CHALMERS
UNIVERSITY OF TECHNOLOGY

KNeeRF: NOVEL VIEW SYNTHESIS OF A KNEE'S INTERIOR

A 3D MODELING APPROACH TO OPTIMIZE GRAFT PLACEMENT

Master's thesis in MPALG & MPSYS

LUKAS SANDMAN
CHARLES STRÖMBLAD

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY

Gothenburg, Sweden 2024
www.chalmers.se

Master's thesis 2024

KNeeRF: NOVEL VIEW SYNTHESIS OF A KNEE'S INTERIOR

A 3D MODELING APPROACH TO OPTIMIZE GRAFT PLACEMENT

LUKAS SANDMAN
CHARLES STRÖMBLAD

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2024

KNeeRF: NOVEL VIEW SYNTHESIS OF A KNEE'S INTERIOR
A 3D MODELING APPROACH TO OPTIMIZE GRAFT PLACEMENT
LUKAS SANDMAN
CHARLES STRÖMBLAD

Examiner: Fredrik Kahl, Electrical Engineering, fredrik.kahl@chalmers.se
Supervisor: Filip Wikman, Tenfifty, filip.wikman@tenfifty.io

Master's Thesis 2024
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

KNeeRF: NOVEL VIEW SYNTHESIS OF A KNEE'S INTERIOR
A 3D MODELING APPROACH TO OPTIMIZE GRAFT PLACEMENT
LUKAS SANDMAN
CHARLES STRÖMBLAD
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Anterior cruciate ligament (ACL) reconstruction surgeries suffer from high relapse rates, due to the hard problem of placing the replacement graft. 3D models of knees could be used as training material for medical professionals to practice placing the ligaments to reduce relapse rates. We have developed a technique that can produce realistic renders of knees' interiors that can further be used in this context.

Our technique is based on Neural Radiance Fields (NeRF) to create novel views of the knee, where we compared the original NeRF implementation and two other implementations, Self-Calibrating NeRF (SCNeRF) and Nerf in the Wild (NeRF-W) to choose a starting point for further development. We landed on using NeRF-W as our baseline model. We have made several extensions to NeRF-W to further improve the application to ACL reconstruction surgeries, such as correcting ray distortions to produce accurate renderings and using segmentation masks to help the model remove medical tools from the renderings. We apply our system to data sets of ACL reconstruction surgeries and demonstrate results that surpass those of the models compared. We present an average improvement compared to NeRF-W with 5.7% in PSNR, 0.69% in SSIM, and 23.5% in LPIPS.

Keywords: NeRF, NeRF-W, SCNeRF, novel view synthesis, computer vision, ACL surgery

Acknowledgements

We would like to thank our supervisor at Tenfifty, Filip Wikman, for all the help regarding our work. He was a big part of the success of the thesis through our discussions about the best approach, explaining necessary theory, and more.

We also would like to give a huge thank you to our supervisor and examiner, Fredrik Kahl, Full Professor at Chalmers, for all the constructive feedback and expertise in the field of computer vision he contributed. He was especially of importance in talks regarding Neural Radiance Fields.

Finally, we thank our dear friend Erik Laitala for his continued support and encouragement during the time of writing this thesis.

Lukas Sandman & Charles Strömblad, Gothenburg, June 2024

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

2D	Two-dimensional
3D	Three-dimensional
ACL	Anterior Cruciate Ligament
AI	Artificial intelligence
BC	Bad conditions
CDF	Cumulative distribution function
CNN	Convolutional neural network
FC	Fully connected
IC	Ideal conditions
LPIPS	Learned Perceptual Image Patch Similarity
MLP	Multilayer perceptron
MSE	Mean squared error
NDC	Normalized Device Coordinate
NeRF	Neural Radiance Fields
NeRF-W	NeRF in the Wild
OC	Okay conditions
PDF	Probability density function
PRD	Projected ray distance
PSNR	Peak Signal-to-Noise Ratio
RGB	Red Green Blue
SfM	Structure from Motion
SCNeRF	Self-Calibrating NeRF
SSIM	Structural Similarity Index

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

Among knee injuries, anterior cruciate ligament (ACL) injuries rank among the most prevalent. According to the Swedish national knee ligament registry, around 4000 cruciate ligament surgeries are performed in Sweden alone [21]. Around 20% of the patients require additional surgery a few years after the initial surgery due to complications with the operated cruciate ligament [21]. An ACL surgery is most often performed by replacing the damaged cruciate ligament with a replacement graft, this graft is inserted in the knee using a minimally invasive arthroscopic technique, where a fiber optic camera together with small instruments are inserted into the knee [20], this procedure is illustrated in Figure 1.1. Using these small instruments and the fiber optic camera, surgeons determine where to optimally place the graft as this placement is important. If the ends of the grafts are placed too far from each other it can lead to the graft being damaged in the future. If the ends are placed too close to each other, the graft might not have full effect, which may prevent the patient from regaining the same range of motion they had before surgery, causing instability in the knee.

To alleviate this problem, we intend to develop a way to easily create renderings of the inside of a knee, which could be used in future work to assist surgeons in placing grafts in ACL reconstruction surgery. We intend to use a technique called Neural Radiance Fields (NeRF) that is used to synthesize realistic 3D scenes from images using deep learning techniques [17].

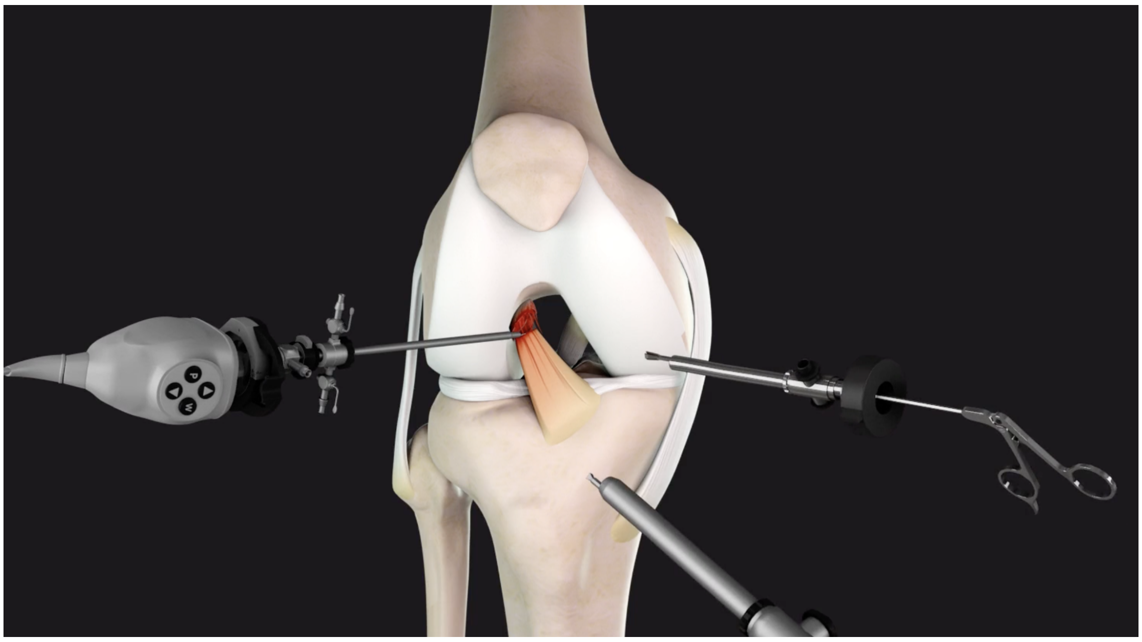
Figure 1.1: Illustration of an ACL reconstruction surgery. An arthroscope along with small medical instruments are inserted into the knee to repair the damaged cruciate ligament [35].

1.2 Related works

The computer vision field has many different approaches to novel view synthesis that are good for different applications. **Structure from motion**: An old technique used to create a 3D scene from images. It requires a lot of pictures and a wide view of the scene that our data sets do not contain. It lays the foundation for the camera parameters as well and gives a sparse point cloud for the scene [23]. **Neural 3D shape representation**: A new technique that maps coordinates to a signed distance function representing the geometry of the scene, however, it requires a ground truth depth map that is not available in this application [11]. **Neural radiance fields (NeRF)**: A state-of-the-art technique in novel view synthesis, it uses neural networks to model and render detailed 3D scenes by learning each scene's radiance and volumetric representation. NeRF, seeks to overcome some of the challenges present in earlier novel view synthesis techniques [17].

We chose to use NeRF in this thesis both due to its state-of-the-art performance along its popularity. Many NeRF implementations exist, each with properties fitting for this application. Beyond the original NeRF implementation, this thesis also studies two iterations of it, NeRF in the Wild (NeRF-W) and Self-Calibrating NeRF (SCNeRF). NeRF-W can handle variable lighting and transient objects that exist in our dataset and are prevalent challenges in ACL surgeries. To get an accurate rendering, we intend to test SCNeRF [10], which is a plugin that can be used with other NeRF models to process distorted images accurately. SCNeRF learns camera intrinsics (internal parameters) and extrinsics (camera poses) without requiring the lengthy process of an SfM initialization.

1.3 Aim

This thesis aims to see if it is possible to create novel views of the interiors of knees using data from ACL surgeries. This thesis is part of a larger project in which the end goal is to develop a tool to help medical professionals during ACL reconstruction surgery.

Many different versions of NeRF are intended for different use cases. Firstly, we intend to compare some of these and then further develop the model or models that seem most promising in this case.

1.4 Limitations

There are some hurdles in using NeRF in this application since NeRF is developed to handle rigid and non-dynamic scenes [17]. A knee is not rigid geometry, it can deform during the operation, and it contains organic matter that does not stay in one place and can block the field of view of the camera lens. During surgery, surgical instruments may also obstruct the view of the knee.

Additional challenges to address include problems with the lighting in the knee, which vary with camera movements. Furthermore, during ACL surgeries, the camera predominantly moves in one dimension, posing a challenge for NeRF, which relies on capturing diverse angles of the scene to be synthesized [17]. These problems will be evaluated by splitting the data into different data sets of varying quality as described in Section 3.1 and evaluating the different models on these different data sets.

The NeRF models intended to be evaluated require orientation and position of the camera as input, and therefore a structure from motion (SfM) algorithm, such as COLMAP [23] (2.4.1), will be used to get camera poses and parameters of the input images. These camera parameters are vital for most NeRF models to create accurate renderings and therefore a big part of this project will be devoted to adjusting these parameters.

A limitation of the available data is that it is captured using a monocular camera with lens distortion, and most NeRF models have been developed for images captured using cameras without this distortion [17, 22, 30, 15].

1.5 Research questions

- Is it possible to use NeRF or similar models to synthesize novel views of a knee's interior?

- Can we improve upon such a NeRF model and how much better does it perform compared to other NeRF implementations?

1.6 Overview of the thesis

In the above sections, we have outlined the need and importance of this thesis and the relevant background. In Chapter 2 we present the necessary theory to give the reader an understanding of the work. This chapter includes information about computer vision and the various NeRF models explained above.

In Chapter 3 we outline our chosen methodology of this thesis, which includes a section of our data processing (3.1) and our model development (3.2). The model development section consists of our contributions, which include ray undistortion (3.2.1), network architectural changes (3.2.2), and the implementation of segmentation masks (3.2.3).

We present the results of the thesis in Chapter 4, where we show both qualitative and quantitative results of the compared NeRF models and our data sets. We also display novel views rendered by our developed model.

In Chapter 5 we outline some of the difficulties in our work, such as issues with camera parameters, the available data set, and more. The results of an ablation study are also presented. Finally, we present the conclusions of the thesis alongside some interesting approaches to further work in Chapter 6.

2
Theory

This chapter presents relevant theory to give the reader further insight regarding the areas involved in this thesis. The two main areas covered are cameras in computer vision and Neural Radiance Fields.

2.1 Camera models

Cameras play a crucial role in the field of computer vision, enabling a digital representation of the physical world. Consequently, accurately modeling camera parameters is of paramount importance.

2.1.1 Intrinsic parameters

One of the simplest camera models is the pinhole camera model, which can be described by placing a barrier with a small opening, aperture, between a 3D object and a photographic film or sensor [8]. The barrier acts as a way to allow only certain rays of light to hit the film; without it, the film would be influenced by rays from all visible points on the 3D object, which would lead to a blurry image.

The pinhole camera can be formally defined, with the photographic film often referred to as the image plane, the aperture as the pinhole O, or the center of the camera. The distance between the pinhole O and the image plane is defined as the focal length f [8]. Let $P = \begin{bmatrix} x & y & z \end{bmatrix}^T$ be a point on some 3D object visible by the camera. P is then projected onto the image plane $\prod'$, giving the point $P' = \begin{bmatrix} x' & y' \end{bmatrix}^T$. The pinhole can also be projected onto $\prod'$, producing the point C'. A camera coordinate system $\begin{bmatrix} i & j & k \end{bmatrix}^T$ is centered at O with the axis k perpendicular to the image plane, pointing in the opposite direction of the camera. The line between C' and O is referred to as the optical axis of the camera system. A visual representation of a pinhole camera can be seen in Figure 2.1.

The 3D world imprints itself on the image plane and can be explained using triangles. In Figure 2.1 the triangle formed by O, C', P' and the triangle formed by O, D, P are similar, where $D = (0, 0, z)$. The law of similar triangles states that two similar triangles, $\triangle ABC$ and $\triangle AB'C'$ share the same ratio between their adjacent and opposite sides. This is formulated as follows

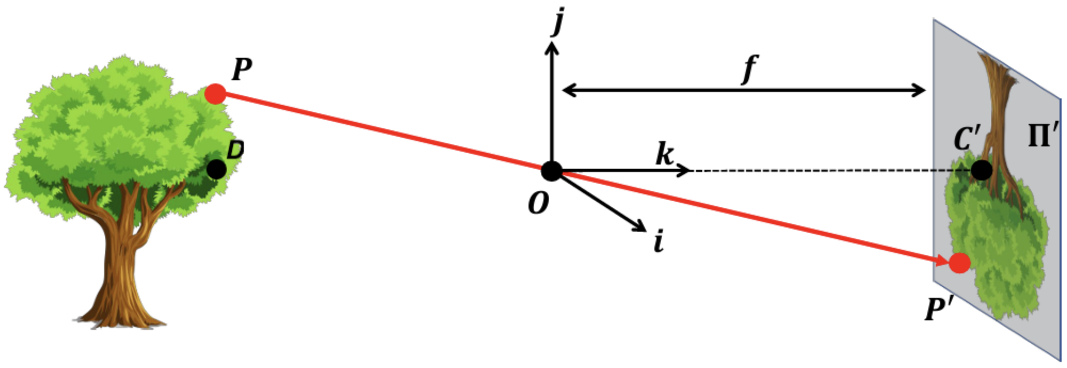

Figure 2.1: Formal representation of the pinhole camera model [8].

$$\frac{BC}{AB} = \frac{B'C'}{AB'}. \tag{2.1}$$

Using this formula, the image coordinates x', y' can be calculated using world coordinates as follows,

$$\frac{DP}{OD} = \frac{C'P'}{OC'} \Rightarrow \frac{DP}{z} = \frac{C'P'}{f}, \tag{2.2}$$

where the distance $OC' = f$ and the distance $OD = z$, the coordinate y' can then be calculated in the following way

$$\frac{P_yD}{z} = \frac{C'_yP_y}{f} \Rightarrow \frac{y}{z} = \frac{y'}{f} \Rightarrow y' = f\frac{y}{z}, \tag{2.3}$$

where P_y, C'_y and P'_y represent the y coordinate of the points P, C' and P'. The distance from P_y to D is of course just y, in the same spirit, the distance from C'_y to P'_y is just y' since $C'_y = 0$. The coordinate x' is calculated similarly and thus the point P' in the image plane is defined as $P' = \begin{bmatrix} f\frac{x}{z} & f\frac{y}{z} \end{bmatrix}^T$.

Digital images are most often referenced in another system than those in the image plane, digital images are split into discrete pixels while the image plane is continuous [8]. Physical sensors, such as lenses, can introduce distortions to the mapping; thus, additional transformations need to be introduced to be able to map 3D points in the world to pixel coordinates.

The camera matrix model is a model that describes the parameters that allow us to represent a 3D point P as image coordinates P'. The parameters c_x, c_y represent how the image plane and the digital pixel coordinates differ by some translation. As stated before, the image coordinates have their center at C', the image center, while pixel coordinates typically have their origin at the lower left corner of the image called normalized device coordinates (NDC) [8]. To handle this, the 2D points in the image plane and the image are translated by a vector $\begin{bmatrix} c_x & c_y \end{bmatrix}^T$. The mapping is now updated as follows

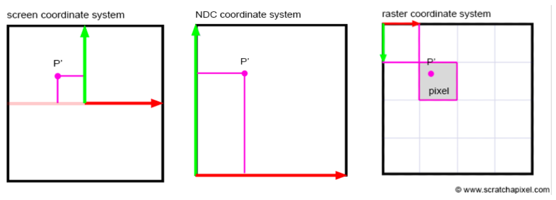

Figure 2.2: Coordinate system conversion from screen space to NDC space to raster space [24].

$$P' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} f\frac{x}{z} + c_x \\ f\frac{y}{z} + c_y \end{bmatrix}. \tag{2.4}$$

Since digital images are expressed in pixels and not in physical measurements, two new parameters k and l are introduced. These parameters can be described as the changes of units on the two axes of the image plane and their units are $\frac{pixels}{cm}$ [8]. The previous mapping is adjusted to be

$$P' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} fk\frac{x}{z} + c_x \\ fl\frac{y}{z} + c_y \end{bmatrix} = \begin{bmatrix} \alpha\frac{x}{z} + c_x \\ \beta\frac{y}{z} + c_y \end{bmatrix}. \tag{2.5}$$

In ray rendering the convention is to use the top left corner as the origin with the y axis pointing in the negative direction, called raster space. The full transformation from screen to raster coordinate space can be seen in Figure 2.2 and the coordinates become $\begin{bmatrix} x & -y & -z \end{bmatrix}^T$.

Since the projection $P \rightarrow P'$ is not linear, it cannot be represented as a matrix-vector product, which would simplify future derivations. To solve this, homogeneous coordinates are introduced. A new coordinate is introduced, so that any point $P' = \begin{bmatrix} x' & y' \end{bmatrix}^T$ instead becomes $P'_h = \begin{bmatrix} x' & y' & 1 \end{bmatrix}^T$, correspondingly, any point $P = \begin{bmatrix} x & y & z \end{bmatrix}^T$ becomes $P_h = \begin{bmatrix} x & y & z & 1 \end{bmatrix}^T$. This is referred to as the homogeneous coordinate system. Using this system, the mapping is described as

$$P'_h = \begin{bmatrix} \alpha x + c_x z \\ \beta y + c_y z \\ z \end{bmatrix} = \begin{bmatrix} \alpha & 0 & c_x & 0 \\ 0 & \beta & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & 0 & c_x & 0 \\ 0 & \beta & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} P_h. \tag{2.6}$$

Since all work henceforth will be done in homogeneous coordinates, the h index is dropped. The above transformation can be broken down further

$$P' = \begin{bmatrix} \alpha & 0 & c_x & 0 \\ 0 & \beta & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} P = \begin{bmatrix} \alpha & 0 & c_x \\ 0 & \beta & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I & 0 \end{bmatrix} P = K \begin{bmatrix} I & 0 \end{bmatrix} P. \tag{2.7}$$

The matrix K is called the camera matrix. There is an extension of K that includes the camera skewness, which means that the angles of the coordinate system axes are not exactly 90 degrees. As most cameras have zero skew, the skew is assumed to be zero in this case. A camera's optical zoom affects the focal length and thus

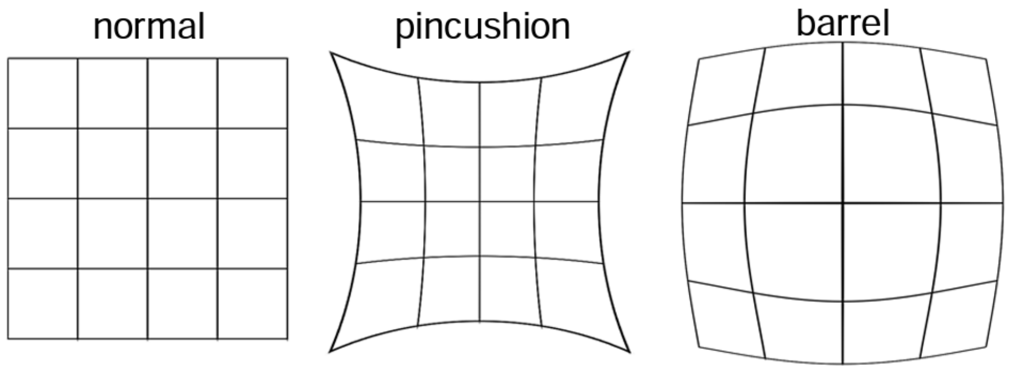

Figure 2.3: Demonstration of how pincushion and barrel distortions affect an image [8].

will affect K. The parameters of K are known as the intrinsic parameters and are unique to each given camera and zoom value.

2.1.2 Lens distortion

If the aperture of a pinhole camera is too small the image becomes dark and if it is too large the image becomes blurry. By adding a lens instead of the pinhole we increase the aperture size and thus the amount of light rays that pass, while also focusing the rays on the screen. A lens solves some of the issues of a pinhole camera but also introduces new ones, such as distortion effects [8].

The most common distortion effect is called radial distortion, which causes the image magnification to decrease or increase as a function of the distance from the optical axis. The radial distortion is caused by certain parts of the lens having variant focal lengths. The two most common forms of radial distortions are pincushion distortion, when the magnification increases, and barrel distortion when the magnification decreases. A demonstration of these effects can be seen in Figure 2.3.

One of the most popular models to represent distortion is the Brown-Conrady model, which is based on the work of Brown [2] and Conrady [5]. OpenCV [1] is an open-source software library for computer vision and machine learning, offering an extensive range of tools for image and video processing. It employs a simplified version of this model,

$$\begin{aligned} r &= \sqrt{x^2 + y^2}, \\ \theta_d &= \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6}, \\ x_{distorted} &= x\theta_d + 2p_1 xy + p_2(r^2 + 2x^2), \\ y_{distorted} &= y\theta_d + p_1(r^2 + y^2) + 2p_2 xy, \end{aligned} \tag{2.8}$$

where x, y represent the coordinates in image space and k, p represent the camera's radial and tangential distortion parameters respectively. Since this model is based on standard lens physics, it does not handle ultra-wide lenses well [1]. A model that does this is the Kannala-Brandt model [12]. The Kannala-Brandt model represents distortion as a function of the angle θ, the angle between the optical axis and the incoming ray.

Let P be a point in 3D coordinates, the coordinate vector of P in the camera reference is $X_c = \begin{bmatrix} x & y & z \end{bmatrix}^T$ and due to homogeneous coordinates, the projection coordinates of P are $(x_p, y_p) = (\frac{x}{z}, \frac{y}{z})$ [29]. Kannala-Brandt then models the fisheye distortion as

$$\theta_d = \theta(1 + k_1\theta^2 + k_2\theta^4 + k_3\theta^6 + k_4\theta^8) \tag{2.9}$$

where $r^2 = x_p^2 + y_p^2$ and $\theta = \arctan(r)$. Using this, the distorted coordinates (x_d, y_d), are calculated as

$$\begin{aligned} x_d &= \tfrac{\theta_d}{r} x_p, \\ y_d &= \tfrac{\theta_d}{r} y_p. \end{aligned} \tag{2.10}$$

The final pixel coordinates (u, v) can then be computed as

$$\begin{aligned} u &= f_x(x_d + \alpha y_d) + c_x, \\ v &= f_y y_d + c_y \end{aligned} \tag{2.11}$$

where f_x, f_y are the focal lengths of the camera along the x and y axes, and α is the skew coefficient which is assumed to be 0. [29].

2.1.3 Extrinsic parameters

To relate points from the world reference system to a camera system we need to perform an additional transformation of a point P. This translation is described by a rotation matrix R and a translation vector t. Given a point P_w in a world reference system, the camera coordinates P, can be computed as

$$P = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} P_w. \tag{2.12}$$

Substituting this into Eq. 2.7, and simplifying gives

$$P' = K \begin{bmatrix} R & t \end{bmatrix} P_w. \tag{2.13}$$

The parameters R and t are known as the extrinsic parameters, they are external and do not depend on the camera. The extrinsic parameters, alternatively referred to as camera poses, describe the position and orientation of a camera in space relative to a scene.

2.2 Alpha compositing

Alpha compositing or alpha blending is a technique used to combine an image with a background to display partial or full transparency. Compositing involves rendering pixels in multiple layers and then merging them into a unified image, referred to as the composite.

If there is a situation where a foreground image should be inserted before a background, the foreground and background pixels need to be blended since each pixel can only display a single color. In alpha composting, a parameter α determines how much each image commits to the color in a pixel, also defined as the opacity of an image [36]. Let α be the opacity of the foreground image and c_f be the pixel color of the foreground. The final pixel color, c, is then determined by merging c_f with the pixel color of the background, c_b, using the formula $c = \alpha c_f + (1 - \alpha)c_b$.

In the case of blending colors of several images, employing a technique that involves breaking down the process into smaller parts is best used. At each step, handle the object closest to the eye while treating the remaining images as a single entity [36]. This strategy can be seen in Figure 2.4, where the color and opacity of the *nth* object as seen from the eye are defined as (c_n, α_n). The final pixel color produced with N pictures can be calculated as follows

$$\begin{aligned} c = {} & \alpha_1 c_1 + (1-\alpha_1)\alpha_2 c_2 \\ & + (1-\alpha_1)(1-\alpha_2)\alpha_3 c_3 \\ & \ldots \\ & + (1-\alpha_1)(1-\alpha_2)\cdots(1-\alpha_{N-1})c_N \\ = {} & \sum_{i=1}^{N}\left(\alpha_i c_i \prod_{j=1}^{i-1}(1-\alpha_j)\right) \end{aligned} \tag{2.14}$$

2.3 Masking

Masking is a technique in computer graphics that is used to conceal parts of an image. It can be used to create certain effects or select certain regions of an image. A mask uses bitwise operations to turn certain bits or pixels on or off. To hide certain pixels in an image and show others, a multiplicative mask can be passed with 0 at the positions of the pixels meant to be hidden and 1 at the positions intended to be kept, this mask is then multiplied with the image to produce a masked image [33]. An example of how image masking works can be seen in Figure 2.5.

During the training of a neural network specifically tasked with rendering images using a pixel-wise loss, a mask is useful to force the network to focus on important parts; this is especially useful if the training images contain parts that are not of interest. As depicted in Figure 3.1, images captured by an arthroscopic camera feature

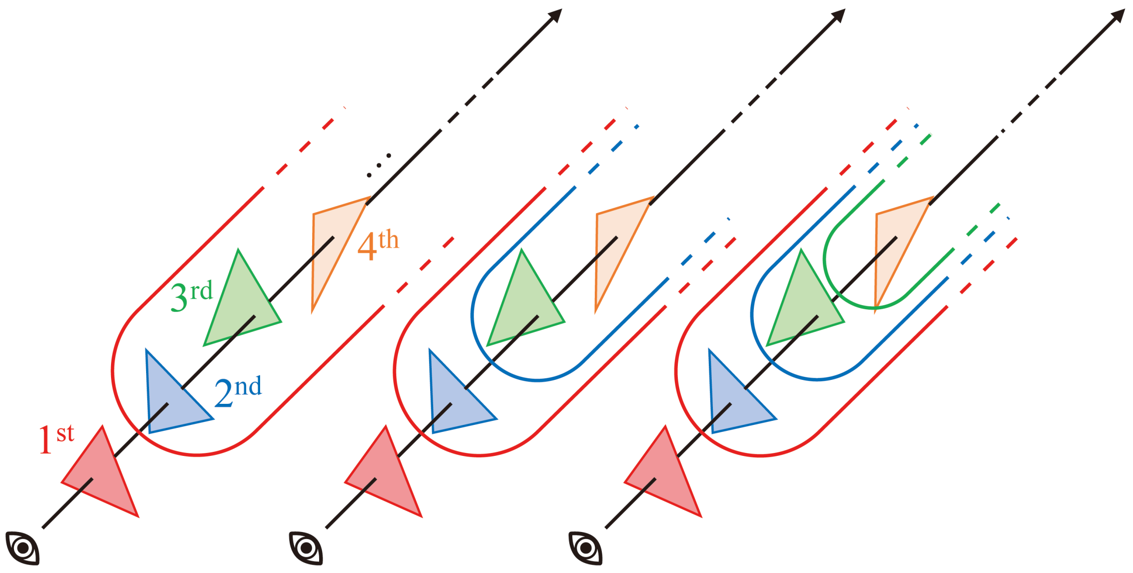

Figure 2.4: Example of alpha compositing during composting of several objects [36].

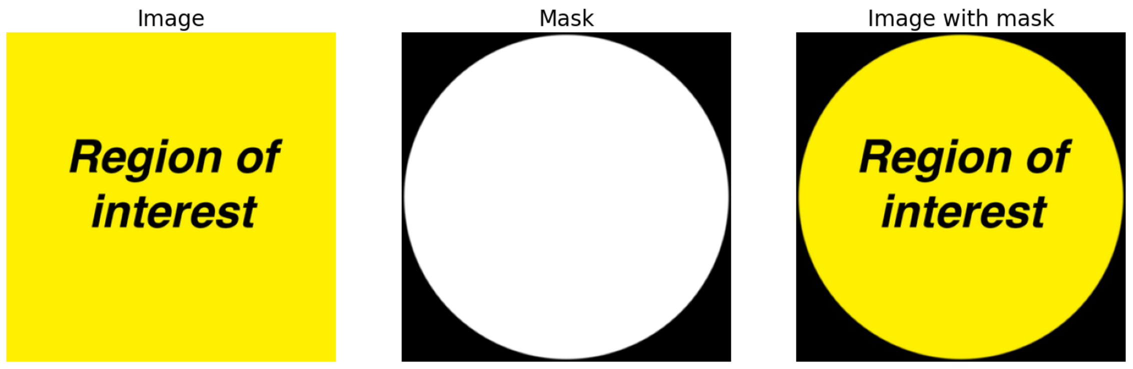

Figure 2.5: Example of image masking.

a black border (pixel values are 0) surrounding the central lens projection. To force the network to learn the relevant parts of an image, a mask can be employed. When the network has produced an image, it will compute a loss between the predicted image and ground truth and then backpropagate this loss through the network to update the network weights. The mask is used to set the values of the non-relevant parts of the loss to zero to not impact the learning of the scene. If the mask were not to be applied, the network would quickly realize that producing images with these non-relevant parts yields a lower loss, and thus will try to do so.

2.4 Camera parameter estimation

If ground truth camera parameters are not known they can be estimated with the help of the techniques detailed in this section.

2.4.1 Structure from Motion and COLMAP

Structure from motion (SfM) is a technique for creating a 3D model from different viewpoints in a 2D plane [23]. It is a way to extract camera poses and intrinsics for a given scene. A popular SfM pipeline used to extract camera parameters is called COLMAP [23].

COLMAP begins by identifying unique patterns or characteristics, referred to as features, in each image, which can then be used to match images to each other. These features should be detectable independently of the geometric changes of the camera to be used between different pictures from varying angles. The algorithm uses these features to match pictures into pairs with shared features. These pairs are verified in the next step through geometric verification, as some of the picture pairs might be false matches because of features that might look similar but represent different points in the scene. Geometric verification tries to map the features of different images through geometric transformations via matrix manipulation of the pictures. First, it estimates the fundamental matrix that maps the points between the two images and uses that to check if at least N_F features correspond between the images. The fundamental matrix describes the relationship between the points in two pictures and is a geometric constraint that arises from the epipolar geometry between the images. Epipolar geometry describes how 3D points project onto the 2D plane and how these relate between two images.

RANSAC [6], a method to remove outliers in data, is used to estimate the fundamental matrix and test the image pairs to see that they have enough inliers since matching is prone to noise. This is an iterative process that tries to find a fitting matrix and then tests the number of matching points. If a sufficient number of points of interest can be assigned to each other through these transformations, the pairs are verified. After these matches have been completed, they are bundled into a graph called the scene graph which consists of the images, the features, and the geometric relationships between pictures.

COLMAP then starts the reconstruction of the scene with this graph. It starts by choosing an image pair as the initial starting point of the reconstruction. The starting pair is essential, as a good starting point determines the robustness of the reconstruction. COLMAP starts by matching images to this first pair by looking for images that share triangulated features with the incorporated images. Choosing the next image is important because one wrong choice can cascade and cause the whole reconstruction to become faulty.

COLMAP implements a multiresolution analysis to score each image match to select the best candidate [23]. This analysis discretizes all images into grids and scores each grid square with a point if there is a feature there. The scoring system is based on both the number of features and the distribution of the features. As each cell in the grid is only counted once, a more uniform distribution is given a higher score. After this, the images with the highest points get incorporated with the images before them; see Figure 2.6 for an example of this.

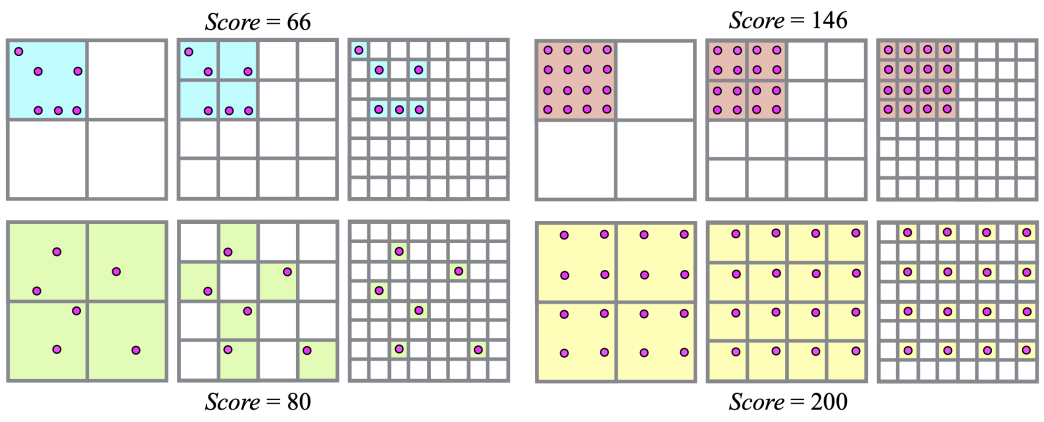

Figure 2.6: COLMAP's multiresolution analysis for image matching, a more uniform spread of features corresponds to a higher score [23].

During the image matching, COLMAP uses the poses of the new images to be able to track each new viewing angle. On each image, COLMAP uses triangulation to make sure that the new image observes the existing scene points. This is decided via the angle needed between the images to align their features; this is calculated as follows

$$\mathbf{X}_{ab} \sim \tau(P'_a, P'_b, E_a, E_b). \tag{2.15}$$

Where $\mathbf{X}_{ab}$ is the triangulated point in 3D space calculated using $\tau(\cdot)$, the chosen triangulation method. The arguments of $\tau(\cdot)$ are $\bar{P}'_a, \bar{P}'_b$ the coordinates of the feature in image a and b and E_a, E_b the world to camera projections with $E = [R^T - R^T t]$. $\mathbf{X}_{ab}$ is then used together with the translation vector of the images to calculate the angle

$$\cos\alpha = \frac{t_a - \mathbf{X}_{ab}}{||t_a - \mathbf{X}_{ab}||_2} \cdot \frac{t_b - \mathbf{X}_{ab}}{||t_b - \mathbf{X}_{ab}||_2}. \tag{2.16}$$

If the triangulation angle, α, is sufficient so that the new image observes the points and the depth is positive, the new image is registered in the model and is added as part of the scene and its corresponding view. The reason for all the redundancy is that features can easily be mistaken for each other. In the early stages of model construction, whenever a new image is registered and triangulated, COLMAP runs local bundle adjustment to make sure errors are taken care of and do not propagate through the whole model. When the model has grown through image registration to a sufficient percentage of all the images, it begins to perform global bundle adjustments over the whole set.

Bundle adjustment is an optimization technique to refine the 3D structure created. The main task is to catch degenerate solutions before they can cascade the model to a point of no return. Degenerate solutions are solutions that might seem viable but do not represent the best solution and can therefore incur errors in the model. Every time the scene is extended with a picture, it might affect the camera parameters calculated for the included pictures. This mostly affects the images closest to the new picture, therefore COLMAP partitions the scene into groups of highly overlapping images $\mathcal{G} = \mathbf{G}_r | r = 1 \dots N_G$ where each group shares camera parameters $\mathbf{G}_r$. Bundle adjustment calculates the re-projection loss that is defined as

$$\mathcal{L} = \sum_j \rho_j(||\pi(\mathbf{G}_r, \mathbf{E}_c, \mathbf{X}_k) - x_j||_2^2), \tag{2.17}$$

where $\pi(\cdot)$ is the function that projects the scene points onto the image plane. $\mathbf{G}_r$, $\mathbf{E}_c$, and $\mathbf{X}_k$ are the extrinsics of the camera group, the pose of the camera for a specific image, and the point parameters, respectively. The loss function, ρ_j, compares $\pi(\cdot)$ to the actual point in image space, x_j. Using this loss, COLMAP updates the 3D model, and the camera poses, and computes the loss again until it converges. After convergence, COLMAP saves the extrinsics for all images and the intrinsics for the cameras used.

2.5 Neural Radiance Fields

Mildenhall et al. [17] describe a method that takes as input a 5D coordinate (spatial location $\mathbf{x} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ and a 2D viewing direction (θ, ϕ)) and outputs an emitted color $\mathbf{c} = (r, g, b)$ and volume density σ. The direction is in practice represented as a 3D Cartesian unit vector $\mathbf{d}$. These 5D coordinates are sampled along camera rays and fed into a multilayer perceptron (MLP) to produce $\mathbf{c}$ and σ. Volume rendering techniques are then used to create images using these values. The depicted scenes are then optimized by minimizing the error between the created and ground truth images [17].

The 5D neural radiance field expresses a scene as the volume density and directional emitted radiance of any point in space. Camera rays are marched through the scene to create a sampled set of 3D points that are then fed into the MLP to predict a color and volume density, a visual representation of this can be seen in Figure 2.7. These points are sampled in a vector $\mathbf{t}$, consisting of distances $t_k \in \mathbf{t}$ from the chosen scene bounds t_n and t_f. To further optimize the performance of NeRF, a positional encoding γ is applied to each of the three coordinate values in $\mathbf{x}$ and the three components of $\mathbf{d}$. The encoding function used is

$$\gamma(p) = \left[\sin(2^0\pi p), \cos(2^0\pi p), \dots, \sin(2^{L-1}\pi p), \cos(2^{L-1}\pi p)\right]^T \tag{2.18}$$

where L is a hyperparameter. The encoding function maps the input to a higher dimensional space to better fit data containing higher frequency variation, allowing the NeRF model to more accurately represent high-frequency geometries and textures [17].

To render the color of any ray in a scene NeRF employs techniques from classical volume rendering to output new 2D views. NeRF achieves this by estimating the integral $C(r)$ in Eq. 2.19 which computes the expected color for a camera ray $\mathbf{r}(t) = \mathbf{r}_o + t\mathbf{r}_d$, where $\mathbf{r}_o$ and $\mathbf{r}_d$ are the ray origin and direction respectively.

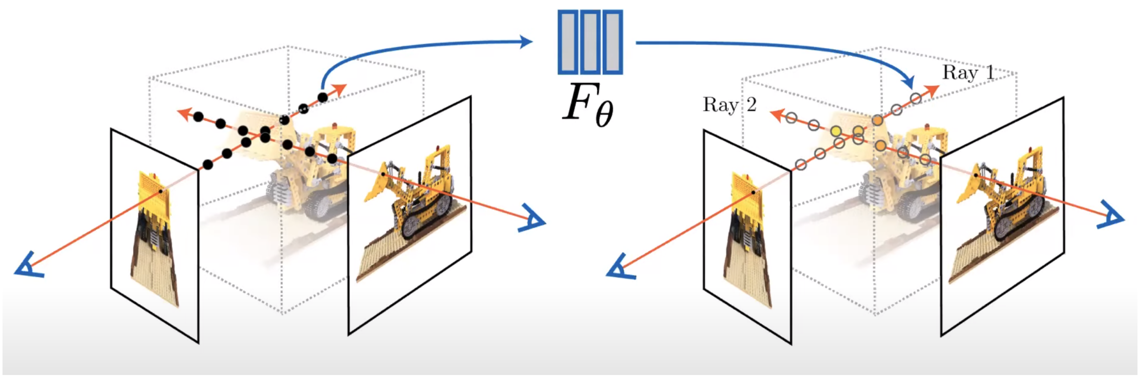

Figure 2.7: Visual representation of how 5D points on camera rays are sampled in NeRF and then fed into an MLP network to produce a color and volume density [17].

$$C(r) = \int_{t_n}^{t_f} T(t,r)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t),\mathbf{r}_d)dt, \text{ where } T(t,r) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right). \tag{2.19}$$

The function $T(t,r)$ denotes the probability that a ray will travel from t_n to t without hitting any other particle. The volume density $\sigma(\mathbf{x})$ is the probability that a ray will terminate at a particle at location $\mathbf{x}$.

$C(r)$ is numerically estimated by quadrature using a stratified sampling approach, where stratified sampling is a sampling method where the samples are divided into subgroups or strata, based on some characteristics, and samples are then taken from each stratum. The purpose is to ensure a proportional representation of each stratum in the samples. Quadrature is a numerical method for approximating a definitive integral and is based on the idea of evaluating a function at a finite number of points and then using a weighted sum to approximate the integral. The interval $[t_n, t_f]$ is partitioned into N evenly spaced bins, and then a sample t_i is drawn uniformly at random from each bin

$$t_i \sim \mathcal{U}\left[t_n + \frac{i-1}{N}(t_f - t_n), t_n + \frac{i}{N}(t_f - t_n)\right]. \tag{2.20}$$

A stratified sampling approach results in the MLP being evaluated over a continuum of positions during the optimization stage even though a discrete set of samples is being used. These samples are then used to estimate $C(\mathbf{r})$ using a quadrature rule from [16] by Max,

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))\mathbf{c}_i, \text{ where } T_i = \exp\left(-\sum_{j=1}^{i-1}\sigma_j\delta_j\right) \tag{2.21}$$

where $\delta_i = t_{i+1} - t_i$ is the distance between neighbouring samples. The process of computing $\hat{C}(\mathbf{r})$ from the given set of $(\mathbf{c}_i, \sigma_i)$ values is straightforwardly differentiable. It simplifies to conventional alpha compositing when utilizing alpha values $\alpha = 1 - \exp(-\sigma_i\delta_i)$.

For each ray, there are N samples. The first $N-1$ samples can be considered foreground objects with opacity $\alpha_i := 1 - \exp(-\sigma_i \delta_i)$ and color c_i, where the Nth sample is the background; the blended pixel value then is computed using Eq. 2.14,

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} \left(\alpha_i c_i \prod_{j=1}^{i-1} (1 - \alpha_j) \right) = \sum_{i=1}^{N} \alpha_i c_i T_i. \tag{2.22}$$

NeRF makes use of two networks to represent scenes, a “coarse” network and a “fine” network. This two-network structure allows for a much more efficient rendering of scenes, the coarse network gives a rough estimate of the density and radiance of the scene, allowing NeRF to quickly converge to a solution, while the fine network refines the coarse estimate and produces high-quality results. First, a sample of N_c locations is sampled using stratified sampling and the coarse network is evaluated at these locations using Eqs. 2.20 and 2.21. Using the output of the coarse network, a more informed sampling of points is performed along each ray. This is done by rewriting Eq. 2.21 as a weighted sum of all the colors sampled c_i, along the ray

$$\hat{C}_c(\mathbf{r}) = \sum_{i=1}^{N_c} w_i c_i, \quad w_i = T_i(1 - \exp(-\sigma_i \delta_i)). \tag{2.23}$$

These weights are then normalized to $\hat{w}_i = w_i / \sum_{i=1}^{N_c} w_j$ producing a piecewise-constant probability density function (PDF) along the ray, and from this distribution, a new sampling of N_f samples is performed using inverse transform sampling. Inverse transform sampling is a method of generating random samples from any probability distribution given its cumulative distribution function (CDF). It works by inverting the CDF to find the corresponding value of the random variable for a given random number from the uniform distribution. Evaluation of the fine network is performed at the union of sample sets, and the final rendered color of the ray $\hat{C}_f(\mathbf{r})$ is computed using Equation 2.21 using all samples $N_c + N_f$. This procedure produces more samples from regions that are expected to contain more visible content. The authors of NeRF refer to this process as hierarchical volume sampling and in Figure 2.8 an illustration of this process can be seen.

At each training step, a batch of camera rays from all pixels in the dataset are randomly selected, and hierarchical sampling is used to sample a set from both the coarse and fine networks. The above-explained volume rendering procedure is used to render the color from each ray from both the sample sets. The loss is then calculated as the total squared error between the rendered and ground truth pixel colors for both of the two renderings

$$\mathcal{L} = \sum_{\mathbf{r} \in \mathcal{R}} \left[\|\hat{C}_c(\mathbf{r}) - C(\mathbf{r})\|_2^2 + \|\hat{C}_f(\mathbf{r}) - C(\mathbf{r})\|_2^2 \right] \tag{2.24}$$

where $\mathcal{R}$ is the set of rays in each batch, $C(\mathbf{r})$, being the ground truth RGB values from ray $\mathbf{r}$ while $\hat{C}_c$ and $\hat{C}_f$ are the predicted RGB values from the coarse and fine volumes respectively [17].

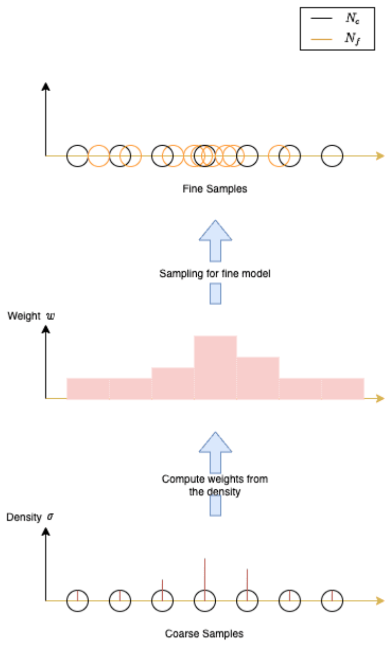

Figure 2.8: Illustration of how hierarchical volume sampling is performed in NeRF [7].

2.5.1 NeRF in the Wild

In [15] Martin-Brualla et al. present an extension of the original NeRF method, which they call NeRF in the Wild (NeRF-W), which is capable of modeling real-world phenomena in uncontrolled images [15]. NeRF-W is trained on a data set of internet images of famous landmarks. NeRF-W differs from NeRF in that it does not assume consistency in its input views, where consistency means that a point in space has the same intensity in two images given an observation from the same viewing direction and position. The phototourism data set used to train NeRF-W violates this assumption by two phenomena, photometric variation, and transient objects.

Photometric variation; time and atmospheric conditions affect the illumination of objects in a given scene. This issue is enhanced by camera settings and post-processing steps, such as white balance, variation in auto-exposure settings, and tone mapping [15]. **Transient objects**; internet photographs of landmarks rarely exist without moving objects or occluders such as tourists blocking the view.

To handle variable lighting and photometric post-processing, NeRF-W extends Eq. 2.21 from NeRF, here referred to as $\mathcal{R}(\mathbf{r}, \mathbf{c}_i, \sigma)$, by replacing the image-independent radiance $\mathbf{c}(t)$ with an image-dependent radiance, $\mathbf{c}_i(t) = MLP_{\theta_2}\left(\mathbf{z}(t), \gamma_{\mathbf{d}}, \boldsymbol{\ell}_i^{(a)}\right)$. NeRF-W is composed of three networks; a base component, MLP_{θ_1}, a static component, MLP_{θ_2} and a transient component, MLP_{θ_3}. The input of the second network (MLP_{θ_2}) is $\mathbf{z}(t)$, the output of the first network (MLP_{θ_1}), along with the encoding function for the viewing direction, $\gamma_{\mathbf{d}}(\mathbf{d})$, and finally $\boldsymbol{\ell}_i^{(a)}$ is an appearance embedding vector of image $\mathcal{I}_i$.

By using these appearance embeddings as input to the part of the network that emits color, the NeRF-W model is given some freedom in varying the emitted radiance of a scene while at the same time ensuring that the 3D geometry (predicted by MLP_{θ_1}) stays static and the same for all images.

The transient objects phenomenon is managed by defining the MLP that emits color in NeRF as the “static” head of the NeRF-W model, in addition, a “transient” head is added that emits its own color and density enabling the model to compose images with occluders without introducing artifacts into the scene. The transient head emits a field of uncertainty, allowing the model to accommodate the reconstruction loss to ignore pixels and 3D locations deemed unreliable since they are likely to contain occluders. The transient head is built on top of the volume rendering formulation in Eq. 2.21 by including transient counterparts $\sigma_i^{(\tau)}, \mathbf{c}_i^{(\tau)}$:

$$\begin{gathered}\hat{\mathbf{C}}_i(\mathbf{r}) = \textstyle\sum_{i=1}^{N} T_i\Big((1-\exp(-\sigma_i\delta_i))\mathbf{c}_i + (1-\exp(-\sigma_i^{\tau}\delta_i))\mathbf{c}_i^{\tau}\Big) \\ \text{where } T_i = \exp\Big(-\textstyle\sum_{j=1}^{i-1}(\sigma_j+\sigma_j^{\tau})\delta_j\Big).\end{gathered} \tag{2.25}$$

The expected color of $\mathbf{r}(t)$ becomes the alpha composite of the static and transient components. The uncertainty of the color $\mathbf{C}_i(\mathbf{r})$ is modeled with an isotropic normal distribution, with variance $\beta_i(\mathbf{r})^2$ and mean $\hat{\mathbf{C}}_i(\mathbf{r})$. The variance is rendered identically to the color via alpha compositing by using the transient density σ_i^{τ}:

$$\hat{\beta}_i(\mathbf{r}) = \mathcal{R}(\mathbf{r}, \beta_i, \sigma_i^{\tau}). \tag{2.26}$$

To let the transient component of a scene vary across images, each training image $\mathcal{I}_i$ is given a second embedding $\boldsymbol{\ell}_i^{(\tau)} \in \mathbb{R}^{n^{(\tau)}}$, which is then fed to the transient MLP together with the output of the first network, $\mathbf{z}(t)$:

$$\Big[\sigma_i^{(\tau)}(t), \mathbf{c}_i^{(\tau)}(t), \tilde{\beta}_i(t)\Big] = MLP_{\theta_3}\Big(\mathbf{z}(t), \boldsymbol{\ell}_i^{(\tau)}\Big), \tag{2.27}$$

$$\beta_i(t) = \beta_{min} + \log\Big(1+\exp\Big(\tilde{\beta}_i(t)\Big)\Big). \tag{2.28}$$

An illustration of the model architecture of NeRF-W can be seen in Figure 2.9.

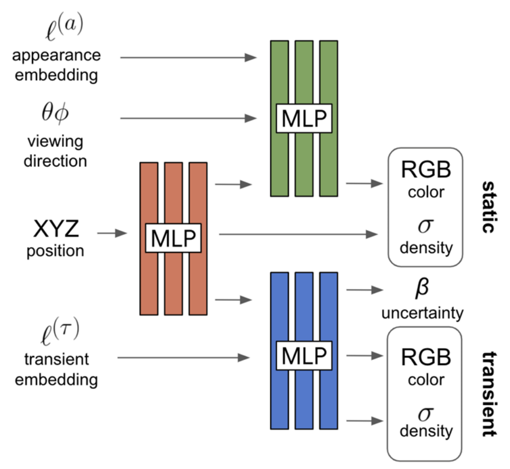

Figure 2.9: Overview of the NeRF-W model architecture. Given an appearance embedding, viewing direction, 3D position, and transient embedding NeRF-W produces static and transient colors alongside a measure of uncertainty [15].

The loss for a ray $\mathbf{r}$ in image i with ground truth color $\mathbf{C}_i(\mathbf{r})$ is

$$L_i(\mathbf{r}) = \frac{\|\mathbf{C}_i(\mathbf{r}) - \hat{\mathbf{C}}_i(\mathbf{r})\|_2^2}{2\beta_i(\mathbf{r})^2} + \frac{\log \beta_i(\mathbf{r})^2}{2} + \frac{\lambda_u}{N}\sum_{i=1}^{N} \sigma_i^{(\tau)}. \tag{2.29}$$

The first two terms of the loss are the shifted negative log-likelihood of $\mathbf{C}_i(\mathbf{r})$ given a normal distribution with mean $\hat{\mathbf{C}}_i(\mathbf{r})$ and variance $\beta_i(\mathbf{r})^2$. The third term is an L_1 regularization term with a multiplier λ_u on the transient density. This dissuades the model from explaining away static phenomena using transient density.

As NeRF, NeRF-W optimizes two networks at the same time, a fine model as described above and a coarse model that only uses the latent appearance embeddings. Alongside the parameters θ, NeRF-W optimizes per-image appearance embeddings $\{\boldsymbol{\ell}_i^{(a)}\}_{i=1}^N$ and transient embeddings $\{\boldsymbol{\ell}_i^{(\tau)}\}_{i=1}^N$. The final loss function of NeRF-W is,

$$\sum_{ij} L_i(\mathbf{r}_{ij}) + \frac{1}{2}\|\mathbf{C}(\mathbf{r}_{ij}) + \hat{\mathbf{C}}_i^c(\mathbf{r}_{ij})\|_2^2 \tag{2.30}$$

where λ_u, β_{min} and embedding dimensionalities $n^{(a)}$ and $n^{(\tau)}$ are hyperparameters.

2.5.2 Self-Calibrating NeRF

Jeong et al. [10] present Self-Calibrating NeRF (SCNeRF), an online plugin for other NeRF models that aims to solve both the positioning problem and the problem of calibrating a camera's parameters. The algorithm learns the geometric parameters with the help of NeRF's representation of rays. Since most cameras have non-linear distortion, the camera parameters must be calibrated or estimated before training. The SCNeRF algorithm is based on the unprojection of pixels to rays

$$
\begin{aligned}
r_d &= RK^{-1}p, \\
r_o &= t.
\end{aligned}
\tag{2.31}
$$

Where r_d, r_o and t are the projected ray direction, ray origin, and the point of that origin in 3D space, respectively. R is the extrinsics of the camera and K is its intrinsics. With the help of higher-order differentials, the distortion is taken into account. The algorithm continuously optimizes the camera parameters during run-time.

The algorithm is built on three main parts; differentiable self-calibrating cameras, geometric and photometric consistency, and optimizing geometry and camera.

The first part of the algorithm is built on the pinhole camera model as described in Eq. 2.7 with the K matrix being used to map pixels to rays in the image plane. Due to the intrinsic parameters being non-convex, meaning that there exist a lot of minima, the K matrix is decomposed into

$$
K = \begin{bmatrix} \alpha_0 + \Delta\alpha & 0 & c_{x0} + \Delta c_x & 0 \\ 0 & \beta_0 + \Delta\beta & c_{y0} + \Delta c_y & 0 \end{bmatrix}. \tag{2.32}
$$

The zero-terms in the matrix are the initialized camera values that are provided to SCNeRF, either from COLMAP estimations or initialized at half the height and width of the picture, and the Δ-terms are the residuals that are used to improve the intrinsics of the camera, the extrinsic matrix is decomposed in the same way. Trying to directly learn these parameters will result in non-orthogonal matrices and break the pose estimations. Therefore, the rotation matrix is extended to a 6-vector representation proposed by Zhou et al. in [38]. The rotations and translations then become

$$
\begin{aligned}
R &= f(a_0 + \Delta a), \\
\mathbf{t} &= t_0 + \Delta t,
\end{aligned}
\tag{2.33}
$$

where $f(\cdot)$ is the mapping from 6-vector representation back to 3-vector representation and $a_0 + \Delta a$ is the extrinsic parameters represented in 6D space. The algorithm uses the decomposed matrices to unproject the pixels to ray vectors and is rewritten as

$$
\begin{aligned}
r_d = RK^{-1}p &\rightarrow r_d = f(a_0 + \Delta a) * (K + \Delta K)^{-1}p, \\
r_o = t &\rightarrow r_o = t_0 + \Delta t.
\end{aligned}
\tag{2.34}
$$

The rays are therefore a sum of the initial values and the residuals and the algorithm can thus use the gradients of the rays to optimize the residuals. It does not optimize the initial values. Since the camera contains a lens that distorts the edges of the image, SCNerF extends the camera model to include radial distortions. It uses the fourth-order radial distortion model, which is expressed as

$$\begin{aligned} n &= ((\mathbf{p}_x - c_x)/f_x, (\mathbf{p}_y - c_y)/f_y, 1), && (2.35)\\ d &= (1 + \mathbf{k}_1 n_x^2 + \mathbf{k}_2 n_x^4, 1 + \mathbf{k}_1 n_y^2 + \mathbf{k}_2 n_y^4), && (2.36)\\ \mathbf{p}' &= (\mathbf{p}_x d_x, \mathbf{p}_y d_y, 1), && (2.37)\\ \mathbf{r}_d &= RK^{-1}\mathbf{p}', && (2.38)\\ \mathbf{r}_o &= t, && (2.39) \end{aligned}$$

where $\mathbf{k} = [k_1 + z_{k1}, k_2 + z_{k2}]$ are the radial distortion parameters , $\mathbf{p}$ the pixel coordinates, and d being the radial distortion.

The optical aberrations of real-world camera models are impossible to represent mathematically. Therefore, this aberration noise is modeled with raxel parameters that consist of the camera parameters; coordinates for the pixels, and the corresponding ray's position and directions [18]. The raxel parameters used in SCNeRF are the ray residuals, the offset of the rays in 2D space, z_d,z_o and they are described as

$$\begin{aligned} z_d &= \Delta r_d(p),\\ z_o &= \Delta r_o(p), \end{aligned} \qquad (2.40)$$

where p is the image coordinate. The rays are then formulated as

$$\begin{aligned} r_d' &= r_d + z_d,\\ r_o' &= r_o + z_o. \end{aligned} \qquad (2.41)$$

To extract the ray distortion parameters, bilinear interpolation is run on the images

$$z_d = \sum_{x=p_x}^{p_x+1} \sum_{y=p_y}^{p_y+1} (1 - |x - p_x|)(1 - |y - p_y|). \qquad (2.42)$$

Bilinear interpolation calculates the values for points in between given points based on their distance to x and y. Figure 2.10 describes the generation of the rays, both in direction and offset.

To calibrate the camera parameters, SCNeRF incorporates geometric and photometric consistency, since it gives additional constraints to calibrate. Geometric consistency is achieved through the projected ray distance (PRD). PRD is measured on the difference between two rays that meet at the same 3D point in the scene; see Figure 2.11. In theory, they should align, but with an imperfect camera model, they can be misaligned. This difference is used as a novel loss parameter. Let there be two pictures, image A and image B, viewing the same points in space and let a point on the ray from image A be $x_A(t_A) = r_{o,A} + t_A r_{d,A}$ and the same point for image B, $x_B(t_B) = r_{o,B} + t_B r_{d,B}$.

$$\begin{tikzcd}
 & & z_k \arrow[d] & & \mathbf{p} \arrow[d, no head] & & & \\
 & k \arrow[r] & + \arrow[r] & \mathbf{k} \arrow[r] & {} \arrow[d, "\text{Radial Distortion}"] & & & \\
 & z_K \arrow[d] & & & \mathbf{p}' \arrow[d] & & z_{\mathbf{r}_d} \arrow[d] & \\
K \arrow[r] & + \arrow[r] & K \arrow[rr] & & R \cdot K^{-1} \cdot \mathbf{p}' \arrow[r] & \mathbf{r}_d \arrow[r] & + \arrow[r] & \mathbf{r}'_d \\
R|t \arrow[r] & + \arrow[r] & R|t \arrow[rr] \arrow[urr] & & t \arrow[r] & \mathbf{r}_o \arrow[r] & + \arrow[r] & \mathbf{r}'_o \\
 & z_{R|t} \arrow[u] & & & & & z_{\mathbf{r}_o} \arrow[u] &
\end{tikzcd}$$

Figure 2.10: Diagram illustrating the process by which rays are constructed from various parameters within SCNeRF [10].

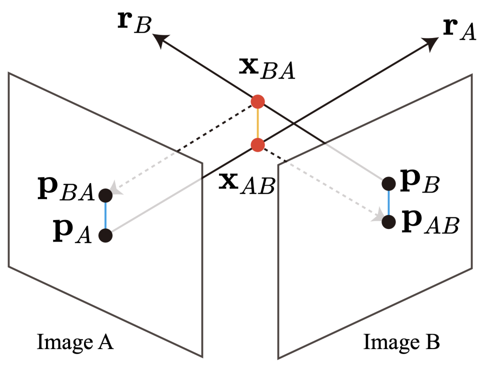

Figure 2.11: Representation of the PRD calculation. Given two images and a point in space the distance between them can be calculated [10].

The distance between these points, $\hat{d} = \overline{x_A x_B}$, is given by the following calculation,

$$\hat{d} = \frac{|(r_{A,o} - r_{B,o}) \cdot (r_{A,d} \times r_{B,d})|}{|r_{A,d} \times r_{B,d}|}. \tag{2.43}$$

This is the distance in space between these points but as features further away from the camera are impacted more by distortion and non-perfect models, SCNeRF reprojects these points down to image space to measure the distance on the 2D plane where this impact is normalized. This is done via

$$d_\pi = \frac{||\pi_A(\mathbf{x}_B) - \mathbf{p}_A|| + ||\pi_B(\mathbf{x}_A) - \mathbf{p}_B||}{2}, \tag{2.44}$$

where $\pi(\cdot)$ is the function that projects the features to the image plane and $\mathbf{p}_A, \mathbf{p}_B$ are the pixels in the image plane corresponding to the features. The distance is represented in Figure 2.11. As this distance should be zero it is used as a new loss function to refine the parameters that the rays are built upon.

The photometric consistency ensures that the colors align with the viewpoint, making sure that the color and occupancy are viewable from the camera position. Using NeRF's radiance fields, SCNeRF reconstructs the color and occupancy in the 3D space. This representation is differentiable for both color and position and can thus be used to refine the parameters. During rendering, a ray is parameterized with the variables in Eqs. 2.32-2.34 and Eq. 2.42.

The loss of the network is computed using the volumetric rendering output $\hat{C}(r)$, the ground truth $C(p)$ and the images pixel coordinates $\mathcal{I}$,

$$\mathcal{L} = \sum_{p \in \mathcal{I}} ||C(p) - \hat{C}(r(p))||_2^2. \tag{2.45}$$

The gradients of the parameters can be obtained by differentiating this function. For example, the intrinsics gradient that is used to calibrate the camera can be represented with

$$\frac{\partial \mathcal{L}}{\partial \Delta K} = \frac{\partial \mathcal{L}}{\partial r} \left(\frac{\partial r}{\partial r_d} \frac{\partial r_d}{\partial \Delta K} + \frac{\partial r}{\partial r_o} \frac{\partial r_o}{\partial \Delta K} + \frac{\partial \mathcal{L}}{\partial r_d} \frac{\partial r_d}{\partial \Delta K} \right). \tag{2.46}$$

To execute the algorithm the network needs to first be at a point where the geometric representation is fine enough to do the geometric consistency checks. Therefore SCNeRF allows the network to train to obtain a coarse geometry since it will allow the more complex parameters, such as distortion, to be optimized at a better starting point to avoid degenerate solutions of the parameters. After learning the geometry, the algorithm introduces the radial distortion parameters, nonlinear noise of the direction rays, and origin rays. This process is represented in Algorithm 2 in Appendix A.

2.6 Comparison metrics

In the sections below we describe the metrics used to determine how well a model performs by comparing the rendered images to ground-truth images.

2.6.1 Peak Signal-to-Noise Ratio

Peak signal-to-noise ratio (PSNR) is a rudimentary way to determine the correlation between two pictures [9]. It does this on a pixel-by-pixel basis and therefore does not take into concern any features. PSNR is described in Eq 2.47

$$PSNR = 10 * \log_{10}(\frac{MAX_I^2}{MSE}), \tag{2.47}$$

where MAX_I is the maximum possible pixel value of the image, if the pixels are represented by 8 bits, this value is 255. MSE is the mean squared error between two images and is defined as follows:

$$\text{MSE} = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \|f(i,j) - g(i,j)\|^2 \tag{2.48}$$

where $f(i,j)$ and $g(i,j)$ are the ground truth and modeled image pixel values, respectively. The PSNR will be approaching infinity when the error is small, a perfect picture will thus result in infinite PSNR. Therefore a larger PSNR value is better.

2.6.2 Structural Similarity Index

Structural Similarity Index (SSIM) is a way of comparing perceptual differences between images[9]. SSIM takes into account features, therefore providing a better value for human perception of likeness [31]. Given two images f, g, SSIM is described as the product of luminance $l(f, g)$, contrast $c(f, g)$ and structure $s(f, g)$,

$$SSIM(f, g) = l(f, g)c(f, g)s(f, g). \tag{2.49}$$

The luminance component, $l(f, g)$ is defined as

$$l(f, g) = \frac{2\mu_f \mu_g + C_1}{\mu_f^2 + \mu_g^2 + C_1}, \tag{2.50}$$

where $\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i$, is the mean intensity of image x. The constant $C_1 = (K_1 L)^2$, is added to prevent instability when $\mu_f^2 + \mu_g^2$ is close to zero, $K_1 \ll 1$ is a small constant and L is the dynamic range of the pixel values [31]. In an 8-bit image, this value is 255. The contrast component $c(f, g)$ has a similar form

$$c(f, g) = \frac{2\sigma_f \sigma_g + C_2}{\sigma_f^2 + \sigma_g^2 + C_2}, \tag{2.51}$$

where the standard deviation $\sigma_x = \left(\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2\right)^{\frac{1}{2}}$, is used as an estimate of the contrast of an image, $C_2 = (K_2 L)^2$, and $K_2 \ll 1$. The final component of SSIM, the structure $s(f, g)$, can be described as

$$s(f, g) = \frac{\sigma_{fg} + C_3}{\sigma_f + \sigma_g + C_3}. \tag{2.52}$$

Here, a constant $C_3 = \frac{C_2}{2}$ is added as was done in the luminance and contrast components. The covariance of x, y, σ_{xy} can be estimated as

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y). \tag{2.53}$$

This yields the following form of SSIM,

$$SSIM(f, g) = \frac{(2\mu_f \mu_g + C_1)(2\sigma_{fg} + C_2)}{(\mu_f^2 + \mu_g^2 + C_1)(\sigma_f^2 + \sigma_g^2 + C_2)}. \tag{2.54}$$

SSIM will always produce a number in the interval [0,1] where 0 equates to no correlation and 1 meaning the images compared are identical, and thus a higher SSIM score is preferred.

2.6.3 Learned Perceptual Image Patch Similarity

Zhang et al. [37] propose the Learned Perceptual Image Patch Similarity (LPIPS) metric as a more accurate way to determine the likeness of different pictures compared to SSIM and PSNR. LPIPS measures the perceptual similarity between two images. It is based on activations from a pre-trained convolutional neural network

(CNN), that has been trained to perform image classification.

LPIPS extracts the CNN activations for both image patches. The activations represent the CNN's "understanding" of the images, these activations are more closely related to human perception than traditional metrics such as SSIM. After extraction of the activations, LPIPS computes a distance metric between them. This metric is designed to capture the perceptual difference between the image patches, a lower score indicates that the two patches are more similar. LPIPS is very effective at this task [37].

3 Method

The original NeRF method [17] assumes rigid objects, which limits its ability to handle non-rigid scenes effectively, and it also requires near-perfect conditions of the training data, such as a well-lit scene and correct camera parameters. Therefore, we have compared various NeRF models on the available data sets to determine which model performs the best.

The methodology for this approach was as follows; first, the data had to be processed to decide what should be used for training and testing. Some of the models intended to be compared required additional preprocessing steps; this process is described below in Section 3.1.

To be able to choose a model or models to further develop, we compared a few models with each other. In Section 4.1 we outline which models were chosen for comparison and why, along with how we performed the experiments.

After the experimental stage was completed, the model that showed the best results was chosen as the starting point for further development. After this, we entered the development stage, where we developed our model to further improve the results achieved in the previous stage. The development was carried out in an iterative manner, where we continuously evaluated our model and made changes depending on the results.

3.1 Data processing

In AI-driven research and applications, the data used is of great importance. Therefore, a big part of the task was to process the available data into usable parts. The available data of this thesis consists of videos from ACL surgeries performed on patients with various amounts of interfering elements. They are filmed using an arthroscope, an inspection instrument consisting of an image sensor, optical lens, light source, and a mechanical device [26]. The arthroscope records videos that are spatially distorted due to the wide angle of the lens [25], creating a distorted view of the knee's interior. The data consisted of videos recorded in both 60 and 30 fps. The models were trained on image frames extracted from the data. The frames were processed to be 1080x1080 pixels; however, to speed up training, we downsampled the images by a factor of 2 or 3 depending on the model. A mask had been placed in the alpha channel of the images. This mask was extracted and used to mask predictions of the models tested, as explained in Section 2.3.

Another data processing step that was performed on the frames was that they had all been rotated to face the same direction. The original images contained a “pointy” part, as can be seen in Figure 3.1, this part moves around in the videos and to combat this, all frames were rotated so that this part always pointed in the same direction.

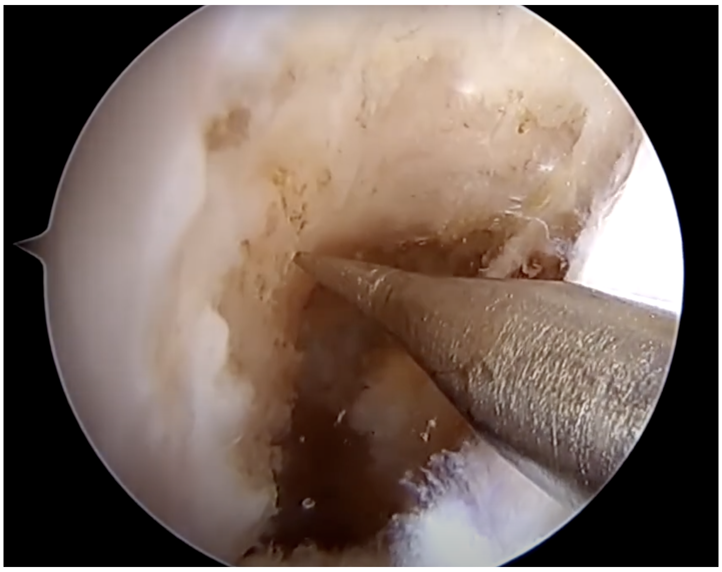

Figure 3.1: A frame from ACL surgery, captured using an arthroscope [4].

Although all data were from actual ACL surgeries, the videos could differ greatly from each other. Some surgeons chose to burn away organic matter in the knee to improve visibility while others did not, some frames had surgical instruments blocking the view of the knee, and the knee could deform during some procedures. Using these assumptions as a base, we split the data into different conditions, to test the models on data of varying conditions for a greater representation of different scenarios present in surgeries.

The data conditions depend on certain criteria, such as whether organic matter or instruments cloud the field of view, the amount of knee deformation, and whether the lighting affects visibility. In Table 3.1 the condition names and their corresponding criteria requirements are presented. The criteria used are displayed below.

- Was there any organic matter clouding the view?
- Were there any surgical tools present?
- Was visibility bad?
- Was there any deformation of the knee?

Before the experimental stage, a manual processing step was performed where each available video was classified by the above conditions. After this manual processing was performed, we chose one video of each condition to be used in training the

Condition name	**Criteria requirements**
Ideal conditions (**IC**)	None of the criteria fulfilled
Okay conditions (**OC**)	1-2 criteria fulfilled
Bad conditions (**BC**)	2-3 criteria fulfilled

Table 3.1: Data conditions and their corresponding criteria requirement.

models. The videos are of varying length and since most NeRF models are quite slow to train we decided not to train on the full data sets but instead extract around 500-600 evenly spaced frames for each video.

An example view of an ACL reconstruction surgery can be seen in Figure 3.1. Due to ethical and privacy concerns, this frame is not from the IC, OC, or BC data set, but sourced from [4]. These ethical reasons prohibit us from showing any of the data or renderings of these data sets. To combat this issue, we were given access to an open source data set that did not fall under the same restrictions, which we will refer to as the Open source data set, if we were to classify the Open source data set using the criteria shown in Table 3.1 it would fall under the BC category. All images and renderings of knees in the thesis will thus be sourced from this data set if not stated otherwise.

To extract camera poses and camera parameters, the SfM pipeline COLMAP (2.4.1) was used on the scenes chosen for training, and the estimated camera parameters for each data set were used during the training of all models. COLMAP allows users to choose from a predefined set of camera models to estimate the intrinsic parameters of a camera. We used the OpenCV model, which is based on the pinhole camera model and models radial coefficients suited for distortions of arthroscopic cameras [27].

An important factor for the test set is that they contained novel views that the model had not seen in the training images. The data sets have quite a narrow camera path and view of the knee, due to how the camera moves in the knees and the general constrained space. This meant that the camera poses of many frames were very close to each other in world space, meaning that some frames were almost identical. To combat this, we decided to filter images that were too close to each other. From each data set of 500-600 frames, every 20th image was chosen as a viable test image; for each of these test images, we checked the distance to all other images and if the distance was below a certain threshold, we left those images out of the training set. Given that each of the data sets had different camera paths, and that we wanted a similar size of training and test set for all data sets, the thresholds of the specific data sets were chosen to minimize this difference while also removing similar images. The thresholds for the different data sets can be seen in Table 3.2. As COLMAP did not provide any unit for the poses it was left out in the table but we calculated the max distance between the poses for each data set and show how the threshold corresponds to this distance in Table 3.2.

Data set	Minimum distance threshold
IC	0.01 (1% of max distance)
OC	0.2 (17% of max distance)
BC	0.1 (9% of max distance)
Open source	0.15 (8% of maximum distance)

Table 3.2: Minimum distance threshold used for the various data sets when filtering images too close to the test images.

In Figure 3.2 we can see the camera path and image positions for the IC data set. We see that there is a large concentration of images at the start and end of the camera path. In the end, we were left with around 400-500 training images and around 25 test images for each data set. Given the nature of the data, the images were still quite close to each other but this filtering allowed us to be more confident in evaluating views not seen before.

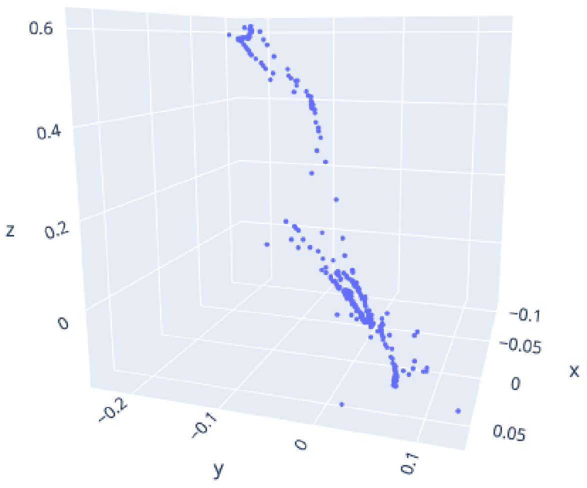

Figure 3.2: Plot over camera poses for the IC data set. Each node represents an image and its position in world space.

3.2 Model development

Based on the results that are presented in 4.1 we chose to use NeRF-W as our baseline model, due to it outperforming both NeRF and SCNeRF. In the following section, we detail the different modifications we added to NeRF-W in the development of our model.

3.2.1 Ray distortion

In the original NeRF-W model, the distortion of the radial lens was not taken into account, which would not accurately model the interior of the knees. As described in Section 2.1.2, lens distortion can be modeled with the help of the polynomials described in Eqs. 2.8 or 2.9. The arthroscopes used in ACL surgeries have a lens distortion, but as discussed in [25], this distortion is pure radial distortion and therefore we decided to use the OpenCV model that is based on Brown-Conrady.

NeRF-W derives the ray directions r_d, based on the camera's intrinsic parameters shown in Eq. 3.1, it does not account for radial distortion, and therefore would not create realistic renderings of knees.

$$r_d \leftarrow [(x - c_x)/f_x, -(y - cy)/f_y, -1] \tag{3.1}$$

To allow for more realistic renderings, we present a new approach to calculating ray directions in Algorithm 1. Using OpenCV's `initUndistortRectifyMap` function to compute the mapping from the distorted image (x, y) to an undistorted image (x', y'). The calculations for the maps can be seen in Eq. 3.2. A map is a description of how the different pixels correspond between the distorted image and the undistorted one.

The function `initUndistortRectifyMap` requires a new intrinsic matrix $[f'_x, f'_y, c'_x, c'_y]$ for the undistorted picture which we obtain from the OpenCV function `getOptimalNewCameraMatrix`. This function estimates the set of intrinsics that describe the camera as if the radial distortion did not exist. The arguments to the function are the intrinsics of the camera, the radial distortion parameters, and α, a value ranging from 0 to 1, which allows control of the pixels in the resulting pictures since undistortion might lose information in the edges of the images as the parameters might force the pixels out to the corners. An α value of 1 keeps all the original pixels in the picture, and a value of 0 gives the best representation of the undistortion. For example, looking at the barrel distortion in Figure 2.3, if you straighten the lines the resulting image will become bigger than the original image. We chose an α value of 0 to give the most accurate image of the knee to the model. This is represented in Figure 3.3. The relevant calculations for these new maps are as follows

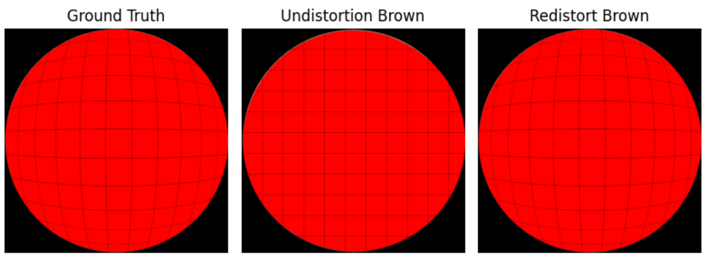

Figure 3.3: An example of the impact of undistortion on the images with the distortion parameters from the IC dataset. The first picture represents the ground truth picture, then the second an undistorted version, and the third the redistortion of the undistorted picture resulting in the ground truth again.

$$\begin{aligned} x' &= (u - c'_x)/f'_x, \\ y' &= (v - c'_y)/f'_y, \\ r^2 &= x'^2 + y'^2, \\ \theta &= \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2}, \\ x'' &= x'\theta, \\ y'' &= y'\theta, \\ map_x(u,v) &= x'' f_x + c_x, \\ map_y(u,v) &= y'' f_y + c_y. \end{aligned} \tag{3.2}$$

With these new maps, the ray directions are corrected for distortion, and thus our model could produce undistorted renderings. Our complete method for computing undistorted rays can be seen in Algorithm 1.

To prevent the loss function from comparing a distorted ground truth with the undistorted rendering, we compared the rendering to an undistorted ground truth. Now the network was able to learn an undistorted representation of the knees. Given that the other compared models did not account for distortion, during evaluation we remapped the rendered pictures back to the same distortion as before and compared to the unaltered ground truths. This was done using OpenCV's `initInverseRectificationMap`. This function calculates the reverse maps needed to redistort the picture using the old intrinsics and radial distortion parameters. As can be seen in Figure 3.3 the radial distortion is characterized by a barrel distortion as explained in Section 2.1.2 and this corresponds to a distortion typical for arthroscopic lenses.

Algorithm 1 Calculations of ray directions in our model

```
Require: intrinsics $(f_x, f_y, c_y, c_x)$, distortion $(k1, k2, k3, k4)$
    new_intrinsics ← getOptimalNewCameraMatrix(intrinsics, distortion, $\alpha$)
    $\text{map}_x, \text{map}_y$ ← initUndistortRectifyMap(intrinsics, new_instrinsics, distortion)
    $r_d \leftarrow [(\text{map}_x - c_x)/f_x, -(\text{map}_y - c_y)/f_y, -1]$
```

3.2.2 Network architecture

The data available show quite narrow views where each view does not differ much from the next. The original implementation of NeRF-W is made to handle large static buildings using a data set of many different views. Given our data sets, we felt the need to increase the complexity of the networks to produce greater results.

As in NeRF-W, our network is still made up of three MLPs. For the base MLP, we use the same structure as in the original NeRF, with eight fully connected (FC) layers with 256 hidden units with ReLU activation functions in between. As in NeRF, we also include a skip connection in the fifth layer where we concatenate the positional encoding of the input location to the layer's activation [17]. The static MLP in our model is composed of one FC layer for the input location and two FC layers with ReLU as activation functions for the view direction input. The static density head that outputs, σ_i, is composed of three fully-connected layers where the activation functions of the first two are ReLU and a softplus for the final layer. The static RGB head uses a single FC layer with a sigmoid activation function. The transient MLP is made up of, six fully-connected layers for the transient encoding input, where each of these layers has a ReLU activation function. The transient density head, outputting $\sigma_i^{(\tau)}$, has three FC layers, where ReLU is used for the first two layers and a softplus function is used for the final layer. The transient RGB head is made up of two FC layers with a ReLU and sigmoid activation function, respectively. The transient uncertainty head, outputting β_i, is just a single FC layer combined with a softplus function. For all layers of the transient and static MLPs, we use 128 hidden units, except for the static density head which uses 256 hidden units.

3.2.3 Segmentation masks

One of the features of NeRF-W that was interesting for this application was the handling of transient objects. As stated before, the data sets contained images with medical tools in them, and to render an accurate model of a knee, these needed to be removed in the final renderings. We had access to tool segmentation masks of the data that displayed where the tools were located in the images. We could use these masks to help the uncertainty field that our model produces to determine where transient objects are located.

The first step in incorporating the segmentation masks was to correct for the distortion; this was done in the same way as was done with the training images (3.2.1). This produced a mask that matched the undistorted tools in the images. We combined the segmentation mask with the lens mask presented in Section 2.3 using the following equation

$$segmask_i \leftarrow segmask_i * (1 - \beta_w) + mask_i * \beta_w, \tag{3.3}$$

where $segmask_i$ and $mask_i$ are the segmentation mask and lens mask of image i, respectively, and β_w is a hyperparameter between 0 and 1. The transient network

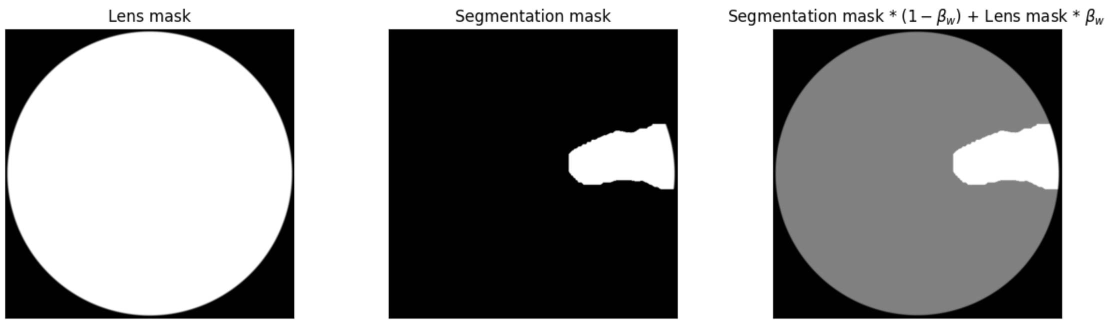

Figure 3.4: Comparisons between the lens mask, a segmentation mask, and the combined segmentation and lens mask used in masking the uncertainty fields in the transient network of our model. The value of β_w is 0.5, and the values of white pixels are 1, grey pixels 0.5, and black pixels 0.

outputs a ray-dependent uncertainty of the observed color of a ray, denoted $\beta_i(t)$ (Eq. 2.28). By masking this uncertainty, we could control how much importance each pixel in the uncertainty field was given, this masking was only done for data sets where segmentation masks were available. Since we knew from the segmentation masks where transient objects existed, we gave full importance to those pixels. Using Eq. 3.3, we also assigned some importance to the pixels inside the lens mask, that is, $\beta_w * pixel\_value$ importance. The reason for not just using the unaltered segmentation masks is that we still wanted to be able to find transient objects inside the lens mask such as organic matter that had not been segmented in the provided segmentation masks and also account for non-perfect segmentations. In Figure 3.4 an example of how a segmentation mask looks after being combined with the lens mask using Eq. 3.3 with $\beta_w = 0.5$ can be seen.

3.3 Model training

To be able to compare as accurately as possible, we decided to train our model with the same parameters as we did for NeRF-W (4.1.2). We trained for 10 epochs with a batch size of 4096, totaling between 120000 and 150000 iterations, depending on the data set. We sampled 64 points per ray from each of the coarse and fine networks for a total of 128 sampled points per ray. We used an Adam optimizer, with an initial learning rate of $\eta_0 = 5 * 10^{-4}$ and a cosine annealing scheduler. The hyperparameters λ_u and β_{min} were set to 0.01 and 0.1 respectively. The embedding dimensionalities $n^{(a)}$ and $n^{(\tau)}$ were 48 and 16. For the positional encoding function frequencies, we used 16 for the position and 4 for the view direction. For the data sets where segmentation masks were available, we used $\beta_w = 0.5$. The number of trainable parameters of our model was $38.4 * 10^3$ for the appearance embeddings, $12.8 * 10^3$ for the transient embeddings, $750 * 10^3$ for the coarse network, and $917 * 10^3$ for the fine network. The total number of trainable parameters for NeRF-W was around $1.7 * 10^6$.

4
Results

In this section, we present the experimental phase of our thesis alongside the results of our model compared to those of the other NeRF implementations.

4.1 Experiments

The experimental phase of this thesis involved an initial comparison of various NeRF models, followed by the selection of the most suitable model as the basis for our own. In the following sections, we outline which models we have compared and why we deemed it reasonable to do so.

4.1.1 NeRF

Due to the nature of the data intended for use in this thesis, we believed that using the original NeRF algorithm would produce quite poor results but we intended to use it as a baseline comparison to the other models.

Before training, each image was downsampled by a factor of 3, resulting in images of size 360x360 pixels. We sampled 64 points per ray from the coarse network and 64 points per ray from the fine network for a total of 128 network queries per ray. During training, we used a batch size of 1024 and trained for 200000 iterations, and we used the PyTorch implementation [34] where we implemented masking support and support for our chosen COLMAP camera model. For the optimizer, we used Adam with an initial learning rate of $\eta_0 = 5 * 10^{-4}$ that exponentially decayed using the following formula:

$$\eta_{n+1} = \eta_0 * \left(0.1^{\frac{n}{250*1000}}\right), \tag{4.1}$$

where n indicates the current step of the training. The number of trainable parameters was $595.8 * 10^3$ for both the coarse and fine networks, resulting in a total number of $1.19 * 10^6$ trainable parameters for NeRF.

4.1.2 NeRF in the Wild

In the data set we are presented with several objects that could be considered transient occluders, due to this we believed that NeRF-W could be a good foundation for our model. Another reason is that given the position of the knee during surgery, the lighting conditions can greatly differ and thus we believed that NeRF-W would be a reasonable choice.

For training, we used a PyTorch Lightning implementation of NeRF-W [13] where we also added masking support and support for our chosen COLMAP camera model. This implementation has some differences from the actual NeRF-W paper; it uses 8 layers of 256 units in the base MLP, while the original NeRF-W uses layers with 512 units. The static RGB head uses one layer instead of four as the original NeRF-W does. Instead of using ReLU as an activation function for the density, it uses a softplus function. The hyperparameter β_{min} (Eq.2.28) is added after the beta composition, which is different from Eqs. 2.26-2.28. Another difference with this implementation is that it adds 3 to the beta loss in Eq. 2.29 to make it empirically positive [13]. The reason for choosing this implementation was that the original authors of NeRF-W had not released their code at the time of writing this thesis.

Before training, each image was downsampled by a factor of 3, resulting in images of size 360x360 pixels. We sampled 64 points per ray from the coarse network and 64 points per ray from the fine network for a total of 128 network queries per ray. A batch size of 4096 was used and we trained for 10 epochs for each data set, which resulted in between 120000 and 150000 iterations depending on the data set. The optimizer used was Adam, with an initial learning rate of $\eta_0 = 5 * 10^{-4}$ and a cosine annealing scheduler, an optimization technique that gradually reduces the learning rate from a high initial value to a low final value, following a cosine function. This helps improve the generalization performance of the model. The L_1 regularizer multiplier λ_u was set to 0.01 and a value of 0.1 was chosen for β_{min}. The embedding dimensions were set to $n^{(a)} = 48$ and $n^{(\tau)} = 16$. The positional encoding function frequencies were chosen as 16 for position and 4 for the view direction. The number of trainable parameters of NeRF-W was $38.4 * 10^3$ for the appearance embeddings, $12.8 * 10^3$ for the transient embeddings, $601 * 10^3$ for the coarse network, and $687 * 10^3$ for the fine network. The total number of trainable parameters for NeRF-W was around $1.3 * 10^6$.

Given the metrics used to measure perceptual image similarity (2.6), a problem occurs when comparing two images containing the same underlying 3D structure with different lighting conditions but are otherwise perfectly aligned. This comparison will result in a significant “perceptual” difference [15]. This poses a challenge in evaluating NeRF-W since the image-specific appearance embeddings $\ell^{(a)}$ are optimized only for images in the training set. To evaluate an image from the test set, the corresponding embedding must be learned. To prevent evaluation on trained images, the authors of NeRF-W solve this issue by optimizing these embeddings on the test set images by only optimizing the appearance embeddings on the *left half* of the image and evaluating on the *right half* [15]. This approach allows the network to adapt to the appearance of each scene, while not optimizing held-out pixels. This works for the phototourism data set that NeRF-W was optimized for, where the images consisted of static landmarks with transient objects in both the left and right parts of the images [15]. In our data sets, and specifically in the BC and OC sets we have medical tools in either the left or right half of the images respectively.

If we were to split the images into left and right halves and optimize the appearance embeddings on one of these halves, this would result in the other half not containing any tools during evaluation, which would not be representative of the training data. Due to this, we decided to optimize the appearance embeddings on the *bottom half* of the image and evaluate on the *top half*, to guarantee that the tool would be seen in both halves. This optimization was carried out for 10 epochs, resulting in around 4000 to 5000 iterations for the test set.

4.1.3 Self-Calibrating NeRF

Since one of the main contributors to the success of the NeRF network is the camera parameters, both intrinsic and extrinsic, we chose to test SCNeRF. It refines the camera parameters during its training, which might result in better reconstructions.

Before training, the images were downsampled by factor 2 to an image size of 540x540 pixels. For training, we used the implementation of SCNeRF on NeRF++ that the authors used for fisheye lenses [10] and that represents the arthroscope used in our data. We also integrated a masking component on the loss function in the same way as we did for NeRF and NeRF-W. We sampled 64 points per ray from the coarse network and 128 from the fine network. We trained with a batch size of 8192 rays. The model was trained for 300000 steps with a learning rate of $\eta_0 = 5 * 10^{-4}$ and a decay factor of 0.1, decaying every 750 steps. The optimizer used was an Adam optimizer. The intrinsics learning started after 100000 steps, the radial distortion after 220000, and the ray learning after 160000 steps. The number of trainable parameters for SCNeRF was $22 * 10^3$ for the camera model and $1.2 * 10^6$ for both the coarse and fine network for a total of $2.4 * 10^6$ trainable parameters.

4.1.4 Comparisons

Comparing these different models on the same data sets allowed us to determine which model achieved the best results, and thus was the best place to start for our model development. Given the test sets, we evaluated the different models to determine which performed the best. The metrics we compared with are described in Section 2.6. In Table 4.1 we see the results for all data sets and models, the best results for each data set and metric are highlighted.

4.2 Quantitative and qualitative results

In Table 4.1 we see quantitative results on all data sets tested for the compared models; NeRF, NeRF-W, SCNeRF, and our model. We see that our model outperforms the other models in all data sets except the Open source set, where NeRF-W achieves better PSNR and LPIPS scores. We present an average improvement over all data sets compared to NeRF-W with 5.7% in PSNR, 0.69% in SSIM, and 23.5% in LPIPS. In Figure 4.1 we present qualitative results in the form of rendered images of the Open source data set, from the compared models. As can be seen, both NeRF

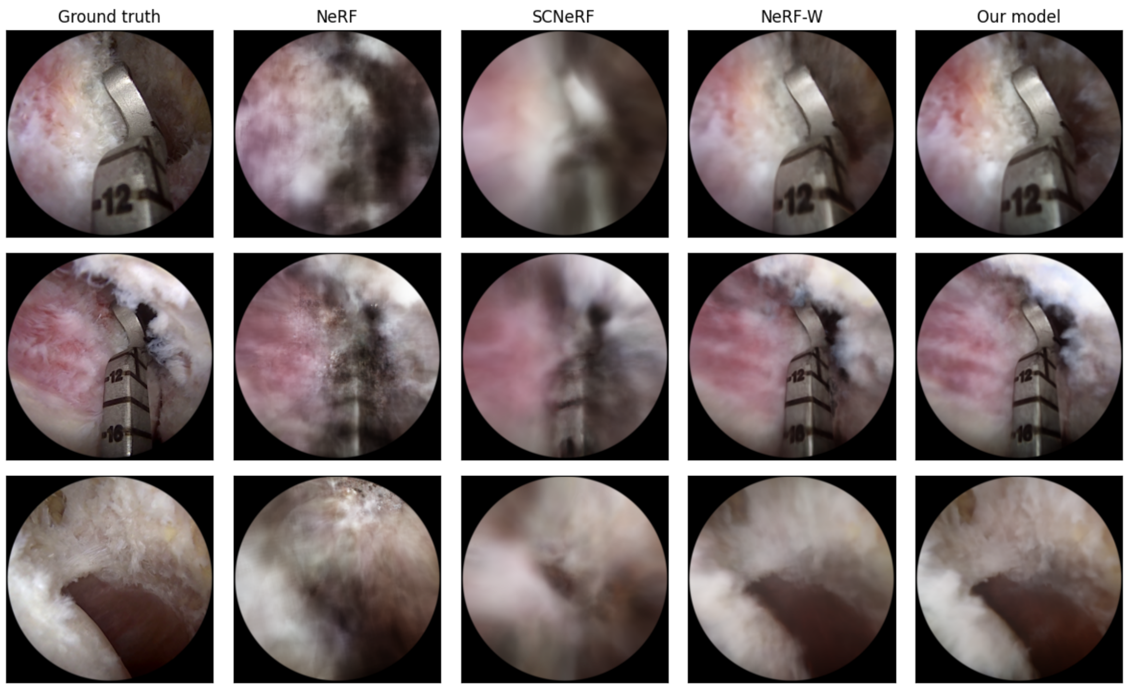

Figure 4.1: Qualitative results from experiments on the Open source data set. We show rendered images of the models NeRF, NeRF-W, SCNeRF, and our model compared to a ground truth image.

and SCNeRF produce quite unfocused renderings. They cannot represent the knee well, however looking at the scores for these models on the Open source data the PSNR for NeRF is better but SCNeRF captures the features more correctly than NeRF as is captured in the SSIM and LPIPS scores.

We see that neither NeRF-W nor our model manages to remove the medical tools present in the images in rows 1 and 2. In the rendering of our model in row 2, we can see that the lighting in the upper right corner does not correspond to the ground truth image however it does manage to capture the stripes to the left of the tool. The same can be seen in the rendering in row 1 where it more correctly renders the left-hand side compared to the upper right. Our model is also able to render the varying organic matter in the background of row 3 better than NeRF-W.

The shortcomings of our model can be seen clearer in Figure 4.2. In the first image, our model overestimates the amount of organic matter in the top right corner and cannot render the finer details of it and instead produces a smeared estimate. In both the first and second images we see that the model struggles with rendering the correct lighting and the results seem a bit blurred, this is especially visible in the second image where the yellowness of the ground truth image is not rendered at all.

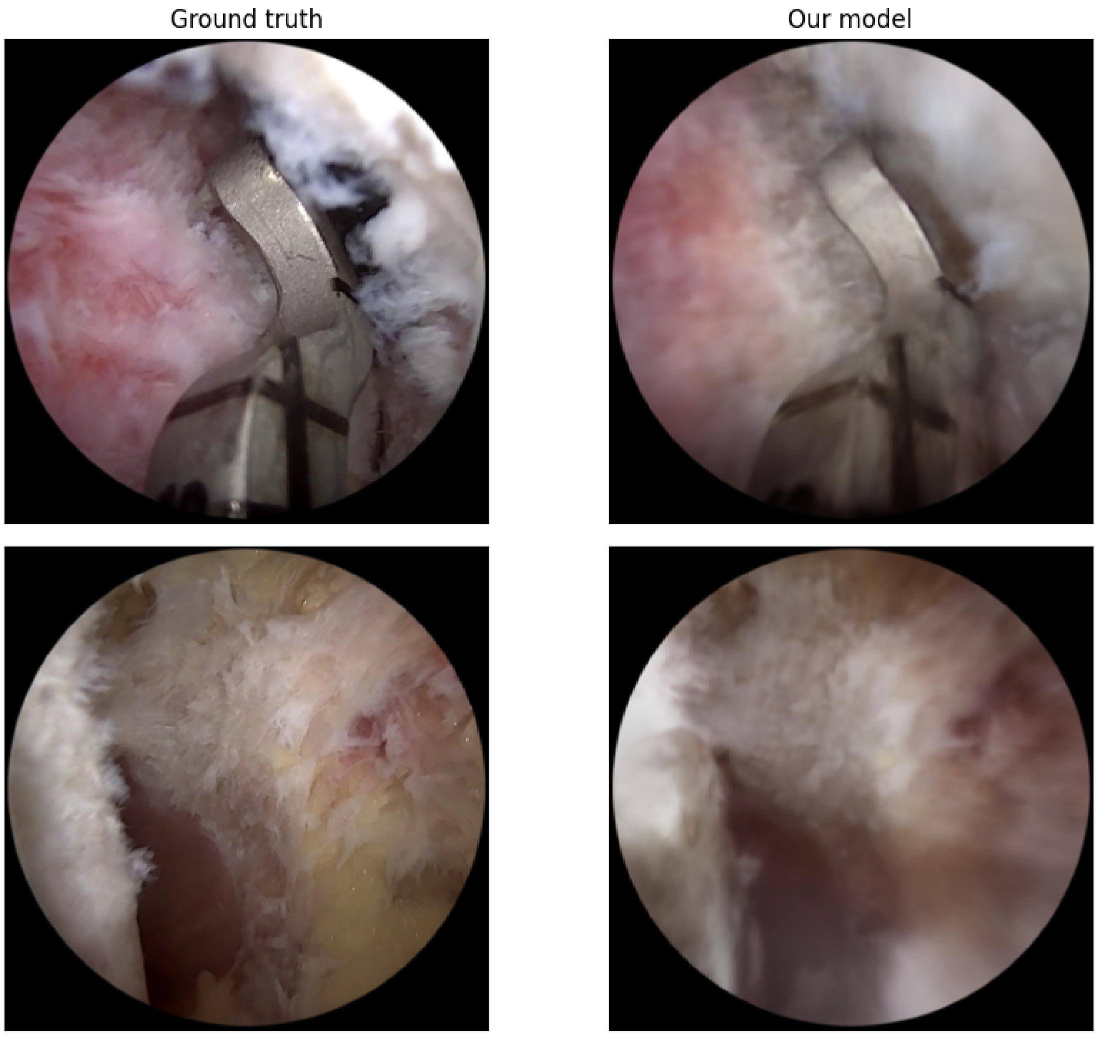

Figure 4.2: Results of our model where it fails in producing high-quality renderings.

	IC data set			OC data set			BC data set			Open source data set		
Method	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
NeRF	24.54	0.901	0.0084	21.16	0.808	0.0118	22.8	0.848	0.0065	19.9	0.804	0.0147
NeRF-W	32.34	0.934	0.0021	22.32	0.847	0.0093	23.69	0.873	0.0047	**23.35**	0.876	**0.0092**
SCNeRF	26.95	0.895	0.0037	18.57	0.819	0.0108	20.06	0.843	0.0054	17.94	0.853	0.0115
Ours	**35.17**	**0.937**	**0.0012**	**24.05**	**0.855**	**0.0071**	**24.71**	**0.878**	**0.0042**	22.51	**0.878**	0.0098

Table 4.1: Quantitative results on the IC, OC, BC, and Open source data sets for NeRF, NeRF-W, SCNeRF, and our model. The best scores for each data set are **highlighted**. Our model outperforms all other tested models in all data sets except the Open source data set, where NeRF-W performs the best.

4.3 Depth maps

In Figure 4.3 we present depth maps for some images from the Open source data set produced by the models NeRF, NeRF-W, SCNeRF, and our model. We do not have access to any ground truth depth maps, and can thus only compare the results quantitatively to an RGB image. The pixel colors of the depth maps have different meanings; a blue pixel corresponds to a deeper point, while a red one corresponds to a closer one. We see that our model produces somewhat promising depth maps. The model manages to estimate that the tool is close to the camera in row 1 and that there is some sort of deeper point in the top right. In row 2 we see similar estimations, however, the model struggles with the depth of the knee in the left-hand part of the image. In row 3 we see that the model has correctly managed to assess the closeness of the lower left-hand part, however, it fails to assess the depth of the top left and bottom right which is true for all models.

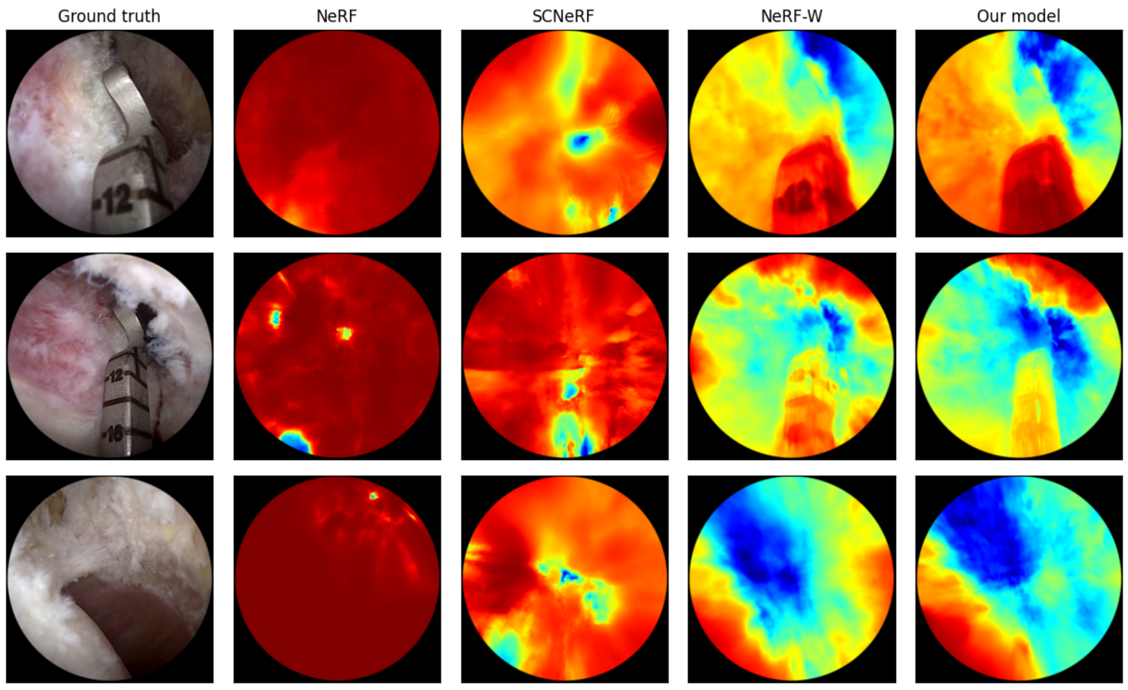

Figure 4.3: Qualitative comparison of depth maps for images in the Open source data sets from models NeRF, NeRF-W, SCNeRF, and our model. No ground-truth depth maps are available so the produced depth maps are compared to a ground-truth RGB image. Blue pixels correspond to a deeper point while red ones a closer point in the images.

4.4 Novel views

In Figure 4.4 we show some novel views produced by our model of the Open source data set.

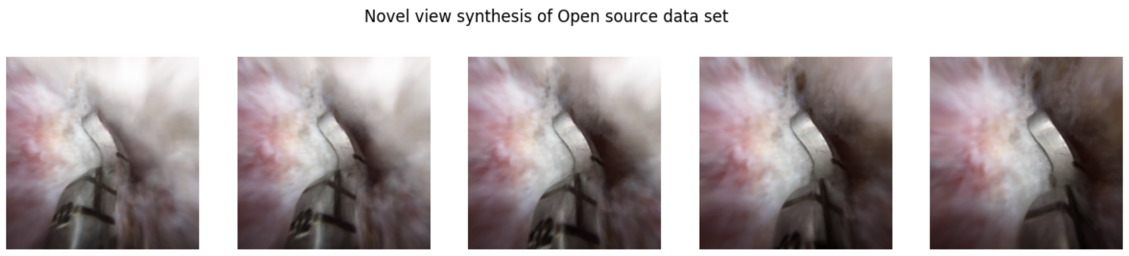

Figure 4.4: Novel views of the Open source data set produced by our model.

5 Discussion

The following section discusses the results of the thesis and the methodologies used.

5.1 Results

We see in Table 4.1 that our model produces high scores compared to other models on the IC data set, which is not surprising. This data set is almost as perfect as possible for this application, and thus the model can accurately render images. We were quite surprised to see that the scores on the BC set were better than those on the OC set for all models. We believed that since the camera conditions of the OC set were superior to those of the BC set, the resulting scores would be better. One possible reason for this is that some of the frames in the OC set show a lot of bubbles, created by the instrument burning away organic matter; these can be hard to model because of the reflection in the bubbles, which differs depending on what viewing angle the bubbles are viewed from. Thus, during training a bubble might look different than the same bubble seen during testing, due to the angle of the camera. These bubbles can also be seen as transient because no bubble stays in the same place for more than a couple of frames. This would explain why NeRF and SCNeRF struggle with this data set. In NeRF-W and our model, these would not be rendered as these models attempt to remove transient objects, and thus would lead to worse comparison scores; this is discussed further in Section 5.6.

The BC set also contains a medical tool that is regarded as transient, but it is much larger than the tool present in the OC set and thus occupies the majority of the view for the duration of the video. Due to this, some parts of this tool could be seen as rigid, which allowed the models to render it more accurately. This is especially visible in the OC and BC set renderings of NeRF-W and our model, which are designed to remove transient objects. In the OC set renders, they perform quite well in removing the tools, which is another reason the scores decrease, as explained in Section 5.6, but in the BC set renders they have a harder time in removing the tool, the Open source renderings in Figure 4.1 show similar results. Therefore, the comparisons with the ground truth images containing this tool will be better.

In Table 5.1 we show PSNR scores of the rendered test images that contained tools and thus were removed by NeRF-W and our model for the OC data set. As explained in Section 2.6 we can pass masks to the PSNR calculation to only compute scores for certain pixels, in Table 5.1 we first show scores for both NeRF-W and our

Model	PSNR ↑ (Lens projection)	PSNR ↑ (Lens projection w/o tool)
NeRF-W	19.05	21.47
Our model	**19.35**	**22.58**

Table 5.1: Comparison of PSNR for rendered images with removed medical tools for the OC data set. The second column corresponds to PSNR scores where the lens mask has been removed, third column corresponds to PSNR scores where both the lens mask and the tool mask have been removed. The best scores in each column are highlighted.

model computed on only the lens projection, with the lens mask removed. We also show the scores computed when we pass both the lens mask and tool segmentation mask, thus ignoring the pixels where the tools existed. As we can see, we achieve higher scores for both models when we pass the segmentation masks as well. Since both models manage to remove the tool to varying degrees, these areas would not correspond well to the ground truth images and thus lead to a decrease in score, which is why we achieve worse scores for both models when only removing the lens mask. This is an indication that the models manage to remove the tools, however, we also see that our model has greater PSNR scores when the segmentation masks are passed, this is due to NeRF-W not quite managing to remove the tools fully and producing artifacts in the renderings outside the segmentation masks and thus leading to a decrease in score. Thus, we can argue that our model performs better in removing the transient tools even though we are not able to show any qualitative results of this due to the ethical and privacy reasons discussed in Sections 3.1 and 5.3.2. We can also argue that our model performs better on the OC data set than the BC data set since we show that it succeeds in removing transient objects and rendering a more accurate depiction of the knee, which was our goal.

In the renderings of the Open source data set in Figure 4.1 we see that both NeRF-W and our model struggle greatly with removing the transient object, we believe that this is the same issue as in the BC set. These two sets are quite similar in the way that a large part of the lens view is occluded by a tool that does not differ much between frames, and thus the model has a hard time distinguishing it as transient. The tool in the Open source set does not move very much between frames and hides what is behind it for a majority of the video, thus making it very hard for the model to estimate the color of those parts since they are barely seen. Even though we pass segmentation masks to "help" the model in finding where the transient object lies, it does not help if the model does not consider that part of the render to be transient.

For the renderings of the OC data set, which we are prohibited from displaying, as discussed in Sections 3.1 and 5.3.2, both models can remove the transient objects. However, it seems that our model does a better job. The renderings of NeRF-W include some faint artifacts of the transient tools while these artifacts are even fainter or non-existent in the images produced by our model. This aligns with what we discussed above, the transient tool in the OC data set is much smaller and moves around a lot more, allowing the model to perceive it as transient given that the views differ from image to image. The model observes the knee obscured by the tool more and is thus better able to estimate the colors in that region.

In the renderings produced by NeRF-W and our model, we see that the models struggle with the finer details, the “fuzzy” part of the knee. The colors are generally correct, but the finer details are blurred and the models do not achieve high fidelity on these parts. The organic matter is characterized by quickly varying colors with no distinct transition and, therefore, our model struggles to represent these fine details. However, it does achieve great fidelity in rendering the tool, which both shows high detail and correct reflection of lighting. Compared to organic matter, the tools are of a consistent color with sharp transitions between colors and are therefore easier for the model to learn.

By examining the renderings of our model compared to those of NeRF-W, we see that our model seems to produce sharper images and with somewhat better color rendition than what NeRF-W does. This is especially visible in the images in rows 1 and 3 in Figure 4.1, where the images produced by NeRF-W seem to be a bit more blurry than what our model produces. However, NeRF-W still achieves better scores on this data set than what our model does, we believe that this is due to our model producing incorrect artifacts of organic matter, as can be seen in Figure A.1. These artifacts would contribute to the overall score more than what the sharpness of our images does, thus allowing NeRF-W to outperform our model on this data set. We believe that these artifacts are the result of our segmentation masking, and more specifically the hyperparameter β_w being too small and giving too little importance to the parts outside the segmentation mask. The model could interpret these parts as transient, but because β_w is too low, we decrease the importance of these pixels too much and thus trick the model into thinking these parts are not transient or at least not as transient as the model first may believe, and will thus render them. It is not hard to understand why the model would believe these parts are transient since the artifacts mostly seem to be fuzzy organic matter that moves around between frames and given different viewpoints can look very different.

Another reason NeRF-W outperforms our model on the Open source data set could be that it seems to render the lighting conditions more correctly. This can be seen by comparing the images in row 2 of Figure 4.1. We see that NeRF-W has a slightly more yellow tone of the organic matter in the top right, which seems to correspond a bit better to the ground truth, our model seems to render these parts in a whiter color. This might indicate that our model needs further optimization of its appearance embeddings.

5.2 Camera parameters

One fundamental part of most NeRF models is the extrinsics and intrinsics of the cameras used. Using estimated parameters from COLMAP does induce an error since we cannot make sure that the calculated intrinsics are the real parameters or how far from the real parameters they are. Since we do not have access to the cameras used in surgeries and therefore cannot estimate the true parameters, our only option was to use the COLMAP parameter estimation, a common choice for this task [17, 15, 10]. If possible we would have used a camera calibration technique on the cameras used to calculate true parameters, such as using checkerboard calibration [28]. Although our model renders undistorted images, there is no way of knowing that the rendered image is an accurate rendition or how far we are from the truth since we might have incorrect distortion parameters. We can only qualitatively assess the renderings and see if they seem reasonable. If we had access to the true parameters, a more informed assessment would have been made.

Another aspect contributing to the challenge with camera parameters is the diverse complexity levels exhibited by the various camera models employed by COLMAP to estimate intrinsic parameters. When more parameters were introduced, the estimations did not quite align with what was expected. The arthroscope used in surgeries has radial distortion with its radial lens but as stated in [25] there is no tangential distortion. We tested a couple of different models with COLMAP and through testing and discussions, we arrived at the OpenCV camera model with 4 distortion parameters. This keeps the radial distortion that is introduced with the radial lens but forgoes the higher-order radial and tangential distortion keeping it more stable.

For data sets where the estimated camera intrinsic parameters were qualitatively worse we chose to use the intrinsics from another set, more specifically the distortion parameters. We could not guarantee that the cameras used in these data sets were the same, but we decided that since they are all arthroscopes used in the same context the difference should be minimal. This sharing of parameters was done for the BC and Open source data sets where we instead of using the estimated distortion parameters of these sets used the parameters estimated for the IC set. The inability of COLMAP to estimate these parameters might be related to the tools being present in most of the pictures and taking up a large part of the screen. The tools themselves are easily used as features and therefore might introduce faulty features in the matching algorithm, as the tools are used in many parts of the knee and, therefore, the location of these features changes from frame to frame. This might cause COLMAP to match images that are not correlated in other ways other than the tool. COLMAP then tries to compensate for the distortion effects to triangulate these features between frames and therefore produces bad distortion parameters.

5.2.1 Self-Calibrating NeRF

When doing experiments with SCNeRF the optimization of the camera parameters did not live up to our expectations. There was a change in the estimated parameters that it used, but in the end, the results themselves did not point to it improving the model. This might be due to the amount of training that we performed for SCNeRF. To keep the comparisons fair, we used a similar amount of training time for all models, which meant that we shortened the training time and added the learnable parameters earlier than in the original paper. In the original SCNeRF implementation, the authors sampled quite a bit more points per ray than what we did; they sampled 128 points from the coarse and 256 points from the fine network. The results in the original paper for fisheye lenses were not that impressive and could not handle the distortion that well. This combined with the computational concerns described below was the main reason we did not incorporate the SCNeRF plugin into our model.

5.3 Ethical and environmental concerns

The impact of this study is hard to quantify, but it is necessary to think of the ethical and environmental impacts of the use of patient data and the growing problem of global power use.

5.3.1 Reliability of renderings

An important ethical factor to consider is how reliable the renderings produced by our model are. Even though the aim of this thesis was not to create a tool to aid in ACL surgeries, it is a possibility that this model can be used in further work toward this goal, and if our model were to be used in the future, we must be able to ensure that the renderings represent an accurate view of the knee. If not, this could be disastrous as a faulty representation could cause medical professionals to make wrong choices, which could cause harm to patients. To ensure reliability the renderings would need to be examined by trained professionals to determine accuracy. Another way to ensure reliability is to train the model using diverse data sets containing variations of anatomical features and tools and assess how well the model performs. Depending on how well the model performs on data containing these variations, changes can be made to improve the model. This is important since such variations can occur during surgery and if the model is not equipped to handle them, will not produce reliable results.

We do not aim for our model to be used as a replacement to any existing technique but instead provide the professionals performing the surgeries with more information to base their decisions on. Thus, the decision always lies with the professional performing the surgery and not the tools they use. Although, we must try to ensure that the tool used is as accurate as possible.

5.3.2 Patient data handling

Given that our data sets contain patient data, we were not allowed to show any of the ground truth images or even the renderings of these images in the thesis. This is due to the ethical application that had been granted to the company that supplied the data, Tenfifty AB, which prohibits this. This posed quite an issue since one of the most interesting parts of a thesis on novel view synthesis is to display the rendered images. Although we present quantitative scores on how the models perform on each dataset, it is hard for a reader to understand how well these scores relate to the models' performance without being able to qualitatively compare between ground truth images and rendered ones.

To overcome this issue, we were also given access to an open source data set of ACL surgery, which did not fall under the same restrictions as the other available data sets. This data set differed a bit from the others, it contained a tool that the other data sets did not contain. This presented a challenge to the segmentation model used to provide the segmentation masks since this model had been trained on the original data sets, as discussed in Section 5.5.

Given that our model requires data from ACL surgeries to train on, problems occur when the data used are not publicly available. Any renderings or models created using confidential data will not be able to be viewed by anyone outside a confidential agreement. Thus, we believe that a larger population would benefit if all the data we had available were publicly available, such as the Open source data set is. This would allow us to present the results more readily and would enable us to more easily apply our results in other contexts. An example would be to create 3D models of knees based on the data, which could be used as training material for ACL surgeries.

5.3.3 Computational concerns

Neural radiance fields are very computationally heavy, and our model is no exception, due to this we decided to train for a maximum of 10 epochs even though the authors of NeRF-W train for 20 in their training. They also sample a lot more samples per ray than we did, 512 points from the coarse and fine network respectively, compared to the 64 points respectively for our training. Given this, we believe we can achieve greater results if trained further. We made a conscious choice to limit our training due to the amount of NeRF models to compare, and the amount of data sets.

The escalating computational demands of computer science and AI have become a bigger concern. A more general example is that the power consumption of all computers increased from 1-2% of the world supply in 2018 to 4-6% in 2020. This consumption is estimated to reach 8-21% in 2030 if the computational needs continue according to Deep Jariwala [14]. This poses some serious questions when using AI for applications with large energy consumption, such as our model during its long training time.

Although the computational needs for individual runs may not be unreasonably demanding, the cumulative impacts are something to take into consideration. We were conscious of these implications while performing our experiments. However, our primary focus was not on optimizing our model in speed and efficiency to reduce the needed energy, we do acknowledge the importance of this issue and it should be kept in mind during development to create responsible and sustainable AI solutions.

5.4 Test set filtration

During the training of the different models, we noticed that the calculations to create test sets by filtering images within a certain distance were performed on incorrect pose values. This meant that some of the resulting test set images were, in fact, within the minimum distance from the training images. The calculation we used took into account only the translation vectors $\mathbf{t}$ to check the distance between the test images and the rest of the set. Accurate distance transformation is done by also taking rotation into account with $P = [\mathbf{R} \mid \mathbf{t}]$. The camera path of the BC set is presented in Figure 5.1 with the left picture being filtered with faulty poses and the right one being done on the complete pose, including rotation. The minimum distance for both graphs was the one used in our incorrect filtering, which can be seen in Table 3.2. As we can see, the number of filtered images changes quite drastically between these two, as the minimum distance used filters out almost the entire set in the right graph. Thus, when filtering correctly, we would have to decrease the minimum distance to get a similar split of training, test, and filtered images compared to the split used.

The mean distance of the camera poses between the test and training set images for the test set used and the one with correct calculations is presented in Table 5.2, the minimum distance for the accurate calculations is 0.01 for all data sets instead of the ones used in Table 3.2. The unit is undefined, as is explained in Section 3.1, here we also show how the distance corresponds to the max distance between points for each data set. As we can see it is mostly the IC data set that is impacted by these calculations.

Data set	Mean distance (Used)	Mean distance (Accurate)
IC	0.247 (28.7% of max distance)	0.317 (36.9% of max distance)
OC	0.328 (28.3% of max distance)	0.321 (27.7% of max distance)
BC	0.49 (43.7% of max distance)	0.48 (42.9% of max distance)
Open Source	0.48 (24.5% of max distance)	0.43 (21.9% of max distance)

Table 5.2: Mean distances between train and test set

As rerunning all the training would be computationally and time-expensive, the decision was made to retain the faulty data sets, considering their low impact in the end.

Given that the variation of view direction of the cameras in the data sets was small, no consideration was taken to the camera direction when filtering. If this had not been the case, we would also need to check the direction of the cameras with a minimum angle within the minimum distance to a test camera to determine whether we should filter it or not. Two cameras with the same position but viewing entirely different points should both be able to exist in the data set.

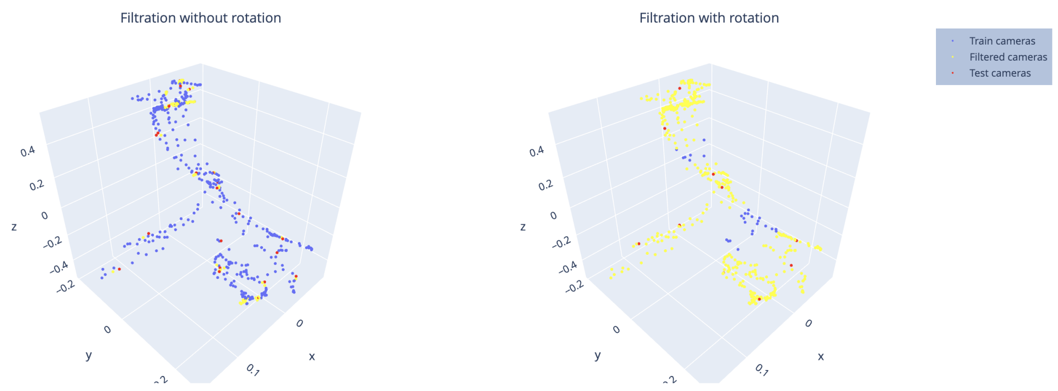

Figure 5.1: Comparison of the BC data set when filtering without rotation vector and with rotation vector. The minimum distance used is 0.1.

5.5 Segmentation masks

The segmentation masks used are created using a trained segmentation model that was trained on data similar to the IC, OC, and BC data sets, that is, containing the same kind of medical tools. The model is not perfect, and thus the segmentation masks produced were not completely accurate in masking the entire tool. A worst-case scenario can be seen in Figure 5.2, this image is taken from the Open source dataset where tools that the segmentation model had not been trained on were present. As we can see, barely half the tool is segmented. To solve the issue of the segmentation model not being able to produce accurate masks for the Open source data set, we manually annotated all tools in this data set. This allowed us to achieve high-accuracy segmentation masks.

Even though we now had segmentation masks of high quality for the Open source set, the quality of the other sets was not of equal quality and thus we felt it necessary to allow the uncertainty fields to also be trained outside the segmentation masks. As explained earlier, this also allowed the model to find transient objects outside the segmentation masks, such as bubbles or organic matter floating around in the knee and clouding the lens view.

As explained in Section 5.1 we believe that the value of the hyperparameter β_w is one of the contributing factors that NeRF-W outperforms our model on the Open source data set. We used the same value of 0.5 for all data sets when this value probably should have been specific to each data set. Some sort of cross-validation hyperparameter tuning should have been performed for each data set, in determining the best value for β_w. This was decided against due to computational and time concerns.

We believe the value of β_w should be higher for data sets such as the Open source set where the transient object is not very transient. In other words, where the object occupies a lot of screen space and moves very little between frames. This is due

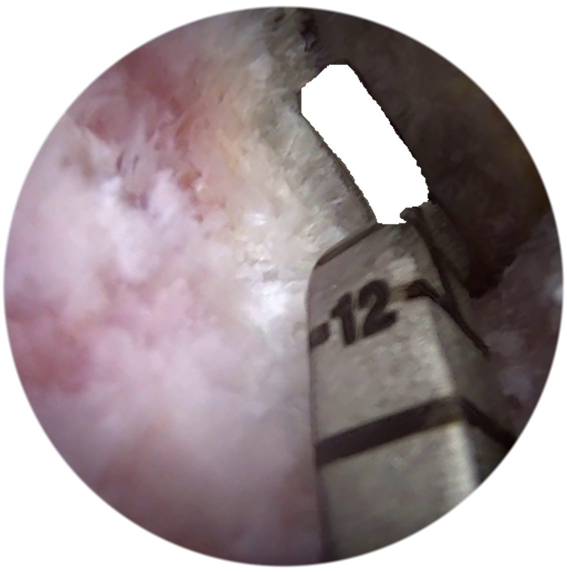

Figure 5.2: A tool segmentation mask layered on the ground truth picture. This segmentation mask is very poor as it barely covers half the instrument. Picture taken from the Open source data set.

to the model encountering increased difficulty in explaining this object as transient, and thus more importance should be given to the pixels outside of the segmentation. As discussed earlier, the BC data set contains a tool of similar size to the one in the Open source set. That tool occupies a large part of the screen, but it is moved around quite a lot, and thus the model has an easier way of estimating the color behind the tool and can model it as transient, and the value β_w does not require to be that high. This could explain why we achieve greater scores than NeRF-W on this data set with $\beta_w = 0.5$.

When using the segmentation masks to help the transient network one should examine the tools in the data. For the OC data set our implementation is working as intended by helping the transient network to focus on the tool. As discussed, the segmentation masks help our model in removing tools without producing any artifacts in the renderings. On the other hand, if the tool is too static, as discussed for the Open source data set, we could have used the mask to force the transient network to consider those parts transient, instead of just helping it. The problem is that the model does not know what is behind the tool and therefore we would need to find a solution to that, either taking the coloring from close points or making it black to represent a tool there.

5.6 Comparison metrics

The metrics used to determine how well a NeRF model performs all measure perceptual similarity between a ground truth image and a rendered image. These metrics only indicate how similar the two images are, and do in no way account for objects or other occluders that have been removed in the rendered image. A problem with this approach given the data sets used to evaluate our model is that some of the images contain occluders, such as medical tools and other objects that our model classifies as transient. As NeRF-W, upon which our model is based, is designed to generate images without transient occluders, the perceptual similarity between the modeled image and a ground truth image containing such occluders will be low. The authors of NeRF-W hand-select a test set without occluders during evaluation, however, given that the whole BC set and Open source sets contain medical instruments, we decided to not filter any images based on occluders, to still be able to test on these sets. The IC set does not contain any occluders, so no choice had to be made there. To be consistent with the other data sets, we decided to not remove the few images with occluders in the OC test set.

Another issue with the comparison metrics used is that the ground truth images contain a black border around the actual lens projection, as can be seen in Figure 3.1. The rendered images do not contain this border. Thus to perform a fair comparison of perceptual image similarity, we decided to apply a mask to the rendered images before evaluating. Without this application of the mask, the scores would be significantly lower, since a large part of the rendered image would not correspond to the black border of the ground truth images. Even though the rendered image would be a completely accurate rendering of the knee, the scores would have been low, since the similarities to the ground-truth images would be small. During the creation of novel views, we, of course, use the rendered images without any alteration to be able to generate truthful views of knees.

There is however an issue with this mask application as well, since it effectively boosts the similarity scores. Giving us “free” points when comparing. Large parts of the rendered image and ground truth are now exactly alike, i.e. the black border surrounding the mask. Since PSNR is a pixel-by-pixel comparison, we can pass the mask to the calculations and thus only compute similarity for the pixels that are inside of the mask. However, given how SSIM and LPIPS are designed, this is not possible, so we decided to present the scores where we compare the rendered images with mask re-application and no further changes, for all metrics including PSNR in all results, except for Table 5.3 where the unmasked PSNR scores for the different data sets and models can be seen.

This issue is most noticeable in the results produced by NeRF on the different data sets, given the scores in Table 4.1 one would assume that NeRF performs only slightly worse than NeRF-W on the BC data set. However when comparing these models qualitatively one would see that this is not the case, this would be easy to see when looking at the renders but as explained in Section 5.3 we unfortunately can not show these renderings. NeRF-W produces renderings that are a lot better than those of NeRF; however, due to the application of the mask, the scores are boosted. One reason we believe that the scores of NeRF-W are not boosted as much is due to the problems explained above, with NeRF-W not rendering the transient objects, thus leading to a lower comparison score.

	IC data set	OC data set	BC data set	Open source data set
Method	PSNR ↑	PSNR ↑	PSNR ↑	PSNR ↑
NeRF	23.46	20.07	21.64	18.82
NeRF-W	31.26	21.23	22.53	**22.26**
SCNeRF	25.85	17.46	18.88	16.84
Ours	**34.09**	**22.97**	**23.55**	21.42

Table 5.3: Unmasked PSNR scores on the different data sets for NeRF, NeRF-W, SCNeRF, and our model. Only pixels within the lens mask have been evaluated.

A problem with using PSNR as a comparison metric is that the values can be misleading. By examining only the results in Table 4.1 one would see that the PSNR value of our model and NeRF differ by quite a small margin for the Open source data set. Thus, one could think that they perform somewhat similarly, however, when examining the renders in Figure 4.1 one sees that our model performs a lot better. PSNR is very sensitive to small changes in pixel values; changes that would otherwise be indistinguishable from the human eye can lead to a significant drop in PSNR score. Since PSNR is based on MSE, the score can significantly increase given that the difference between two pixel colors is small. This is the case if the overall color of an area is correct but the image is blurry, for example. This is an example that PSNR does not correlate well with human perception of image quality. Another reason is that all errors are assumed to be of equal importance, disregarding spatial location. Due to this, both SSIM and LPIPS are better tools for determining perceptual differences between images.

5.7 Size of data set

The amount of images needed to train a NeRF model varies greatly, depending on how much of the scene is captured in each image. These images should optimally show the scene from several different angles so that the model is exposed to several viewpoints of the intended scene. Due to our data sets and application, we decided to use a large set of images during training since the videos available show a very narrow scene with minimal difference from frame to frame. Therefore, by using a smaller set of images we were afraid that the models would not see enough of the knees to render truthful images. We believe that this is the main contributor to our model taking so long to train, and it would be interesting to compare the results using a smaller data set. This was something we discussed doing during the planning of the thesis but had to forego due to time reasons.

Another reason for using more images was to estimate better camera parameters using COLMAP, the complexity of the scene did not pair well with COLMAP's parameter estimation, the parameters seemed to become better the more images we supplied to COLMAP.

5.8 View-direction and Network architecture

One of the model extensions that we studied was to remove the view direction from the network inputs. The basis for this was the narrow viewing angle in the data sets given the very limited space inside the knee. Doing this would also assume that a point has the same color independently of the viewing direction, an assumption we were fine with making since we were more interested in correct geometry than color rendition.

Without the view direction input, we noticed a significant increase in the values of the $L1$ regularization term, $\frac{\lambda_u}{N}\sum_{i=1}^{N}\sigma_i^{(\tau)}$ (Eq.2.29), also called the sigma loss, compared to the original implementation of NeRF-W. Since we removed an input to the static part of the model, we believed that the model incorrectly used the transient part of the network to compensate for this removal. To make up for the lack of view direction input, we decided to add an extra linear layer along with a ReLU function to the part of the static network that earlier handled the view direction as input. We also added additional layers to both the part of the transient network that outputs the transient density $\sigma_i^{(\tau)}$ and the part in the static network that outputs the static density σ_i. These network changes allowed our model to better handle static phenomena and now produce a sigma loss lower than what the original NeRF-W implementation did.

Despite these network changes, our model was still producing a higher loss from the fine model (Eq. 2.29), compared to the standard NeRF-W implementation. We observed that the beta loss, $\frac{\log\beta_i(\mathbf{r})^2}{2}$, was converging to the same value in both implementations after just a few thousand iterations. Thus, we hypothesized that the loss between the rendered image and the ground truth image, $||\mathbf{C}_i(\mathbf{r})-\hat{\mathbf{C}}(\mathbf{r})||_2^2$, must be the main reason for producing a higher total loss since the sigma loss was now lower for our model. This could also be explained by losing the viewing direction as input; the transient part of the network is only active for the fine model. When supplied to the network, the viewing direction is encoded by the positional encoding function in Eq. 2.18, which, as stated before, is used to more accurately represent high-frequency geometries and textures. This would explain why our model would produce images with less detail. To solve this issue, we added extra layers to the part of the transient network that outputs the RGB color. In the end, however, we decided to not pursue this approach further but instead reintroduced the viewing direction to the network but decided to keep the network changes, which now contributed to making our model perform better than the other compared ones.

5.9 Ablation studies

We confirm our design choices by conducting an ablation study; in Table 5.4 we present the results of the OC data set. As our baseline model, we have NeRF-W, since it is the basis of our model, which we see in row 1. In row 2 we show the results of a model where we only add the network changes explained in Section 3.2.2 to the NeRF-W base (NC). The next ablation we tested was to add support for segmentation masks (SM); these results are shown in row 3. Finally, we also added the undistortion of rays (UR), resulting in our full model, shown in row 4. Due to computational and time reasons, we evaluated each ablation after 5 epochs. The hyperparameters used for all ablations are described in Section 3.3.

Ablation	PSNR ↑	SSIM ↑	LPIPS ↓
1. NeRF-W	22.1	0.842	0.0098
2. NeRF-W + NC	22.69	0.836	0.009
3. NeRF-W + NC + SM	23.04	0.837	**0.0077**
4. Full model (NeRF-W + NC + SM + UR)	**23.22**	**0.851**	0.0083

Table 5.4: An ablation study of our model. Ablations are evaluated on the OC data set. The best scores of each metric are **highlighted**

We see in row 2 that we achieved an increase in PSNR compared to NeRF-W when applying the network changes, our model also achieves greater LPIPS, however, NeRF-W does achieve a better SSIM score. In row 3 we see an improvement of all metrics when supplying the segmentation masks to the network. In row 4 the scores of our full model are shown, and here we show improvement in PSNR and SSIM, however, we achieve somewhat worse LPIPS scores compared to the model in row 3, however, we still beat the LPIPS scores of NeRF-W. With this ablation study, we validate our contributions and show that our model produces greater scores than that of the baseline.

6 Conclusion

This thesis aimed to be able to render novel views of the interior of a knee using a NeRF model and also to compare this model with other NeRF implementations. The model we developed achieves a higher score than the compared NeRF models. We also present that we can remove some medical tools from the renderings, a necessary step in creating accurate renderings of the knees. Using our ray undistortion, we can accurately render the knee, compared to the compared models, which render a distorted image. However, the accuracy of undistortion largely depends on the parameters estimated by COLMAP as discussed in Section 5.2. With the help of segmentation masks, we can help the transient part of our model in producing images without tools.

We have shown that our model can produce realistic renderings of the interior of knees, these renderings can be used in further context to help create training material for medical professionals to help reduce the relapse rates of ACL surgeries.

We have shown that it is possible to use a NeRF model to synthesize novel views of the interior of a knee, and we have also shown that our model achieves better scores compared to other NeRF implementations. However, a disadvantage our model has is its long training and rendering time. This disadvantage must be addressed if deciding to proceed with creating a tool that aids surgeons during ACL reconstruction surgery. Another disadvantage is its requirement for segmentation masks, which need to be obtained before training.

6.1 Future work

One big issue with neural radiance fields is that they are slow, the training of our model for a single scene took around 2 days, and rendering a single image took around 30-45 seconds. If the goal is to be able to accurately render a knee's interior to aid medical professionals on where to place grafts, these models need to be faster. There are NeRF models specifically designed for speed and efficiency, such as InstantNGP [19] or Adaptive Shells [32], which could act as a great starting point in further development. This would be an interesting avenue to explore, given more time.

In [32] they construct a varying mesh from which to sample, where they can efficiently handle fuzzy geometry via varying sampling. Given that some parts of the knees contain fuzzy geometry, such as organic matter, we believe that Adaptive Shells could handle these areas well compared to our model which seems to render these parts as blurry.

Another approach that would be interesting to analyze is if it is possible to reduce training time by making the NeRF models more generalizable. Currently, the NeRF models we have compared are all scene-specific, meaning that each scene needs to be trained completely from scratch on a dense set of images. If we were able to train a general NeRF model with the specific function of rendering images of a knee's interior, we would perhaps then just be able to fine-tune that model given a new data set. Given that most knees are quite similar in structure, we believe this would be an interesting approach to explore given more time. There exist some NeRF models that do just this, such as MVSNeRF [3], which could serve as a base if one decides to pursue this approach.

As discussed in Section 5.2 about the camera parameters, an experiment in the calibration of these combined with radial undistortion from our model would produce even more realistic renderings. Since we have used a simpler camera model for our parameter estimation, it would also be interesting to test out radial undistortion using true camera parameters while using a more complex camera model.

Bibliography

[1] G. Bradski. “The OpenCV Library”. In: *Dr. Dobb’s Journal of Software Tools* (2000).

[2] D. Brown. “Decentering Distortion of Lenses”. In: *Photogrammetric Engineering* 32.3 (1966), pp. 444–462.

[3] Anpei Chen et al. “Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 14124–14133.

[4] The Yorkshire Knee Clinic. *What is The ACL? Dave Duffy Explains*. [Online; accessed 13 sep, 2013]. 2020. URL: `https://www.youtube.com/watch?v=T76xzRBlX5E`.

[5] A. Conrady. “Decentered Lens Systems”. In: *Monthly Notice of the Royal Astronomical Society* 79 (1919), pp. 384–390.

[6] M. Fischler and R. Bolles. “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography”. In: *Communications of the ACM* 24.6 (1981), pp. 381–395. URL: `https://dl.acm.org/doi/pdf/10.1145/358669.358692`.

[7] Aritra Roy Gosthipaty and Ritwik Raha. *Computer Graphics and Deep Learning with NeRF using TensorFlow and Keras: Part 2*. `https://pyimagesearch.com/2021/11/17/computer-graphics-and-deep-learning-with-nerf-using-tensorflow-and-keras-part-2/`. Accessed: 12 Oct 2023. 2021.

[8] Kenji Hata and Silvio Savarese. *CS231A Course Notes 1: Camera Models*. `https://web.stanford.edu/class/cs231a/course_notes/01-camera-models.pdf`. [Online; accessed 27-Sep-2023]. 2023.

[9] Alain Horé and Djemel Ziou. “Image Quality Metrics: PSNR vs. SSIM”. In: *2010 20th International Conference on Pattern Recognition*. 2010, pp. 2366–2369. DOI: `10.1109/ICPR.2010.579`.

[10] Yoonwoo Jeong et al. *Self-calibrating neural radiance fields*. Sept. 2021. URL: `https://arxiv.org/abs/2108.13826`.

[11] Chiyu "Max" Jiang et al. “Local Implicit Grid Representations for 3D Scenes”. In: *CoRR* abs/2003.08981 (2020). arXiv: `2003.08981`. URL: `https://arxiv.org/abs/2003.08981`.

[12] Juho Kannala and Sami Brandt. “A Generic Camera Model and Calibration Method for Conventional, Wide-Angle, and Fish-Eye Lenses”. In: *IEEE transactions on pattern analysis and machine intelligence* 28 (Sept. 2006), pp. 1335–40. DOI: `10.1109/TPAMI.2006.153`.

[13] kwea123. *NeRF-pytorch-lightning*. `https://github.com/kwea123/nerf_pl/tree/nerfw`. 2022.

[14] Nathi Magubane. *The hidden costs of AI: Impending energy and resource strain.* Mar. 2023. URL: https://penntoday.upenn.edu/news/hidden-costs-ai-impending-energy-and-resource-strain.

[15] Ricardo Martin-Brualla et al. "NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections". In: *CVPR.* 2021.

[16] Nelson Max. "Optical models for direct volume rendering". In: *IEEE Transactions on Visualization and Computer Graphics* 1.2 (1995), pp. 99–108.

[17] Ben Mildenhall et al. *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis.* 2020. arXiv: 2003.08934 [cs.CV].

[18] Pedro Miraldo and Helder Araujo. *Calibration of Smooth Camera Models.* https://people.kth.se/ miraldo/pub/2012tpami.pdf. Online;accessed 10-October-2023.

[19] Thomas Müller et al. "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding". In: *ACM Trans. Graph.* 41.4 (July 2022), 102:1–102:15. DOI: 10.1145/3528223.3530127. URL: https://doi.org/10.1145/3528223.3530127.

[20] OrthoInfo. *Arthroscopy.* 2021. URL: https://orthoinfo.aaos.org/en/treatment/arthroscopy/ (visited on 06/13/2023).

[21] The Swedish national knee ligament registry. *The Swedish national knee ligament registry, 2021 Annual Report.* https://aclregister.nu/media/uploads/Annual%20reports/rapport2021.pdf. 2021.

[22] Antoni Rosinol, John J. Leonard, and Luca Carlone. *NeRF-SLAM: Real-Time Dense Monocular SLAM with Neural Radiance Fields.* 2022. arXiv: 2210.13641 [cs.CV].

[23] Johannes Lutz Schönberger and Jan-Michael Frahm. "Structure-from-Motion Revisited". In: *Conference on Computer Vision and Pattern Recognition (CVPR).* 2016.

[24] Scratchapixel. *Computing the pixel coordinates of a 3D point.* Oct. 2022. URL: https://scratchapixel.com/lessons/3d-basic-rendering/computing-pixel-coordinates-of-3d-point/mathematics-computing-2d-coordinates-of-3d-points.html.

[25] W.E. Smith, N. Vakil, and S.A. Maislin. "Correction of distortion in endoscope images". In: *IEEE Transactions on Medical Imaging* 11.1 (1992), pp. 117–122. DOI: 10.1109/42.126918.

[26] Wenko Süptitz and Sophie Heimes. "Photonics: Technical Applications of Light (Full Book)". In: https://www.spiedigitallibrary.org/ebooks/PM/Photonics-Technical-Applications-of-Light/1/Photonics-Technical-Applications-of-Light-Full-Book/10.1117/3.2507083.sup. SPIE, May 15, 2016. DOI: 10.1117/3.2507083.sup. URL: https://lens.org/028-027-070-649-38X.

[27] OpenCV dev team. *Camera Calibration and 3D Reconstruction.* http://docs.opencv.org/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html. 2019.

[28] OpenCV dev team. *Camera Calibration and 3D Reconstruction.* https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html. 2019.

[29] OpenCV dev team. *Fisheye camera - Camera Calibration and 3D Reconstruction.* `https://docs.opencv.org/4.x/db/d58/group__calib3d__fisheye.html`. 2023.

[30] Edgar Tretschk et al. “Non-Rigid Neural Radiance Fields: Reconstruction and Novel View Synthesis of a Dynamic Scene From Monocular Video”. In: *IEEE International Conference on Computer Vision (ICCV).* IEEE. 2021.

[31] Zhou Wang et al. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. DOI: `10.1109/TIP.2003.819861`.

[32] Zian Wang et al. “Adaptive Shells for Efficient Neural Radiance Field Rendering”. In: *ACM Trans. Graph.* 42.6 (2023). DOI: `10.1145/3618390`. URL: `https://doi.org/10.1145/3618390`.

[33] Wikipedia contributors. *Mask (computing) — Wikipedia, The Free Encyclopedia.* `https://en.wikipedia.org/w/index.php?title=Mask_(computing)&oldid=1178534310`. [Online; accessed 6-October-2023]. 2023.

[34] Lin Yen-Chen. *NeRF-pytorch.* `https://github.com/yenchenlin/nerf-pytorch/`. 2020.

[35] Your Practice Online Education. *Primary ACL Repair Video.* 2023. URL: `https://www.ypo.education/orthopaedics/knee/primary-acl-repair-t622/video/`.

[36] YU Yue. *NeRF: A Volume Rendering Perspective.* `https://yconquesty.github.io/blog/ml/nerf/nerf_rendering.html`. 2022.

[37] Richard Zhang et al. *The unreasonable effectiveness of deep features as a perceptual metric.* Apr. 2018. URL: `https://arxiv.org/abs/1801.03924`.

[38] Yi Zhou et al. “On the Continuity of Rotation Representations in Neural Networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* June 2019.

A

Appendix 1

Algorithm 2 Joint Optimization of Color Consistency Loss in Self Calibration NeRF and Ray Distance Loss using Curriculum Learning

```
 1: Initialize NeRF parameter $\Theta$
 2: Initialize camera parameter $z_K, z_R|t, z_{rayo}, z_{rayd}, z_k$
 3: Learnable Parameters $S = \Theta$
 4: for $iter = 1, 2, \ldots$ do
 5:     S' = get_params(iter)
 6:     $\mathbf{r_d}, \mathbf{r_o} \leftarrow$ camera model(K,z)
 7:     $\mathcal{L} \leftarrow$ volumetric rendering($\mathbf{r_d}, \mathbf{r_o}, \boldsymbol{\Theta}$)
 8:     if iter$\%n$ == 0 and iter $\geq n_{prd}$ then
 9:         $I' \leftarrow$random($R_I, t_I, \mathcal{I}$)
10:         $\mathcal{C} \leftarrow$ Correspondance($I, I'$)
11:         $\mathcal{L}_{prd} \leftarrow$ Projected ray distance($\mathcal{C}$)
12:         $\mathcal{L} \leftarrow \mathcal{L} + \lambda\mathcal{L}_{prd}$
13:     end if
14:     for $s \in S'$ do
15:         $\mathbf{s} \leftarrow \mathbf{s} + \nabla_s\mathcal{L}$
16:     end for
17: end for
```

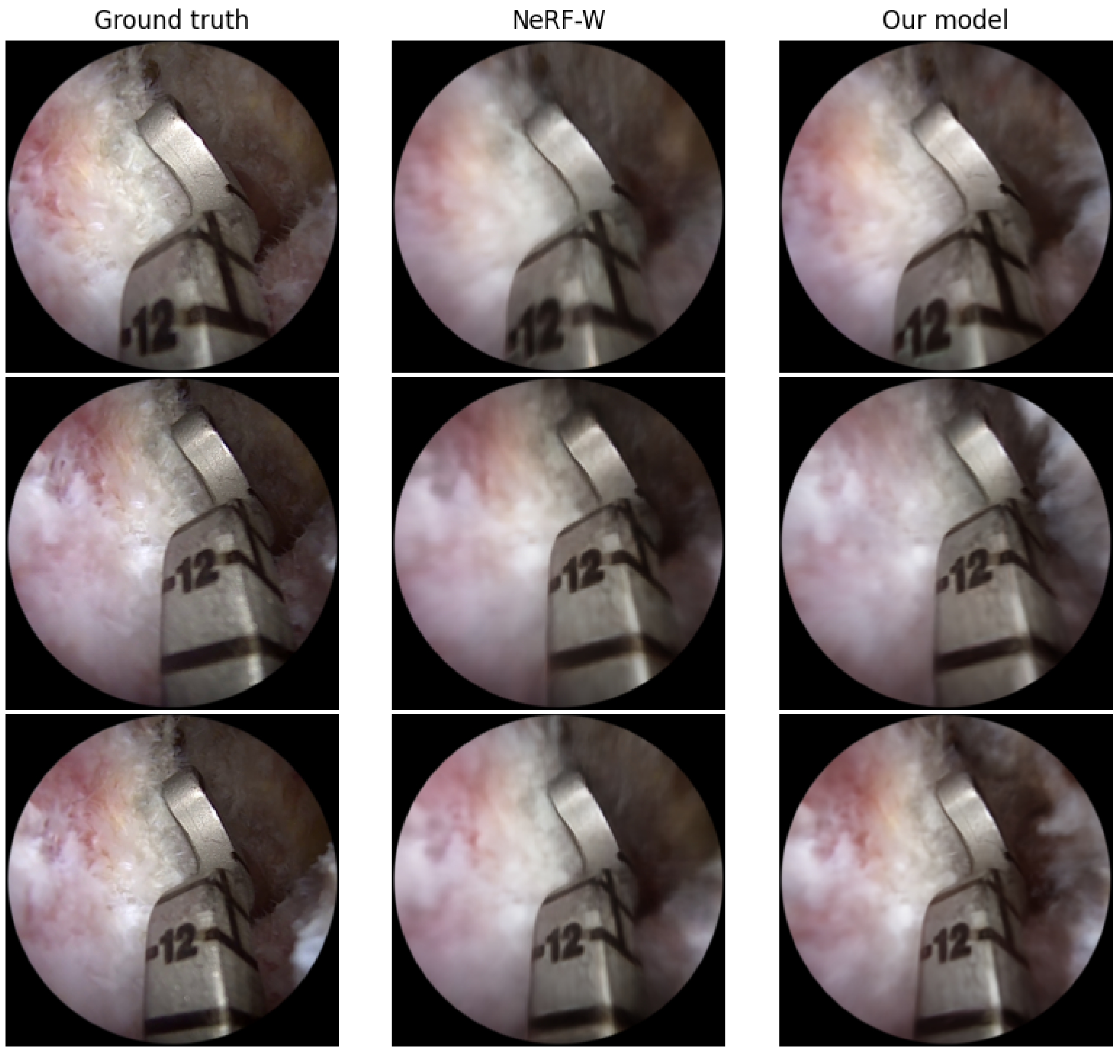

Figure A.1: Comparison of ground truth images and renders produced by NeRF-W and our model. We see that our model has rendered some artifacts that resemble organic matter on the right-hand side, which is not present in the renderings of NeRF-W or the ground truths.