-time applicability over existing methods, while also analyzing apparent flaws. These advancements suggest a significant potential for improving the accuracy and reliability of monocular surface reconstruction in challenging environments.

Despite the promising results, several limitations need to be addressed. Although LE-GS performed well on simulated colon data, it could not yield a reasonable reconstruction for any of the hardware settings. The qualitative analysis of feature maps show that, although insightful, they necessitate further refinement to ensure consistency across different datasets and scenarios. Furthermore, the dependence on monocular cameras poses inherent challenges in depth estimation, particularly in textureless or highly reflective environments where traditional SfM techniques struggle.

Future work can focus on addressing these limitations by exploring various ablations, such as the use of multiple input sources as ”sensor input” to the SLAM algorithm. Notably, the combination of DiffPoseNet with LE-GS and SLAM can help to decouple the problem of pose estimation from gaussian pose and property estimation. Additionally, incorporating advanced machine learning techniques such as domain adaptation and transfer learning could enhance the robustness of the model across various environments. Further, expanding the scope of experimental validation to include more diverse datasets and real-world applications will be crucial in assessing the generalizability and scalability of the proposed approach. By addressing these challenges, a more comprehensive and reliable solution for monocular surface reconstruction in endoscopy and other narrow, hard-to-reach environments can be developed.

REFERENCES

[1] 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 *ECCV*, 2020.

[2] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, “3d gaussian splatting for real-time radiance field rendering,” *ACM Transactions on Graphics*, vol. 42, no. 4, Jul. 2023.

[3] T. Müller, *tiny-cuda-nn*, version 1.7, Apr. 2021.

[4] T. Müller, A. Evans, C. Schied, and A. Keller, “Instant neural graphics primitives with a multiresolution hash encoding,” *ACM Trans. Graph.*, vol. 41, no. 4, 102:1–102:15, Jul. 2022.

[5] Z. Li *et al.*, “Neuralangelo: High-fidelity neural surface reconstruction,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[6] 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.

[7] J. L. Schönberger and J.-M. Frahm, “Structure-from-motion revisited,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[8] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang, “Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction,” *NeurIPS*, 2021.

[9] V. M. Batlle, J. M. M. Montiel, P. Fua, and J. D. Tardós, *LightNeuS: Neural Surface Reconstruction in Endoscopy using Illumination Decline*, arXiv:2309.02777 [cs], Sep. 2023.

[10] L. Yang, B. Kang, Z. Huang, X. Xu, J. Feng, and H. Zhao, “Depth anything: Unleashing the power of large-scale unlabeled data,” in *CVPR*, 2024.

[11] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, “Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 3, 2022.

[12] M. Oquab *et al.*, *DINOv2: Learning robust visual features without supervision*, arXiv:2304.07193, 2023.

[13] S. F. Bhat, R. Birkl, D. Wofk, P. Wonka, and M. Müller, *Zoedepth: Zero-shot transfer by combining relative and metric depth*, arXiv:2302.12288, 2023.

[14] V. M. Batlle, J. Montiel, and J. D. Tardós, "Photometric single-view dense 3D reconstruction in endoscopy," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, ISSN: 2153-0866, Oct. 2022, pp. 4904–4910.

[15] R. Modrzejewski, T. Collins, A. Hostettler, J. Marescaux, and A. Bartoli, "Light modelling and calibration in laparoscopy," *International Journal of Computer Assisted Radiology and Surgery*, vol. 15, no. 5, pp. 859–866, May 2020.

[16] G. Wu *et al.*, "4d gaussian splatting for real-time dynamic scene rendering," *arXiv preprint arXiv:2310.08528*, 2023.

[17] Y. Liu, C. Li, C. Yang, and Y. Yuan, "Endogaussian: Gaussian splatting for deformable surgical scene reconstruction," *arXiv preprint arXiv:2401.12561*, 2024.

[18] A. Cao and J. Johnson, "Hexplane: A fast representation for dynamic scenes," *CVPR*, 2023.

[19] C. Yan *et al.*, "Gs-slam: Dense visual slam with 3d gaussian splatting," in *CVPR*, 2024.

[20] T. L. Bobrow, M. Golhar, R. Vijayan, V. S. Akshintala, J. R. Garcia, and N. J. Durr, "Colonoscopy 3d video dataset with paired depth from 2d-3d registration," *Medical Image Analysis*, p. 102 956, 2023.

[21] C. M. Parameshwara, G. Hari, C. Fermüller, N. J. Sanket, and Y. Aloimonos, "Diff-PoseNet: Direct Differentiable Camera Pose Estimation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2022.