Exploring Deep Learning for Accurate Facial 3D Reconstruction

Manu Vats
ID: 2545697
AC52010 Msc. Project
MSc. in Data Science and Engineering
University of Dundee, 2024
Supervisor: Dr. Ludovic Magerand

Abstract - *Getting an accurate 3D model of a person's face is essential for remote consultations in fields like dentistry and medicine. Recent advancements in computer vision have yielded impressive results, achieving very high levels of precision. Now, there's a rising interest in using deep learning for 3D reconstruction. The aim of this project is to look at the latest deep learning methods for facial 3D reconstruction and figure out their limitations.*
It was initially planned to look at how cropping of the images and background masking affect the quality of the learned representation of the scene. But due to the limited availability of time the project was focused on effects of cropping.
Different experiments which involved cropping the images in varying proportions and changing the center points of the image and feeding the images into the model to see the effect of cropping on the quality of the 3D reconstruction were performed. The results were then compared and helped to conclude.

1 Introduction

3D facial reconstruction is a field which has a lot of applications in various industries. Few domains where facial 3D reconstruction has been used are Medicine, Forensics, Entertainment, Anthropology and Security.
In Forensic Science, 3D facial reconstruction has been used to aid in the recognition of unidentified remains of deceased individuals or suspects based on the skeletal remains. This is very useful especially in cases where traditional 2D methods are insufficient.
Creating accurate 3D models of faces for identification has applications in Security and Biometrics as well. Facial recognition has been a widely researched topic in Security and biometrics.
3D facial reconstruction has widespread applications in the Entertainment industry where it is used in realistic digital models of actors for animation, special effects, and facial motion capture. With the rising prominence of Augmented and Virtual Reality, the applications of 3D facial reconstruction are found in creating realistic avatars and projecting digital information onto faces.
Anthropology has also seen applications of 3D reconstruction particularly in reconstructing the faces of ancient human remains to understand the physical characteristics and lifestyles of ancient human beings.
The area of interest for this project is in the medical or dental field. 3D reconstructions help craniofacial surgeons to visualize the patient's anatomy and plan surgical operations for correcting the facial and skull deformities with a high degree of accuracy and avoid possible failures.
The classical techniques in 3D reconstruction which have been developed over several decades like Stereo Vision, Structure from Motion (SfM), Depth Sensors, Shape from Shading, Photogrammetry and Point Cloud Reconstruction are still used as foundations for modern techniques of computer vision. However, these classical techniques were out of scope of this project.
Now there has been a rising interest in employing Deep Learning networks for 3D reconstruction. One of the models is the state-of-the-art **Neural Radiance Fields (NeRF)** model which was introduced in 2020. It uses a fully connected and a non-convolutional deep network to create photorealistic novel views by synthesizing existing viewpoints. The algorithm takes a 5D coordinate input which comprises of spatial location and viewpoint of the camera and produces an output comprising of volume density of color or view-dependent emitted radiance of the same location (Mildenhal, Srinivasan and Tancik).
The aim of this project initially was to look at the latest deep learning methods for facial 3D reconstruction and figure out their limitations. Therefore, the focus of the project was to investigate the ways to focus on the face rather than the body and the background to create 3D reconstruction of the face. It was initially planned to look at how cropping of the images and background masking affect the quality of the learned representation of the scene. But due to the limited availability of time, the project was focused on effects of cropping. A series of experiments were performed which involved cropping the images in varying proportions and feeding the images into the model. The results of these experiments were compared to conclude that the model was unsuccessful in reconstructing an accurate 3D facial structure using the cropped images.
The dataset used for this project was HUMBI which is a large multi-view image dataset of human body expressions (gaze, face, hand, body, and garment) with natural clothing (Yoon, Yu and Park).

2 Background

2.1 Applications of 3D Reconstruction

Due to its applications in various domains, 3D reconstruction has seen a lot of research done on it. Forensic Facial Reconstruction (Caroline Wilkinson, 2004) is a comprehensive work which evaluates different reconstruction methods. It also delves into the relationship between the soft and hard tissues of the face, challenges faced in reconstruction and tissue variations according to different factors like age, sex, stature, and ethnicity. The latest work Methods of forensic facial reconstruction and human identification: historical background, significance, and limitations (Guleria, Krishan and Sharma) analyzes different methods developed and practiced till date for human facial reconstruction.

(Gordon) stressed on the need of 3D information present in image sequences for 3D pose estimation and explored motion-based face estimation which computes the 3D pose of the head in each frame of a video sequence useful in facial recognition for security and biometrics.

3DFaceGAN (Moschoglou, Ploumpis and Nicolaou), was the first GAN (Generative Adversarial Network) tailored towards modeling the distribution of 3D facial surfaces which captures non-linearity and high frequency details of 3D face shapes unlike the Principal Component Analysis (PCA) and other predecessor techniques and makes up for scarcity of publicly available 3D face databases and the challenges presented by mesh convolutions for deep networks.

Pictonaut, a novel method to automatically synthetize animated shots from motion picture footage was presented recently, which is a hybrid approach combining both the above-mentioned approaches, multi-person 3D human pose estimation and GANs (Tous).

The book Craniofacial Identification (Wilkinson and Rynn) examines all aspects of facial identification for identifying the dead for anthropological as well archaeological purposes, including the determination of facial appearance from the skull, comparison of the skull with the face and the verification of living facial images. The latest development in this area facilitates the building of 3D resources for craniofacial identification using Computed Tomography (CT) Data (Ehrhardt, Falsetti and Falsetti).

Related research in the medical field has led to advancements in techniques to facilitate treatments in various disciplines. Developments in computer technology have made the 3-dimensional (3D) diagnosis of facial soft tissues possible which is useful for 3D soft-tissue evaluation in orthodontic treatment planning and post-treatment results (Baik, Jeon and Lee). Also, a (Kwon, Choi and Kim) to understand the age-related morphological changes of facial soft tissue which significantly helps in achieving improved outcomes of rejuvenating procedures.

2.2 Classical Methods of 3D Reconstruction

The classical methods of 3D reconstruction like Stereo Vision, Structure from Motion (SfM), Depth Sensors, Shape from Shading, Photogrammetry and Point-Cloud Reconstruction have been around for decades.

The book Computer Vision (Szeliski) and 3D Computer Vision breaks down different concepts related to Stereo Vision in detail. A lot of improvements in Stereo Vision like (Lucas and Kanade) integrating the geometric constraints of stereo vision with implicit function representation of Pixel Aligned Implicit Function (Chan, Liu and Lin) and (Ramírez-Hernández, Rodríguez-Quiñonez and Castro-Toscano) have been proposed especially in the last decade.

Many improvements have been suggested to the Structure from Motion technique. A new technique of Structure from Motion (SfM) improves robustness, accuracy, completeness, and scalability of the original SfM (Schönberger and Frahm). A (Westoby, Brasington and Glasser).

Depth Sensors techniques and its applications like In-Home Older Adults' Activity Pattern Monitoring (Momin, Sufian and Barman) and (Waheed, Javeed and Jalal) have been explored.

Shape from Shading, which was first introduced in 1989 (Horn and Brooks), has been proposed to be used with Stereo Vision to improve endoscopic imaging accuracy and reduce the difficulty of minimally invasive surgery (Cao, Wang and Zheng). Also, a new technique had been proposed to be used with (Wang, Wu and Wang). Photogrammetry has been used in combination with Structure from Motion technique for geoscience applications (Westoby, Brasington and Glasser) as already stated above.

Point-Cloud reconstruction techniques, which is one of the more popular classical techniques for 3D reconstruction, has seen a lot of research suggesting improvements and novel methods to enhance quality of reconstruction and reduce time and computational costs. Some of them are a new learning-based Contrastive Adversarial Loss (CALoss) to measure the point cloud reconstruction loss dynamically in a non-linear representation space (Huang, Ding and Zhang), a (Li, Zhu and Lu) and a part-oriented point cloud reconstruction framework which reconstructs the point cloud for individual object parts, and combines the part point clouds into the complete object point cloud thereby reducing the number of learning network parameters and effectively minimizing the calculation time cost and the required memory space.

2.3 Deep Learning Methods for 3D Reconstruction

Since the deep learning methods for 3D reconstruction started gaining popularity, there have been several research papers exploring different logic and foundational methods for 3D reconstruction using deep learning. Based on underlying tasks, the deep learning methods can be classified under the following categories (Samavati and Soryani)-:

- Depth Estimation based: The primary task of the deep learning methods is depth estimation of pixels using 2D images (Samavati and Soryani).

- Reconstructing in the voxel format: The models based on these methods discretize the 3D space into a grid of uniform cubic parts (Samavati and Soryani). This type of representation easily fits into deep learning frameworks (Samavati and Soryani).
- Point-cloud-based 3D reconstruction: Methods that use point cloud representation perform better at fixed memory usage than those representing shapes in voxel grids. (Samavati and Soryani).
- Surface based 3D reconstruction: Surface-based representations are deformable and require fewer resources than voxel-based representations (Samavati and Soryani).
- 3D reconstruction using implicit representation: A surface can be implicitly modeled as the zero-level set of a function that is learned by a Multi-Layer Perceptron. The implicit representations have high memory efficiency; therefore, they can represent shapes at high resolutions. However, the inference time of these approaches is considerable, especially at higher resolutions and the models only accept single-view inputs and do not consider a pipeline for multi-view inputs (Samavati and Soryani).
- Multiple object reconstruction: These models are more suitable for real-world tasks such as robotic navigation or manipulation and 3D scene understanding which require inferring the 3D shape of multiple objects or even an entire scene.
- Multi-view stereo: This refers to the task of reconstructing a 3D shape from calibrated overlapping images captured from different viewpoints (Sinha). Various representations can be used in such algorithms, depending on the application. Most of the learning-based methods use depth maps or volumetric representations (Samavati and Soryani)
- GANs: The adversarial training strategy in GANs contributes to high-quality outputs and compensates to a certain extent for the lack of public databases having high resolution images (Samavati and Soryani).

Some studies have used 3D reconstruction as a downstream task to accomplish other goals. These methods mainly belong to the fields of object pose estimation, novel view prediction (NeRF by (Mildenhal, Srinivasan and Tancik)), 3D semantic segmentation, color reconstruction, and estimating lighting conditions for rendering synthetic objects in a scene (Samavati and Soryani).

The review paper, Deep learning-based 3D reconstruction: a survey by (Samavati and Soryani) also highlights the different kinds of loss functions for training deep learning-based 3D reconstruction algorithms. They are based on the model's output representation (Volumetric based losses, /negative intersection over union (IoU), Cross-entropy, Point-based losses, Mesh-based losses, Losses for 2D supervision). The paper also highlighted evaluation metrics used to evaluate the performance of algorithms. The most common of them are IoU for voxel representation and both CD (Chamfer Distance) and EMD (Earth Mover's Distance) for point cloud and mesh representations. The F1-Score is also known as a robust metric for evaluating reconstruction quality.

2.4 Neural Radiance Fields (NeRF)

We focused on using Neural Radiance Fields based deep network (NeRF) for this project. NeRF was introduced by Mildenhall et al. (2020) which laid the foundation for all the subsequent NeRF models. NeRF models are novel view synthesis methods which use volume rendering with (typically) implicit neural scene representation via Multi-Layer Perceptrons (MLPs) to learn the geometry and lighting of a 3D scene. Since then, NeRF has become a significant development in the field of Computer Vision, allowing for implicit, neural network-based scene representation and novel view synthesis. (Mildenhal, Srinivasan and Tancik)

A static scene is represented as a continuous 5D function that provides the radiance emitted in each direction (θ, φ) at every point (x, y, z) in space, along with a density value indicating the opacity of that point. This function is modeled using a deep fully connected neural network, without convolutional layers, commonly referred to as a multilayer perceptron (MLP). The network is trained to map a single 5D coordinate (x, y, z, θ, φ) to a volume density and view dependent RGB color. (Mildenhal, Srinivasan and Tancik)

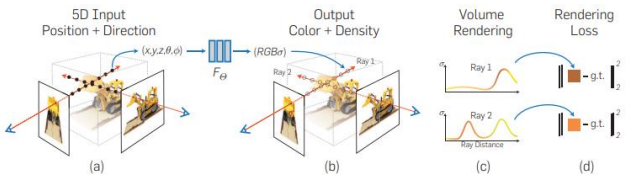


Figure 1. An overview of our neural radiance field scene representation and differentiable rendering procedure.

In the figure above, the images are synthesized by sampling 5D coordinates (location and viewing direction) along camera rays (a), those locations are fed into an MLP to produce a color and volume density (b) and using volume rendering techniques, these values are composite into an image (c). This rendering function is differentiable, so the scene representation can be optimized by minimizing the residual between synthesized and ground truth observed images (d) (Mildenhal, Srinivasan and Tancik).

To render this neural radiance field (NeRF) from a specific viewpoint-:

1. Camera rays are traced through the scene to generate a sampled set of 3D points.
2. Next, we feed these points along with their corresponding 2D viewing directions into the neural network to obtain output colors and densities.

3. Finally, we utilize traditional volume rendering techniques to combine these colors and densities into a 2D image.
(Mildenhal, Srinivasan and Tancik)

This rendering process is inherently differentiable, allowing us to optimize the model using gradient descent. By minimizing the error between observed images and the corresponding views rendered from our representation, we encourage the network to predict a coherent model of the scene. This involves assigning high-volume densities and accurate colors to locations that accurately represent the underlying scene content. (Mildenhal, Srinivasan and Tancik)
This model facilitates the creation of realistic renderings of complex scenes from sparse 2D image data, enabling applications in computer graphics, virtual reality, and augmented reality (Mildenhal, Srinivasan and Tancik).
It was found out that the initial approach of optimizing a neural radiance field representation for intricate scenes falls short in achieving a sufficiently detailed outcome. To overcome this limitation, positional encodings were introduced to transform input 5D coordinates. This adjustment allows the MLP to effectively capture higher-frequency functions, resulting in a more refined representation. (Mildenhal, Srinivasan and Tancik)

2.5 Evaluation Metrics

The Metrics which are used to measure the quality of the synthesized novel images are-:

- **Peak Signal to Noise Ratio (PSNR)**-:

  PSNR is a no-reference quality assessment metric,

$$\textbf{PSNR(I)} = 10 \cdot \log_{10} (MAX(I)^2 / MSE(I))$$

where **MAX(I)** is the maximum possible pixel value in the image (255 for 8bit integer), and **MSE(I)** is the pixelwise mean squared error calculated over all colour channels. PSNR is also commonly used in signal processing and is well understood. (Gao, Graduate Student Member, IEEE and Gao)

- **Structural Similarity Index Measure (SSIM)** –

  SSIM is a full-reference quality assessment metric. For a single patch, this is given by-:

$$\textbf{SSIM(x, y)} = ((2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)) / ((\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2))$$

where $C_i = (K_iL)^2$, L is the dynamic range of the pixels (255 for 8bit integer), and
$K_1 = 0.01$, $K_2 = 0.03$ are constants chosen by the original authors. (Gao, Graduate Student Member, IEEE and Gao)

2.6 Review of NeRF Models

NeRF models have found diverse applications in robotics, urban mapping, autonomous navigation, virtual reality/augmented reality, and more. (Gao, Graduate Student Member, IEEE and Gao).
(Gao, Graduate Student Member, IEEE and Gao) categorized select models on certain innovations as well as applications. Following are the criteria used to categorized NeRF models-:

- Improvements in the Quality of Synthesized Views and Learned Geometry: Some models have aimed to enhance the photometric and geometric aspects of NeRF view synthesis and 3D scene representation, resulting in improved image quality in synthesized views. Some of the models which innovated on better view synthesis are Mip-NeRF, Ref-NeRF and Rap-NeRF. By using supervision of expected depth with point clouds acquired from LiDAR or SfM, these models converge faster, converge to higher final quality, and require fewer training views than the baseline NeRF model. Some of these models are DS-NeRF, NerfingMVS and PointNeRF. Some models have improved in other geometrical aspects. SNeS improved geometry by learning probable symmetries for partly symmetric and partly hidden in-scene objects through soft symmetry constraints on geometry and material properties. S3-NeRF used shadow and shading to infer scene geometry and achieved single image NeRF training, focusing on geometry recovery (Gao, Graduate Student Member, IEEE and Gao).

- Improvements to Training and Inference Speed: This category of models attempts to improve NeRF training and inference speed and can be broadly divided into the two following categories (Gao, Graduate Student Member, IEEE and Gao)-:
  - Baked: Models in the first category train, precompute, and store NeRF MLP evaluation results into more easily accessible data structures. Some examples are Sparse Neural Voxel Grid (SNeRG), PlenOctree, FastNeRF, KiloNeRF, Fourier Plenoctree, MobileNeRF and Efficient NeRF (Gao, Graduate Student Member, IEEE and Gao).
  - Non-baked: This category includes multiple types of innovations. These models commonly (but not always) attempt to learn separate scene features from the learned MLP parameters. Other techniques such as ray termination, empty space skipping, and/or hierarchical sampling. These are also often used to further improve training and inference

speed in conjunction to per-paper innovations. Some of these models are Neural Sparse Voxel Fields (NSVF), Deterministic Integration for Volume Rendering (DIVeR) and Instant NGP (Gao, Graduate Student Member, IEEE and Gao).

- Few Shot/Sparse Training View NeRF: These methods leverage pretrained image feature extraction networks and even use depth/3D geometry supervision sometimes. These models often have lower training time compared to baseline NeRF models. Examples include pixelNeRF, MVSNeRF, DietNeRF, NeuRay, GeoNeRF, RegNeRF, NeRFusion, AutoRF and SinNeRF (Gao, Graduate Student Member, IEEE and Gao).
- Generative and Conditional Models: The generative NeRF models generate 3D geometry conditioned on texts, images, or latent codes and can be classified into-:
    - Generative Adversarial Network-based – GRAF, pi-GAN, EG3D, StyleNeRF, Pix2NeRF (Gao, Graduate Student Member, IEEE and Gao)
    - Diffusion Methods – DreamFusion, Latent-NeRF, Magic3D, RealFusion, Neurallift-360, NeRFDi, NerfDiff and DiffusioNeRF (Gao, Graduate Student Member, IEEE and Gao).
    - Jointly optimized latent models – Edit-NeRF and CLIP-NeRF (Gao, Graduate Student Member, IEEE and Gao).
- Unbounded Scene and Scene Composition: The models of this category approached this problem of additional challenges in image-by-image variation in lighting and appearance posed by outdoor scenes using various methods. Some of them are NeRF-W, NeRF++, GIRAFFE, Fig-NeRF and NeRFReN (Gao, Graduate Student Member, IEEE and Gao).
- Pose Estimation: This category of models performs both pose estimation and implicit scene representation with NeRF. For eg. iNeRF, NeRF--, BARF, GNeRF, GARF and NeRF-SLAM (Gao, Graduate Student Member, IEEE and Gao).
- Adjacent Methods for Neural Rendering:
    - Explicit Methods: Fast MLP-less Volume Rendering: Pleoxel, DVGO, TensoRF, Streaming Radiance Fields (Gao, Graduate Student Member, IEEE and Gao).
    - Others Neural Volume Rendering: IBRNet, Scene Rendering Transformer (SRT) and NeRFormer (Gao, Graduate Student Member, IEEE and Gao).

Gao et al. also classified the models based on the specific applications of NeRF which the innovations of those models focused on.

- Urban: Given current interests in autonomous driving and urban photogrammetry, there has been a recent surge in the development of urban NeRF models, both from street-level and remote-sensing views. The models which focus on these applications are Urban Radiance Fields, MegaNeRF, Block-NeRF (Gao, Graduate Student Member, IEEE and Gao).
- Human Faces and Avatars, and Articulated Objects: This is the application which our project is focused on. A few examples of published models in this category are Nerfies, HyperNeRF, CoNeRF, Neural Body and PREF (Gao, Graduate Student Member, IEEE and Gao).
- Image Processing: Some examples are RawNeRF, HDR-NeRF and SemanticNeRF (Gao, Graduate Student Member, IEEE and Gao).
- Surface Reconstruction: For certain NeRF applications, more explicit representations, such as 3D mesh are desired. The methods introduced in this category used innovative scene representation strategies that change the fundamental behavior of the density MLP. Some of them are UNISURF, Neural Surface, HF-Neus, GeoNeus and Sparse Neus (Gao, Graduate Student Member, IEEE and Gao).

2.7 Nerfstudio framework

To streamline the development and deployment of NeRF research, a modular PyTorch framework, Nerfstudio was proposed by (Tancik, Weber and Ng). This framework includes plug-and-play components for implementing NeRF-based methods, which make it easy for researchers and practitioners to incorporate NeRF into their projects. The modularity of Nerfstudio enabled the development of Nerfacto, a method that combines components from recent papers to achieve a balance between speed and quality, while also remaining flexible to future modifications. This method, Nerfacto, was used to perform all the experiments and was kept unchanged throughout the experiments.

2.8 Datasets

The original NeRF paper presented a synthetic dataset created from Blender (referred to as Realistic Synthetic 360) (Mildenhall et al. 2020). This is often the first dataset considered by NeRF researchers, are the scenes are well bounded, focused on single object, and the dataset is well benchmarked against known models. But apart from this, several other datasets have been proposed for different purposes. For eg. LIFF Dataset (Ben, Srinivasan and Ortiz-Cayon) consisting of 24 real-life scenes captured from handheld cellphone cameras, Tanks and Temples dataset (Knapitsch, Park and Zhou), Nerfies (Park, Sinha and Barron)), etc.

The dataset used for this project is HUMBI dataset (Yoon, Yu and Park). HUMBI is a large multi-view image dataset of human body expressions (gaze, face, hand, body, and garment) with natural clothing. 107 synchronized HD cameras were used to capture more than 700 subjects across gender, ethnicity, age, and style.

3 Specification

The project required running a series of experiments which involved training a fully connected deep learning model on a dataset of images. Following are the recommended specifications for such a project-:

CPU	
Cores *Cores are hardware components that operate as independent processing units in a CPU. The higher the number of cores, the higher the performance is.*	8
Threads *Threads are virtual (logical) cores that come due to Hyper-Threading technology. Imagine a core as a brain and a thread as a thought. If a core has two threads, a computer of 6 cores has 6 brains that can process 12 thoughts at a time.*	16
Max. Clock Speed (Turbo Frequency) *It is the maximum speed of the CPU. The higher the max frequency, the faster is the CPU.*	2.30 GHz.
Processor	11th Gen Intel(R) Core (TM) i7-11800H

RAM	
RAM size *Higher the RAM size, the better is the system at multi-tasking.*	16 GB
RAM bus speed *It is the number of bits transfers done per second. The higher the bus speed, the faster is the system.*	3200 MHz

GPU	
CUDA Compatibility *All NVIDIA GPUs are CUDA compatible. So, a system with NVIDIA GPU is recommended to be chosen.*	Yes
Number of CUDA Cores	2560
Number of Tensor Cores	80
Number of Raytracing Cores	20
Higher the number of cores, the better the system performs at Machine Learning and Deep Learning tasks.	
GPU Memory	4 GB

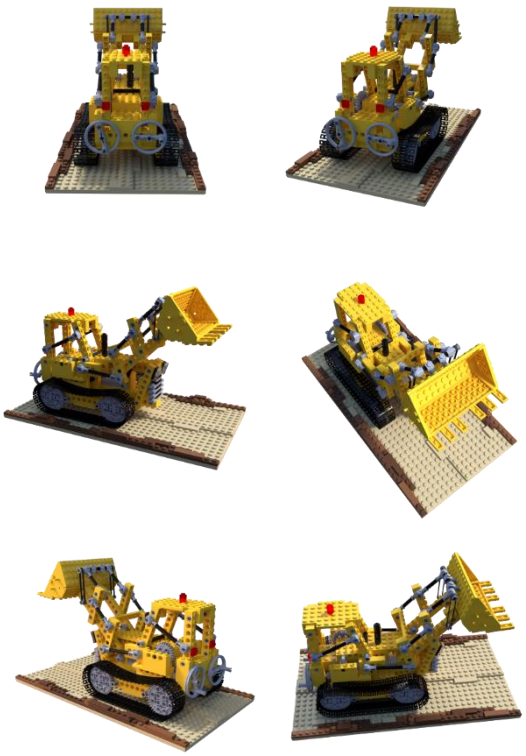
Storage	
Storage Type *SSDs or Solid-State Drives provide significantly better performance than HDDs or Hard Disk Drives in terms of reading and writing data, program execution, etc.*	SSD
Storage size	500 GB

Other requirements used to train the model are-:

- CUDA
- Anaconda (It is a distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows.)

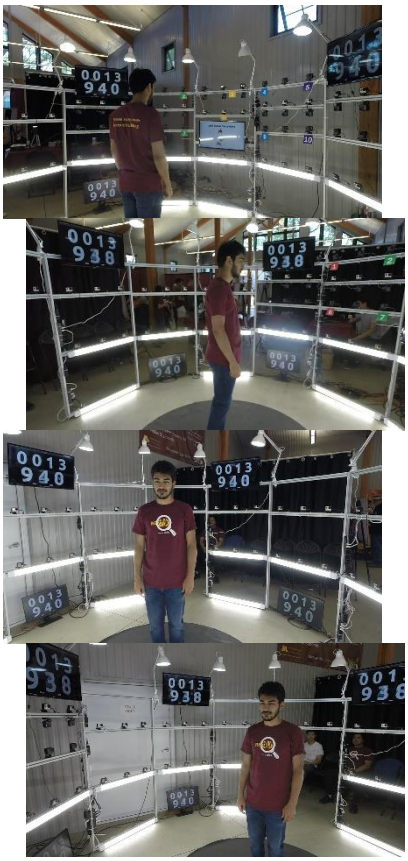
3.1 Dataset Used

D-NeRF dataset was first used to see the initial runs of the model on sample datasets.
D-NeRF has images of different objects with approximately 8 scenes each and the number of images for each object is less than 250.
D-NeRF dataset has low dynamic range (LDR).
Following are the sample images from lego part of D-NeRF dataset-:



After training on sample dataset, the dataset used for this project was HUMBI dataset (Zhixuan Yu et al. 2020). HUMBI is a large multi-view image dataset of human body expressions (gaze, face, hand, body, and garment) with natural clothing. 107 synchronized HD cameras were used to capture more than 700 subjects across gender, ethnicity, age, and style. With the multi-view image streams, we also get reconstruction of high-fidelity body expressions using 3D mesh models. However, the 3D mesh models were out of the scope of this project. HUMBI is highly effective in learning and reconstructing a complete human model and is complementary to the existing datasets of human body expressions with limited views and subjects such as MPII-Gaze, Multi-PIE, Human3.6M, and Panoptic Studio datasets.
One pose of one subject of HUMBI dataset was used to train the models in all experiments. The images were selected where at least a little part of the face, be it from the sides is visible.
Number of images = 57

Following are some images from the dataset-:



4 Design

The project required running a series of experiments which involved training a fully connected deep learning model on a dataset of images. Following are the recommended specifications for such a project-:

CPU	
Cores *Cores are hardware components that operate as independent processing units in a CPU. The higher the number of cores, the higher the performance is.*	8
Threads *Threads are virtual (logical) cores that come due to Hyper-Threading technology. Imagine a core as a brain and a thread as a thought. If a core has two threads, a computer of 6 cores has 6 brains that can process 12 thoughts at a time.*	16
Max. Clock Speed (Turbo Frequency) *It is the maximum speed of the CPU. The higher the max frequency, the faster is the CPU.*	2.30 GHz.
Processor	11th Gen Intel(R) Core (TM) i7-11800H

Table 1. CPU Requirements

RAM	
RAM size *Higher the RAM size, the better is the system at multi-tasking.*	16 GB
RAM bus speed *It is the number of bits transfers done per second. The higher the bus speed, the faster is the system.*	3200 MHz

Table 2. RAM Requirements

GPU	
CUDA Compatibility *All NVIDIA GPUs are CUDA compatible. So, a system with NVIDIA GPU is recommended to be chosen.*	Yes
Number of CUDA Cores	2560
Number of Tensor Cores	80
Number of Raytracing Cores	20
Higher the number of cores, the better the system performs at Machine Learning and Deep Learning tasks.	
GPU Memory	4 GB

Table 3. GPU Requirements

Storage	
Storage Type *SSDs or Solid-State Drives provide significantly better performance than HDDs or Hard Disk Drives in terms of reading and writing data, program execution, etc.*	SSD
Storage size	500 GB

Table 4. Storage Requirements

Other requirements used to train the model are-:

- CUDA
- Anaconda (It is a distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows.)

4.1 Dataset Used

Blender dataset was first used to see the initial runs of the model on sample datasets.

Blender has images of different objects with about 8 scenes each, and the number of images for each object is 250 to 999 (Tancik, Weber and Ng).
Blender dataset has low dynamic range (LDR).
Following are the sample images from lego part of **Blender** dataset-:

Figure 2: Samples from Lego part of Blender Dataset

After training on sample dataset, the dataset used for this project was HUMBI dataset (Yoon, Yu and Park). HUMBI is a large multi-view image dataset of human body expressions (gaze, face, hand, body, and garment) with natural clothing. 107 synchronized HD cameras were used to capture more than 700 subjects across gender, ethnicity, age, and style. With the multi-view image streams, we also get reconstruction of high-fidelity body expressions using 3D mesh models. However, the 3D mesh models were out of the scope of this project. HUMBI is highly effective in learning and reconstructing a complete human model and is complementary to the existing datasets of human body expressions with limited views and subjects such as MPII-Gaze, Multi-PIE, Human3.6M, and Panoptic Studio datasets.
One pose of one subject of HUMBI dataset was used to train the models in all experiments. The images were selected where at least a little part of the face, be it from the sides is visible.
Number of images = 57

Following are some images from the dataset-:

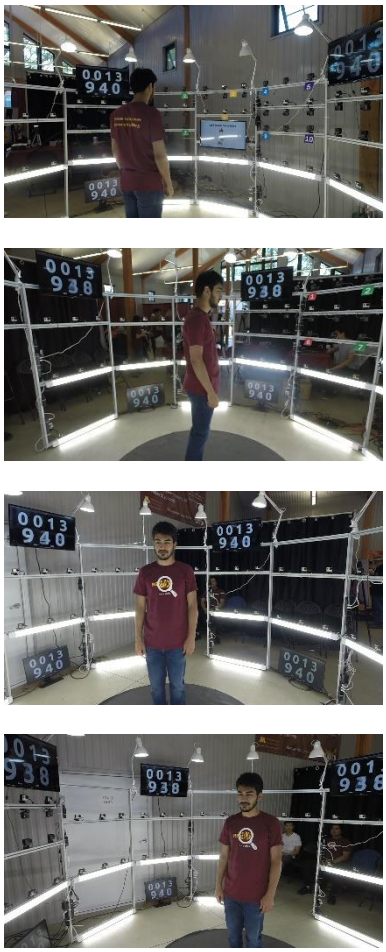


Figure 3. HUMBI Dataset

5 Implementation and Testing

To ensure the smooth working of the model after installation, NeRF was implemented on a sample dataset first. The dataset used for this implementation was the **lego** subset from the **blender**dataset which is a collection of lego construction toy images taken from different angles.
The **blender** dataset was first downloaded using the below code-:

```
# Download the full D-NeRF dataset of dynamic synthetic scenes
ns-download-data blender
```

5.1.1 Process Data

The dataset which comes with the installation of Nerfstudio doesn't need processing as they are already processed according to the data parsers and pipelines of **NeRF** models.
The dataset already comes segregated into training, validation, and test datasets. Each subset of the dataset has

its own json file which contains the camera intrinsic and extrinsic parameters and the transform matrix.

5.1.2 Training

After the processing stage, the model is trained on the processed dataset. Following is the command to train the model-:

```
#Train model
ns-train nerfacto --data data/blender/lego blender-data
```

5.1.3 Output

The output rendered video is attached in the appendix by-:

Appendix\Blender_Lego_Experiment\5.1.3 rendered_video

5.1.3.1 Evaluation Metrics

PSNR, SSIM Metrics

Metric	Value
PSNR	20.013229370117188
SSIM	0.7760244011878967

Table 5: Evaluation Metrics - Lego Dataset

The metrics can also be seen in the output file attached in the appendix as-:

Appendix\Blender_Lego_Experiment\5.1.3.1 output

5.2 Experiment 1: Using cropped images given in HUMBI dataset

HUMBI provides cropped versions of their images in its dataset where each image has just the head of the subject and not the body. In the first part of the experiment, we are using those cropped images and not cropping the full-body images on our own. Some of those images are as follows-:

Figure 4. Cropped images from HUMBI Dataset

5.2.1 Processing Data

As it has been stated before, the custom dataset needs to be processed by NeRF before the model is trained to be converted into the suitable form for NeRF. Therefore, HUMBI dataset needs to be processed before it is used for training the model. Following command line code is used to process the data-:

```
ns-process-data images --data {DATA_PATH} --output-dir {PROCESSED_DATA_DIR}
```

where-:

{DATA_PATH} – Path of the images

{PROCESSED_DATA_DIR} – Output path for processed images.

After the images are processed, a json file named transforms.json is generated which contains all the camera properties (camera type, intrinsic parameters and extrinsic parameters) and transform matrix.

The json file is attached in the appendix as -:
Appendix\5.2 HUMBI_cropped_images\5.2.1 transforms.json

The information in this json file is used by NeRF model to train.

5.2.2 Training

After the processing stage, the model is trained on the processed dataset. Following is the command to train the model-:

```
# Train model
ns-train nerfacto --data {PROCESSED_DATA_DIR}
```

5.2.3 Output

The output video will be attached in the appendix as-:
Appendix\5.2 HUMBI_cropped_images\5.2.3 rendered_video

5.2.3.1 Evaluation Metrics

Metric	Value
PSNR	13.562848091125488
SSIM	0.28983962535858154

Table 6. Evaluation Metrics - Given HUMBI Cropped dataset

The output an also be found attached in the appendix as --> **Appendix\5.2 HUMBI_cropped_images\5.2.3.1 output**

As you can see, we hardly see any visible facial reconstruction in this experiment. In the next series of experiments, we will crop the images on our own in varying proportions and modify the transfroms.json file.

5.3 Experiment 2: Cropping images with

varying principal points

5.3.1 Part 1: Cropping without head detection

5.3.1.1 Method/Logic

Images were cropped in varying proportions. Different cropping experiments were performed depending upon the proportion of cropping. The logic followed for each cropping experiment is as follows-:
Step 1: Calculate the shape (height and width) of the original image.
Step 2: Calculate the shape (height and width) of the image after reducing it by a certain percentage, let's say x%. For e.g. If the image shape is (600, 400) where height = 600 and width = 400, it is reduced by let's say 10% to (540, 360).
Step 3: Calculate the offset after cropping.

X-offset = 600-540 = 60
Y-offset = 400-360 = 50

Step 4: After getting the new dimensions or shape of the image, you need to provide the start and end points for both the x-axis and y-axis of the image. Thie means, you need to provide-:

- start_x - (starting point on x-axis from Left)
- end_x – (ending point on x-axis from Left)
- start_y - (starting point on y-axis from Top)
- end_y - (ending point on x-axis from Top)

So,
start_x = (Original Image width - Half of X-offset)/2
end_x = (Original Image width + Half of X-offset)/2
start_y = 0
We are not cropping from the top so as to avoid getting the head cropped out.
end_y = (Original Image height + Half of Y-offset)

Step 5: You feed these start and end points to the cropping command to achieve the cropped image.
In each of these cropping experiments, principal points or center points were recalculated for the image after being cropped. They were calculated as-:
New_principal_points = [Cropped_image_height/2, Cropped_image_width/2]

5.3.1.2 Code for cropping.

The code used for cropping in this experiment is attached in the appendix as --:
Appendix\5.3.1 Experiment 2\5.3.1 Cropping without head detection\5.3.1.2 Cropping.ipynb

5.3.1.3 Processing

The data generated after cropping is processed using the same command line code as mentioned in previous experiment. The json file transforms.json is generated after the processing. However, we don't use the same json file for training.
Instead, we the json file obtained after processing the full body images without cropping to maintain most of the original camera parameters of the full body images and modify the remaining ones.
The parameters we change are-:

- Principal points or center points
- Camera Type (From "OPENCY_FISHEYE" to "SIMPLE PINHOLE". "OPENCV_FISHEYE" is a curved camera which we don't need as we don't want the images to be distorted for a curved camera. We keep the camera to "SIMPLE PINHOLE")

The parameters we remove are-:

- Distortion parameters which we need to specify if we are using "OPENCV_FISHEYE" camera

The parameters we keep unchanged are-:

- Focal length of the camera
- Transform matrix

5.3.1.4 Code for modifying transforms.json

The code used for modifying **transform,json** *in this experiment is attached in the appendix as--:*
Appendix\5.3 Experiment 2\5.3.1 Cropping without head detection\5.3.1 10%_cropping\5.3.1.4 Modify_json

5.3.1.5 Training

We use the same training command line code as used in the previous experiment. We will now see all the results for varying cropping proportions.

5.3.1.6 10% cropping

In this experiment, the images were cropped by 10% of their original dimensions. Following are some samples of this dataset-:

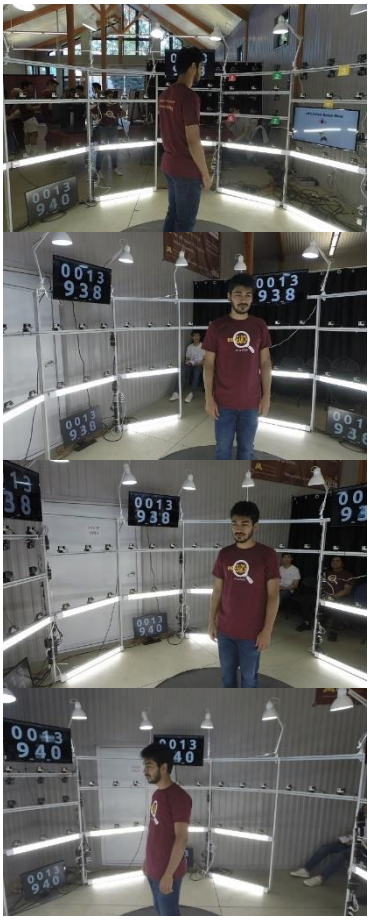


Figure 5. 10% Cropped images (Exp. 2)

5.3.1.6.1 Output

The rendered output video is attached in the appendix as-:
Appendix\5.3 Experiment 2\5.3.1 Cropping without head detection\5.3.1 10%_cropping\ 5.3.1.6.1 rendered_video

As you can see, there is a slight drop in the quality of 3D reconstruction obtained from this experiment as compared to 3D reconstruction of un-cropped images.

5.3.1.6.2 Metrics

Metric	Value
PSNR	17.138280868530273
SSIM	0.6458516716957092

Table 7. Evaluation Metrics - 10% Cropped (Exp. 2)

The metrics can also be found in the output file attached in the appendix as-:
Appendix\5.3 Experiment 2\5.3.1 Cropping without head detection\5.3.1.6 10%_cropping\5.3.1.6.2 output

We see a drop in PSNR values as compared to 3D reconstruction of un-cropped images. This suggests a drop in similarity in synthesized images obtained after training the model on cropped images.

5.3.1.7 20% cropping

In this experiment, the images were cropped by 20% of their original dimensions. Following are the samples from this dataset-:

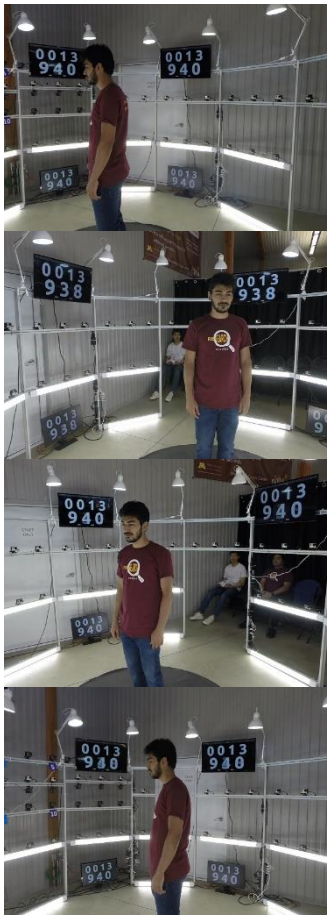


Figure 6. 20% Cropped images (Exp. 2)

5.3.1.7.1 Output

The rendered output video is attached in the appendix as-:
Appendix\5.3 Experiment 2\5.3.1 Cropping without head detection\5.3.1.7 20%_cropping\5.3.1.7.1 rendered_video

As you can see, there is a slight drop in the quality of 3D reconstruction obtained from this experiment as compared to 3D reconstruction of 10% cropped images and uncropped images.

5.3.1.7.2 Metrics

PSNR, SSIM Metrics

Metric	Value
PSNR	16.302303314208984

SSIM	0.6433947086334229

Table 8. Evaluation Metrics - 20% Cropped (Exp. 2)

The metrics can also be found in the output file attached in the appendix as-:
Appendix\5.3 Experiment 2\5.3.1 Cropping without head detection\5.3.1.7 20%_cropping\5.3.1.7.2 output
We see a drop in PSNR values as compared to 3D reconstruction of 10% cropped and un-cropped images. This suggests a drop in similarity in synthesized images after the images are cropped even more.

5.3.1.8 30% cropping

In this experiment, the images were cropped by 30% of their original dimensions. Following are the samples from this dataset-:

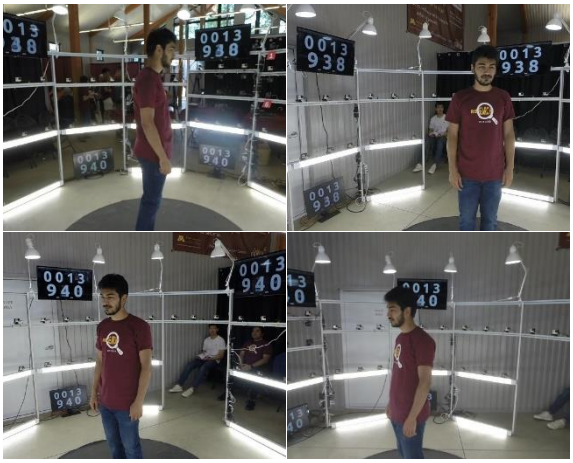


Figure 7. 30% cropped images (Exp. 2)

5.3.1.8.1 Output

The rendered output video is attached in the appendix as-:
Appendix\5.3 Experiment 2\5.3.1 Cropping without head detection\5.3.1.8 30%_cropping\5.3.1.8.1 rendered_video

As you can see, there is a slight drop in the quality of 3D reconstruction obtained from this experiment as compared to 3D reconstruction of 20% cropped images, 10% cropped images and un-cropped images.

5.3.1.8.2 Metrics

Metric	Value
PSNR	15.350258827209473
SSIM	0.6406750679016113

Table 9. Evaluation Metrics - 30% Cropped (Exp. 2)

The metrics can also be found in the output file attached in the appendix as-:
Appendix\5.3 Experiment 2\5.3.1 Cropping without head detection\5.3.1.8 30%_cropping\5.3.1.8.2 output
We see a drop in PSNR values as compared to 3D reconstruction of 20% cropped images, 10% cropped and un-cropped images. This suggests, as in previous case, a drop in similarity in synthesized images after the images are cropped even more.

5.3.2 Part 2: Cropping with head detection

5.3.2.1 Method/Logic

In this part of the experiment, the images were cropped after the head is detected in the images using object detection techniques. The model used for object detection was YOLOV5 by Ultralytics.
After the head is detected and we obtain a bounding box around the head in each of the images, we adjust the size of the bounding box to vary the proportion of cropping.
The logic followed for cropping is similar to the previous part of the experiment.
Step1: First the original dimensions of the images are known. Then the dimensions of the image are reduced by a certain percentage by which the images are needed to be cropped. For e.g., if the original image shape is (600, 400) and we need to crop the images by 40%, the image dimensions are calculated after them by 40%.
Step 2: Then we calculate the difference between the obtained cropped image dimensions and the bounding box dimensions. This difference tells us how much we need to adjust the bounding box dimensions to be same as the dimensions calculated for cropped images.
Step 3: We then adjust the dimensions of the bounding box to the required level. Here, we are cropping the top part of the images unlike in the previous experiment, but we are also taking care that the top edge of the bounding box doesn't move out of the required dimension of the image. This can happen because the head is nearer to the top edge. So, we incrementally move the top edge of the bounding box further top until it reaches the required cropped dimension at the top and stops there. And depending on how much the top edge of the bounding box has moved, we adjust the increment to the lower edge of the bounding box. The left and right edges of the bounding box are moved equally.
Step 4: Then using the adjusted bounding box dimensions, the image is cropped and fed in to the NeRF model.

5.3.2.2 Code for object detection and cropping.

The code for object detection and cropping is attached in the appendix as-:
Appendix\5.3 Experiment 2\5.3.2 Cropping with head detection\ 5.3.2.2 head_detection_and_cropping.ipynb

5.3.2.3 Processing

The data generated after cropping is processed using the same command line code as mentioned in previous experiment. The json file transforms.json is generated after the processing. Here too, we don't use the same json file for training. We use the json file obtained after processing the full body images without cropping to maintain most of the original camera parameters of the full body images and modify the remaining ones.
The same parameters as in previous experiments are changed, kept constant or removed.

5.3.2.4 Code for modifying transforms.json

The code for modifying **transform.json** *is attached in the appendix as-:*
Appendix\6.3 Experiment 2\6.3.2 Cropping with head detection\ 6.3.2.4 Modify_json.ipynb

5.3.2.5 Training

We use the same training command line code as used in the previous experiment. We will now see all the results for varying cropping proportions.

5.3.2.6 40% cropping

In this experiment, the images were cropped by 40% of their original dimensions. Following are some samples of this dataset-:

Figure 8. 40% Cropped images (Exp. 2)

5.3.2.6.1 Output

The rendered output video is attached in the appendix as-:
Appendix\5.3 Experiment 2\5.3.2 Cropping with head detection\5.2.3.6 40%_cropping\5.2.3.6.1 rendered_video

As you can see, there is a significant drop in the quality of 3D reconstruction obtained from this experiment as compared to that in earlier cropped images and un-cropped images.

5.3.2.6.2 Metrics

PSNR, SSIM Metrics

Metric	Value
PSNR	13.040422439575195
SSIM	0.37912800908088684

Table 10. Evaluation Metrics - 40% Cropped (Exp. 2)

The metrics can also be found in the output file attached in the appendix as-:
Appendix\5.3 Experiment 2\5.3.1 Cropping without head detection\5.3.2.1 40%_cropping\5.3.2.6.2 output

We see a drop in PSNR values as compared to that in earlier cropped images and un-cropped images. This suggests a drop in similarity in synthesized images obtained after the images are cropped even more.

5.3.2.7 50% cropping

In this experiment, the images were cropped by 50% of their original dimensions. Following are the samples from this dataset-:

Figure 9. 50% Cropped Images (Exp. 2)

5.3.2.7.1 Output

The rendered output video is attached in the appendix as-:
Appendix\5.3 Experiment 2\5.3.2 Cropping with head detection\5.3.2.7 50%_cropping\5.2.3.7.1 rendered_video

As you can see, there is a slight drop in the quality of 3D reconstruction obtained from this experiment as compared to that in earlier cropped images and un-cropped images.

5.3.2.7.2 Metrics

PSNR, SSIM Metrics

Metric	Value
PSNR	13.254485130310059
SSIM	0.40191778540611267

Table 11. Evaluation Metrics - 50% Cropped (Exp. 2)

The metrics can also be found in the output file attached in the appendix as-:
Appendix\5.3 Experiment 2\5.3.1 Cropping without head detection\5.3.2.7 50%_cropping\5.3.2.7.2 output

We see a drop in PSNR values as compared to that in earlier cropped images and un-cropped images. This suggests a drop in similarity in synthesized images after the images are cropped even more.

5.3.2.8 60% cropping

In this experiment, the images were cropped by 60% of their original dimensions. Following are the samples from this dataset-:

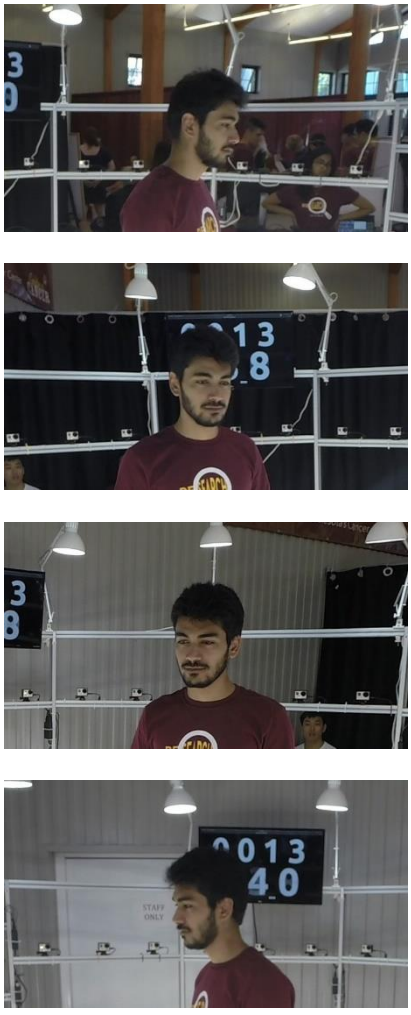


Figure 10. 60% Cropped Images (Exp. 2)

5.3.2.8.1 Output

The rendered output video is attached in the appendix as-:
Appendix\5.3 Experiment 2\5.3.2 Cropping with head detection\5.3.2.8 60%_cropping\5.2.3.8.1 rendered_video

As you can see, there is a slight drop in the quality of 3D reconstruction obtained from this experiment as compared to that in earlier cropped images and un-cropped images.

5.3.2.8.2 Metrics

PSNR, SSIM Metrics

Metric	Value
PSNR	12.829471588134766

SSIM	0.37636062502861023

Table 12. Evaluation Metrics - 60% cropped (Exp. 2)

The metrics can also be found in the output file attached in the appendix as-:
Appendix\5.3 Experiment 2\5.3.1 Cropping without head detection\5.3.2.8 60%_cropping\5.3.2.8.2 output

We see a drop in PSNR values as compared to that in earlier cropped images and un-cropped images. This suggests, as in previous case, a drop in similarity in synthesized images after the images are cropped even more.

5.4 Experiment 3: Cropping images with keeping principal points fixed

5.4.1 Method/Logic

Images were cropped in varying proportions but here the principal points or the center points were kept the same as in un-cropped full body images. Different cropping experiments were performed depending upon the proportion of cropping. The logic followed for each cropping experiment is as follows-:
Step 1: Calculate the shape (height and width) of the original image.
Step 2: Calculate the shape (height and width) of the image after reducing it by a certain percentage, let's say x%. For e.g. If the image shape is (600, 400) where height = 600 and width = 400, it is reduced by let's say 10% to (540, 360).
Step 3: Calculate the offset after cropping.

X-offset = 600-540 = 60
Y-offset = 400-360 = 50

Step 4: After getting the new dimensions or shape of the image, you need to provide the start and end points for both the x-axis and y-axis of the image. Thie means, you need to provide-:

- start_x - (starting point on x-axis from Left)
- end_x – (ending point on x-axis from Left)
- start_y - (starting point on y-axis from Top)
- end_y - (ending point on x-axis from Top)

So,

start_x = (Original Image width - Half of X-offset)/2
end_x = (Original Image width + Half of X-offset)/2
start_y = (Original Image height - Half of Y-offset)/2
end_y = (Original Image height + Half of Y-offset)/2

Step 5: You feed these start and end points to the cropping command to achieve the cropped image.

5.4.2 Code for cropping.

The code used for cropping in this experiment is attached in the appendix as-:
Appendix\5.4 Experiment 3\5.4 Cropping.ipynb

5.4.3 Processing

The data generated after cropping is processed using the same command line code as mentioned in previous experiment. The json file transforms.json is generated after the processing. Same as in the previous experiment, we don't use the same json file for training.
We use the json file obtained after processing the full body images without cropping to maintain most of the original camera parameters of the full body images and modify the remaining ones.
Same as in the previous experiments, the parameters we change are-:

- Principal points or center points
- Camera Type (From "OPENCY_FISHEYE" to "SIMPLE PINHOLE". "OPENCV_FISHEYE" is a curved camera which we don't need as we don't want the images to be distorted for a curved camera. We keep the camera to "SIMPLE PINHOLE")

The parameters we remove are-:

- Distortion parameters which we need to specify if we are using "OPENCV_FISHEYE" camera.

The parameters we keep unchanged are-:

- Focal length of the camera
- Transform matrix

5.4.4 Code for modifying transforms.json

The code to modify the **transforms.json.***is attached in the appendix as -:*
Appendix\5.4 Experiment 3\5.4.4 Modify_json

5.4.5 Training

We use the same training command line code as used in the previous experiment. We will now see all the results for varying cropping proportions.

5.4.6 10% cropping

In this experiment, the images were cropped by 10% of their original dimensions. Following are some samples of this dataset-:

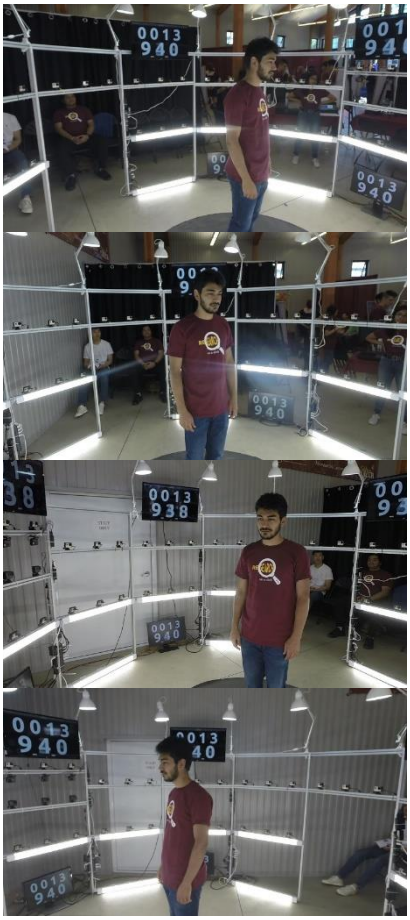


Figure 11. 10% cropped images (Exp. 3)

5.4.6.1 Output

The rendered output video is attached in the appendix as-:
Appendix\5.4 Experiment 3\5.4.1 10%_cropping\5.4.1.1 rendered_video

As you can see, there is a very slight drop in the quality of 3D reconstruction obtained from this experiment as compared to 3D reconstruction of un-cropped images. But it is important to note that the quality of reconstruction here is better than that obtained after cropping the image by the same percentage but adjusting the principal points.

5.4.6.2 Metrics

PSNR, SSIM Metrics

Metric	Value
PSNR	17.94802474975586
SSIM	2.240077495574951

Table 13. Evaluation Metrics - 10% cropped (Exp. 3)

The metrics can also be found in the output file attached in the appendix as-:
Appendix\5.4 Experiment 3\5.4.1 10%_cropping\5.4.1.2 output

5.4.7 20% cropping

In this experiment, the images were cropped by 20% of their original dimensions. Following are the samples from this dataset-:

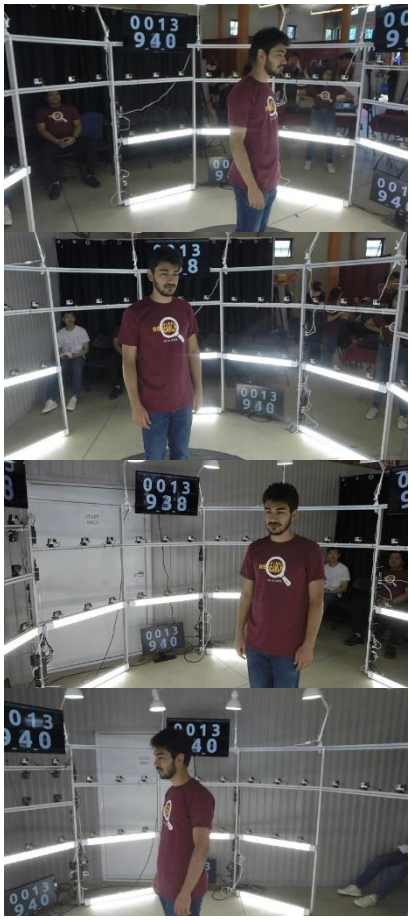


Figure 12. 20% cropped images (Exp. 3)

5.4.7.1 Output

The rendered output video is attached in the appendix as-:
Appendix\5.4 Experiment 3\5.4.2 20%_cropping\5.4.2.1 rendered_video

As you can see, there is a slight drop in the quality of 3D reconstruction obtained from this experiment as compared to 3D reconstruction of 10% cropped images and uncropped images. But it is important to note that the quality of reconstruction here is quite better than that obtained after cropping the image by the same percentage but adjusting the principal points.

5.4.7.2 Metrics

Metric	Value
PSNR	16.302303314208984
SSIM	4.014810562133789

Table 14. Evaluation Metrics - 20% cropped (Exp. 3)

The metrics can also be found in the output file attached in the appendix as-:
Appendix\5.4 Experiment 3\5.4.2 20%_cropping\5.4.2.2 output

5.4.8 30% cropping

In this experiment, the images were cropped by 30% of their original dimensions. Following are the samples from this dataset-:

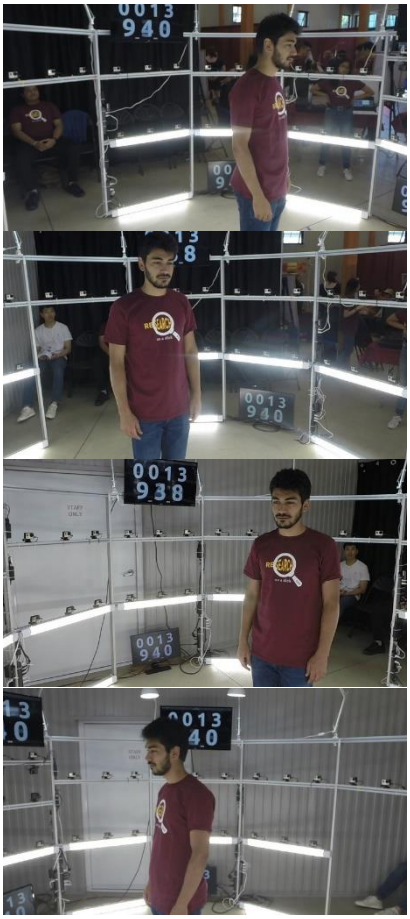


Figure 13. 30% cropped images (Exp. 3)

5.4.8.1 Output

The rendered output video is attached in the appendix as-:
Appendix\5.4 Experiment 3\5.4.3 30%_cropping\5.4.3.1 rendered_video

As you can see, there is a slight drop in the quality of 3D reconstruction obtained from this experiment as compared to 3D reconstruction of 20% cropped images, 10% cropped images and un-cropped images. But it is important to note that the quality of reconstruction here is quite better than that obtained after cropping the image by the same percentage but adjusting the principal points.

5.4.8.1.1 Metrics

Metric	Value
PSNR	17.947553634643555
SSIM	0.6437798738479614

Table 15. Evaluation Metrics - 30% Cropped (Exp. 3)

The metrics can also be found in the output file attached in the appendix as-:
Appendix\5.4 Experiment 3\5.4.3 30%_cropping\5.4.3.2 output

5.4.9 40% cropping

In this experiment, the images were cropped by 40% of their original dimensions. Following are the samples from this dataset-:

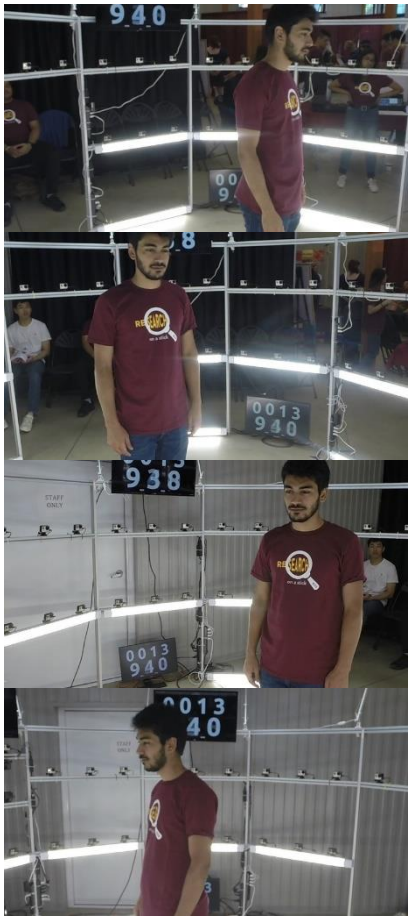


Figure 14. 40% cropped images (Exp. 3)

5.4.9.1 Output

The rendered output video is attached in the appendix as-:
Appendix\5.4 Experiment 3\5.4.4 40%_cropping\5.4.4.1 rendered_video

As you can see, there is a slight drop in the quality of 3D reconstruction obtained from this experiment as compared to 3D reconstruction of 30% cropped images, 20% cropped images, 10% cropped images and un-cropped images. But it is important to note that the quality of

reconstruction here is quite better than that obtained after cropping the image by the same percentage but adjusting the principal points.

5.4.9.1.1 Metrics

PSNR, SSIM Metrics

Metric	Value
PSNR	19.120681762695312
SSIM	1.5519497394561768

The metrics can also be found in the output file attached in the appendix as-:
Appendix\5.4 Experiment 3\5.4.4 40%_cropping\5.4.4.2 output

5.4.10 50% cropping

In this experiment, the images were cropped by 50% of their original dimensions. Following are the samples from this dataset-:

Figure 15. 50% Cropped images (Exp. 3)

5.4.10.1 Output

The rendered output video is attached in the appendix as-:
Appendix\5.4 Experiment 3\5.4.5 50%_cropping\5.4.5.1 rendered_video

As you can see, there is a slight drop in the quality of 3D reconstruction obtained from this experiment as compared to 3D reconstruction of 40% cropped images, 30% cropped images, 20% cropped images, 10% cropped images and un-cropped images. But it is important to note that the quality of reconstruction here is much better than that obtained after cropping the image by the same percentage but adjusting the principal points.

5.4.10.1.1 Metrics

PSNR, SSIM Metrics

Metric	Value
PSNR	19.554271697998047
SSIM	0.668535590171814

Table 16. Evaluation Metrics - 50% cropped (Exp. 3)

The metrics can also be found in the output file attached in the appendix as-:
Appendix\5.4 Experiment 3\5.4.5 50%_cropping\5.4.5.2 output

5.4.11 60% cropping

In this experiment, the images were cropped by 60% of their original dimensions. Following are the samples from this dataset-:

Figure 16. 60% cropped images (Exp. 3)

5.4.11.1 Output

The rendered output video is attached in the appendix as-:
Appendix\5.4 Experiment 3\5.4.6 60%_cropping\5.4.6.1 rendered_video

As you can see, there is a slight drop in the quality of 3D reconstruction obtained from this experiment as compared to 3D reconstruction of 50% cropped images, 40% cropped images, 30% cropped images, 20% cropped images, 10% cropped images and un-cropped images. But it is important to note that the quality of reconstruction here is much better than that obtained after cropping the image by the same percentage but adjusting the principal points.

5.4.11.1.1 Metrics

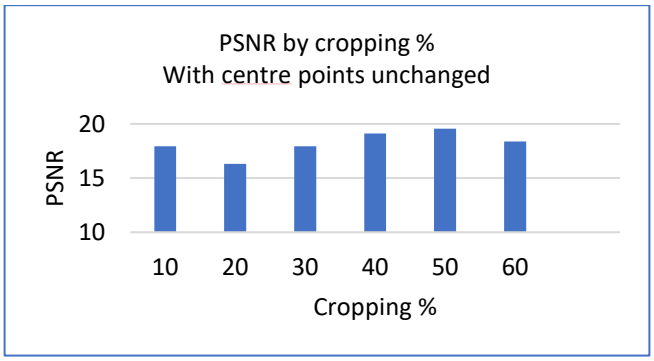
PSNR, SSIM Metrics

Metric	Value
PSNR	18.3771915435791
SSIM	0.6153876185417175

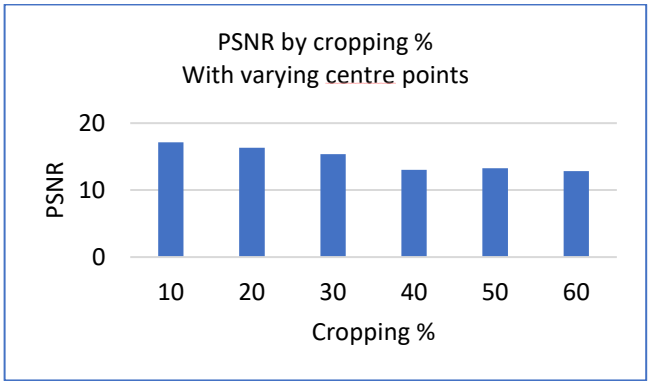
Table 17. Evaluation Metrics - 60% cropped (Exp. 3)

The metrics can also be found in the output file attached in the appendix as-:
Appendix\5.4 Experiment 3\5.4.6 60%_cropping\5.4.6.2 output

6 Conclusion



Graph 1. PSNR by Cropping% (Exp. 2)



Graph 2. PSNR by Cropping% (Exp. 3)

It is helpful to compare the results of Experiment 2 and 3. Experiment 1 was merely done to see if we are even able to obtain a 3D reconstruction from the cropped images or not. We clearly see a trend in dropping PSNR values in Experiment 2 where we were adjusting the principal points according to the cropped images. But we don't see any such trend in Experiment 3.
After performing these experiments, following conclusions were reached-:

- The model was unsuccessful in reconstructing an accurate 3D facial structure using the cropped images. The reasons may be attributed to-:
  - **Lower quality and information in cropped images** – We know that to produce a 3D reconstruction from a given set of images, there should be significant similarity among the given images to synthesize novel views. As we crop images more and more, the number of pixels in the images are reduced and the images lose resolution. We see after cropping the images up to a certain extent, the images are not good enough or the information in those images is not sufficient to find similarity among them. Hence the model is unable to produce a 3D reconstruction.
  - **Absence of enough texture on facial areas as compared to other parts of the images** – Generally we find enough texture and color in images. If we look at the HUMBI dataset, there is a lot of texture on the body of the subject due to the shape of the body and the color of the clothes. There is significant texture in the background as well due to the presence of different objects and people. But a face is generally smooth. There is not enough texture or color to find a significant number of matching or similarity points among the images. Hence this may be another reason for unsuccessful 3D reconstruction of facial images.
- Keeping the principal points or center points fixed didn't degrade the quality of 3D reconstruction as much as in the case of

adjusting the points according to the cropped images. The reason of training data variability can be attributed to this issue. In machine learning, reducing variability in the training data often leads to better generalization performance. By keeping the principal points fixed, we effectively reduce the variability introduced by changes in projection geometry caused by adjusting the principal points. This reduction in variability can help the model learn more robust features and relationships, ultimately improving the quality of the 3D reconstruction. Contrary to that, high variability in data can make the model unable to adjust the weights according to the drastic changes in the data caused by adjusting principal points.

7 Appraisal

Although I did my best in this project and covered all necessary theory and explored all the possible ways, I could get a good result given the availability of time, I would consider the following points if I were doing the whole project again-:

- **Go deep into the inner workings of NeRF model** – If I could have spent more time to understand the inner workings of NeRF model from scratch, I would probably have found a way to make the model work for the lower resolution images or cropped images. I would have tried increasing the number of parameters or layers in the neural network model which may lead to better quality of reconstruction.
- **Better Dataset** – I would choose a better dataset which is solely designed for 3D facial reconstruction. As shown in the previous sections, cropping a dataset reduces the resolution of the image and thus the model is unable to create a 3D reconstruction.

8 Future Work

I give following recommendations related to this area of work-:

1. I would suggest to deep dive into the inner workings of the NeRF model so that the model can be improved or optimized for a variety of datasets and operations such as synthesis of lower resolution or cropped images.
2. I would suggest working on how to modify the camera parameters and transformation matrix if the images are cropped. Maybe this particular study can help in improvement of 3D reconstruction of cropped images.
3. Choose an appropriate dataset specially designed for facial reconstruction so that it can help us to get further in this experiment and get even more understanding of how the 3D reconstruction of facial structures can be improved or optimized.

9 Acknowledgments

I would like to thank my project supervisor, Dr. Ludovic Magerand for his unconditional support and guidance.

10 References

Baik, Hyoung-Seon, Jai-Min Jeon and Hwa-Jin Lee. *Facial soft-tissue analysis of Korean adults with normal occlusion using a 3-dimensional laser scanner*. 2007.

Ben, Mildenhall, et al. “LLFF (Local Light Field Fusion).” (2019).

Cao, Ziyi, et al. *The algorithm of stereo vision and shape from shading based on endoscope imaging*. 2022.

Chan, Kennard Yanting, et al. “Fine Structure-Aware Sampling: A New Sampling Training Scheme for Pixel-Aligned Implicit Models in Single-View Human Reconstruction.” (2024).

Ehrhardt, Terrie Simmons, Catyana Falsetti and Anthony Falsetti. *Using Computed Tomography (CT) Data to Build 3D Resources for Forensic Craniofacial Identification*. 2021.

Gao, Kyle (Yilin), et al. “NeRF: Neural Radiance Field in 3D Vision, Introduction and Review.” (2023).

Gordon, Gaile G. “3D Pose Estimation of the Face from Video.” (1998).

Guleria, Ankit, et al. “Methods of forensic facial reconstruction and human identification: historical background, significance, and limitations.” (2023).

Horn, Berthold K. P. and Michael J. Brooks. *Shape from Shading*. 1989.

Huang, Tianxin, et al. *Learning to Measure the Point Cloud Reconstruction Loss in a Representation Space*. 2023.

Knapitsch, Arno, et al. “Tanks and temples: benchmarking large-scale scene reconstruction.” (20117).

Kwon, Soon-Hyo, et al. *Three-Dimensional Photogrammetric Study on Age-Related Facial Characteristics in Korean Females*. 2021.

Li, Bin, Shiao Zhu and Yi Lu. *A Single Stage and Single View 3D Point Cloud Reconstruction Network Based on DetNet*. 2022.

Lucas, Bruce D. and Takeo Kanade. “An Iterative Image Registration Technique with an Application to Stereo Vision (IJCAI).” (1981).
Mildenhal, Ben, et al. “NeRF-Representing Scenes as Neural Radiance Fields for View Synthesis.” 2020.
Momin, Md Sarfaraz, et al. “In-Home Older Adults’ Activity Pattern Monitoring Using Depth Sensors: A Review.” (2022).
Moschoglou, Stylianos, et al. “3DFaceGAN: Adversarial Nets for 3D Face Representation, Generation, and Translation.” (2019).
Park, Keunhong, et al. “Nerfies: Deformable Neural Radiance Fields.” (2021).
Ramírez-Hernández, Luis R, et al. “Improve three-dimensional point localization accuracy in stereo vision systems using a novel camera calibration method.” (2020).
Samavati, Taha and Mohsen Soryani. “Deep learning-based 3D reconstruction: a survey.” (2023).
Schönberger, Johannes Lutz and Jan-Michael Frahm. “Structure-from-Motion Revisited.” (2016).
Sinha, Sudipta N. “Multiview Stereo.” 2014.
Szeliski, Richard. *Computer Vision: Algorithms and Applications*. 2010.
Tancik, Matthew, et al. “Nerfstudio: A Modular Framework for Neural Radiance Field Development.” (2023).
Tous, Ruben. “Pictonaut: movie cartoonization using 3D human pose estimation and GANs.” (2023).
Waheed, Manahil, Madiha Javeed and Ahmad Jalal. “A Novel Deep Learning Model for Understanding Two-Person Interactions Using Depth Sensors.” (2021).
Wang, Shuo, et al. *3D Particle Surface Reconstruction from Multi-view 2D Images with Structure from Motion and Shape from Shading*. 2020.
Westoby, Matt, et al. “Structure-from-Motion photogrammetry: a novel, low-cost tool for geomorphological applications.” (2012).
Wilkinson, Caroline and Christopher Rynn. *Craniofacial Identification*. 2012, n.d.
Yoon, Jae Shin, et al. “Challenge, HUMBI: A Large Multiview Dataset of Human Body Expressions and Benchmark.” (2021).

11 Appendices

The appendix is attached as a zipped folder with this report. Following items are attached for each experiment-:

- Rendered video of the output
- Evaluation Metrics
- transfroms.json containing the camera parameters and transfom_matrix of each set of images
- config.yml file containing configuration details of the **nerfacto** training model