Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження підходу поля нейронного випромінювання та його застосунків
(тема)

Виконав:
студент 2 курсу, групи СШІм-21-2
Гура А. О.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)

Керівник проф. Кулішова Н. Є.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Гурі Андрію Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження підходу поля нейронного випромінювання та його застосунків

затверджена наказом університету від 31 березня 20 23 р. № 306Ст

2. Термін подання студентом роботи до екзаменаційної комісії 23 травня 20 23 р.

3. Вихідні дані до роботи Науково-технічні публікації, дані Інтернет-джерел з інформацією про проведені дослідження у галузі роботи з підходом поля нейронного випромінювання та його застосунків

4. Перелік питань, що потрібно опрацювати в роботі

1) Аналіз предметної галузі

2) Дослідження методів, алгоритмів та підходів поля нейронного випромінювання

3) Проведення експериментального дослідження

4) Аналіз результатів проведеного дослідження

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)_____

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної галузі	05.04.2023	виконано
2	Аналіз принципу роботи моделей NeRF	10.04.2023	виконано
3	Аналіз існуючих модифікацій та покращень NeRF	17.04.2023	виконано
4	Експериментальне дослідження моделей	23.04.2023	виконано
5	Написання пояснювальної записки	28.04.2023	виконано
6	Попередній захист	19.05.2023	виконано
7	Захист перед ЕК	23.05.2023	

Дата видачі завдання 3 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Кулішова Н. Є.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 77 с., 41 рис., 2 табл., 1 дод., 29 джерел.

3D МОДЕЛЮВАННЯ, ВІРТУАЛЬНА РЕАЛЬНІСТЬ, ГЛИБИННЕ НАВЧАННЯ, КОМП’ЮТЕРНИЙ ЗІР, НЕЙРОННІ МЕРЕЖІ, ОБРОБКА ЗОБРАЖЕНЬ ТА ВІДЕО, ПОЛЕ НЕЙРОННОГО ВИПРОМІНЮВАННЯ, РЕКОНСТРУКЦІЯ СЦЕН

Об'єкт дослідження – види архітектур та підходів поля нейронного випромінювання.

Предмет дослідження – існуючі архітектури та методи полей нейронного випромінювання, прикладні задачі використання полей нейронного випромінювання.

Мета роботи – дослідити види архітектур та підходів полей нейронного випромінювання, дослідити практичне використання данних моделей та перевірити їх на практиці.

Методи дослідження – аналіз існуючих підходів полей нейронного випромінювання, а також областей практичного використання даних підходів, практична реалізація програми та проведення практичного експерименту, а також обробка та виконнання аналіз отриманих результатів.

Під час виконання кваліфікаційної роботи проведений теоретичний аналіз літературних джерел щодо методів полей нейронного випромінювання та алгоритмів їх навчання, наукових публікацій щодо розробки полей нейронного випромінювання та безпосереднього їх використання для прикладних задач. Було виділено основні переваги та недоліки існуючих методів та основні області використання даних підходів. На основі найбільш обіцяючих архітектур та підходів була проведена практична робота та реалізовано декілька програмних модулей для перевірки якості моделей та демонстрації результатів.

ABSTRACT

Explanatory note: 77 p., 41 fig., 2 tabl., 1 ann., 29 sources.

3D MODELLING, COMPUTER VISION, DEEP LEARNING, IMAGE AND VIDEO PROCESSING, NEURAL NETWORKS, NEURAL RADIENCE FIELD, SCENE RECONSTRUCTION, VIRTUAL REALITY

The object of the research – types of NeRF architectures and approaches.

The subject of the research – existing NeRF architectures, approaches and areas of practical applications.

The purpose of work – research different architectures and approaches of Neural Radience Fields, investigate areas of application of these models and test them in practice.

Research methods – research of existing NeRF approaches, research of application areas of these models, program implementation and conducting of experiments, processing and analysis of the obtained results.

During the qualification work, a theoretical analysis of literary sources on the methods of neural radiation fields and their training algorithms, scientific publications on the development of neural radiation fields and their direct use for applied problems was carried out. The main advantages and disadvantages of existing methods and the main areas of use of these approaches were highlighted. Based on the most promising architectures and approaches, practical work was carried out and several software modules were implemented to check the quality of the models and demonstrate the results.

CONTENT

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

2D – two-dimensional – двовимірне;

3D – three-dimensional – тривимірне;

AI – Artificial Intelligence – штучний інтелект;

CPU – Central Processing Unit – центральний процессор;

CNN – Convolutional Neural Network – згорткова нейронна мережа;

GAN – Generative Adversarial Network – генеративна змагальна мережа;

GPU – Graphics Processing Unit – графічний процессор;

IoT – Internet of Things – інтернет речей;

ML – Machine Learning – машинне навчання;

MLP – Multi Layer Perceptron – багатошаровий персептрон;

NeRF – Neural Radiance Field – поле нейронного випромінювання;

ReLU – Rectified Linear Unit – випрямлений лінійний блок;

SfM – Structure from Motion – структура з руху.

INTRODUCTION

We live in a remarkable era where Artificial Intelligence is revolutionizing every aspect of our lives. From self-driving cars to personalized recommendations, Artificial Intelligence transformi the manner we interact with the world. One of the most exciting and promising areas of AI research is deep learning, which has led to breakthroughs in many different fields and areas including computer vision (CV), natural language processing (NLP), and robotics.

Deep learning models have shown incredible performance in tasks that were previously considered challenging or impossible. These models can learn complex representations from vast amounts of data and generalize to new unseen examples. As a result, deep learning has enabled significant progress in areas such as image recognition, object detection, and natural language understanding.

Among all the variety of different deep learning models, Neural Radiance Field (NeRF) was recently implemented and suggested as a powerful approach for modeling complex 3D scenes using deep learning. NeRF can generate photorealistic images with accurate geometry and lighting, which makes it an exciting area of research for computer graphics, virtual reality, and robotics.

Neural Radiance Field (NeRF) is full of the potential to enhance people's lives in many different ways. One significant application of NeRF is in the entertainment industry, where it can be used to create realistic virtual environments for gaming, movies, and other media. This technology can also help architects and designers visualize their designs in 3D before construction, leading to more accurate and efficient designs.

Additionally, NeRF can be used in medical imaging to create detailed and accurate representations of anatomical structures, improving diagnoses and treatment plans. Furthermore, NeRF's ability to generate realistic 3D models can help with remote training and education, such as in surgical procedures and other hands-on professions. With its versatility and potential, NeRF is poised to revolutionize various industries and contribute to making people's lives better

The relevance of NeRF lies in its potential to address long-standing challenges in computer graphics and virtual reality, such as realistic scene rendering, interactive navigation, and immersive experiences. NeRF also holds great promise for applications in robotics, such as 3D object recognition and localization.

This work aims to provide an overview of the NeRF framework, its technical aspects, applications, approaches, and future directions.

1 DOMAIN ANALYSIS AND PROBLEM STATEMENT

In today's world, we are surrounded by technology that has advanced beyond our wildest imaginations. These technological advancements have revolutionized the way we live, work, and interact with each other. From the internet to smartphones, we have the ability to reach to a vast array of tools and resources that make our lives easier and a lot more comfortable. One of the most significant advantages of this technology is automation, which has revolutionized the way we work and exist.

Thanks to all the technology we have, we can now automate a wide range of tasks and processes, making them faster, more efficient, and less error-prone. Automation has eliminated the need for many manual processes that were once time-consuming and labor-intensive. This has not only made our lives easier but has also opened up new possibilities for businesses and industries. For example, automation has enabled companies to automate their operations, make it faster and cheaper, and increase the quality of their goods and services.

Furthermore, technology has also allowed us to solve many problems that were once considered unsolvable. From healthcare to education, technology has revolutionized the way we approach complex problems and find solutions. For instance, with the help of ML (Machine Learning) algorithms and artificial intelligence, healthcare professionals can now diagnose and treat diseases more accurately and efficiently. Similarly, technology has transformed education, making it more accessible and engaging for students around the world.

Moreover, technology has made it possible to connect people from all corners of the globe, breaking down geographic barriers and enabling us to collaborate and exchange ideas on a global scale. This has led to the emergence of new technologies and innovations that are transforming the way we live, work, and interact with each other. As we continue pushing the boundaries of what is possible with technology, we can expect to see even more exciting developments that will shape our future in unimaginable ways.

The fields of Data Science and Machine Learning have played a significant role in these advancements. With Data Science, we can extract insights and knowledge from vast amounts of data. Machine Learning algorithms and techniques have enabled us to develop systems that can learn using all the data we have and make predictions or decisions based on that data. These technologies have eliminated the need for many manual processes, making them faster and more efficient.

Technology has had a profound impact on our lives, enabling us to automate tasks, solve previously unsolvable problems, and connect with people from all over the world. As we continue to develop the technology, we can expect to see even more exciting developments that will shape our future in unimaginable ways. The fields of Data Science and Machine Learning are at the forefront of this technological revolution, and we can expect to see continued advancements that will further improve our lives and transform the world we live in.

1.1 Data Science and Machine Learning

Machine Learning (ML) and Data Science are two closely related fields that have gained huge popularity among scientists in the latest years due to their potential to extract insights and value from vast amounts of data. Machine Learning is a subaread of artificial intelligence (AI) that enables models (algorithms) to learn from data, identify patterns [1], and make decisions without being explicitly programmed. On the other hand, Data Science is an interdisciplinary field that involves using statistical, mathematical, and computational methods to extract insights from data.

The popularity of Machine Learning and Data Science can be attributed to the vast amounts of data generated by organizations and individuals every day. With the appearance and development of digital technologies and the Internet of Things (IoT), data is being generated at an unbelivably fast rate. Machine Learning and Data Science offer powerful tools to analyze this data, gain insights,

and make informed decisions. From predicting customer behavior to optimizing business operations and developing personalized medical treatments, ML and Data Science are transforming industries across the board.

Machine Learning and Data Science have the potential to improve our lives in many ways. For instance, in healthcare, ML and Data Science can be used to develop personalized treatments and predict diseases' outbreak, thus improving patient outcomes and reducing healthcare costs. In the financial sector, ML and Data Science can be used to identify fraudulent activities, predict stock prices, and make better investment decisions. In the transportation industry, ML and Data Science can be used to optimize traffic flow, improve safety, and reduce carbon emissions.

The relevance of Machine Learning and Data Science now is because they are rapidly evolving, and organizations that do not embrace them risk being left behind. With the increasing volume and complexity of data being generated, the demand for skilled ML and Data Science professionals is on the rise. Organizations that can harness the power of ML and Data Science can gain a competitive advantage, drive innovation, and make better decisions. In summary, Machine Learning and Data Science are highly relevant today because they offer powerful tools to unlock insights from data, improve decision-making, and transform industries.

1.2 Deep learning and Computer Vision

Deep Learning is a subarea of Machine Learning (fig. 1.1), and it focuses mostly on building artificial neural networks that can use large datasets to learn from them. These neural networks are composed of multiple layers of interconnected nodes that can recognize patterns in data [2], make decisions, and improve their accuracy over time. In addition deep learning has revolutionized and improved the field of Computer Vision by enabling machines to interpret and understand visual data, such as images and videos.

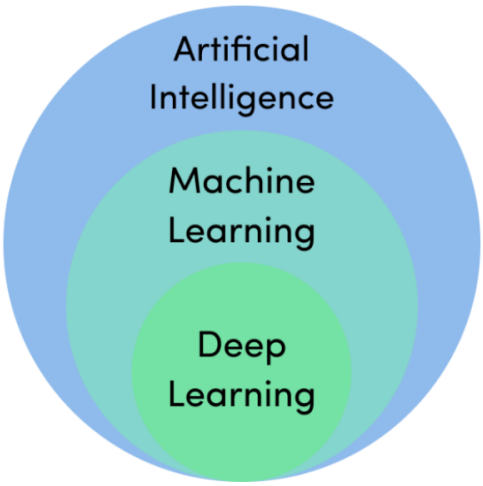


Figure 1.1 – AI, Machine Learning and Deep Learning hierarchy

Computer Vision is the field of AI that focuses on enabling machines to interpret and analyze visual data. With Deep Learning algorithms, Computer Vision has made tremendous progress in recent years, making it possible for machines and models to recognize objects, faces, and scenes with incredible accuracy. Some examples of applications of Computer Vision and Deep Learning include self-driving cars, facial recognition systems [3], and medical image analysis.

Self-driving cars (fig. 1.2) are a prime example of the power of Computer Vision and Deep Learning. These cars use cameras and sensors to gather data about their environment and use Deep Learning algorithms to interpret this data and make decisions about how to navigate. Facial recognition systems use Deep Learning algorithms to analyze images and videos and identify specific individuals. These systems are used in security and law enforcement to identify suspects and prevent crimes.

Figure 1.2 – Self-driving cars

Medical image analysis is another area where Computer Vision and Deep Learning are making a significant impact. These technologies are being used to analyze medical images such as X-rays, MRIs, and CT scans to identify diseases and develop personalized treatments. In addition, Computer Vision and Deep Learning are being used to analyze biological images, such as those produced by microscopes, to gain insights into cellular processes and improve drug discovery.

One of the key factors that have made these technologies possible today is the availability of vast amounts of data and powerful computing resources. With the rise of informational technologies and the Internet of Things (IoT), data is being generated at the super fast rate. At the same time, advances in the accelerated computing power, such as the availability of Graphics Processing Units (GPUs), have made it possible to process and analyze this data as fast as it has never been possible before.

In summary, Deep Learning and Computer Vision are two fields that have made remarkable progress in recent years, enabling machines to interpret and understand visual data. Applications of these technologies include self-driving cars, facial recognition systems, and medical image analysis. The availability of

vast amounts of data and powerful computing resources has made these technologies possible today, and they are expected to continue transforming industries across the board.

1.3 3D Scene Reconstruction and Modelling

3D Scene Reconstruction and Modeling is the process of generating a three-dimensional model of a scene or object from a set of 2D images or videos. The goal of 3D scene reconstruction is to create a virtual 3D representation of the real world (fig. 1.3), which can be used for different goals and purposes such as computer graphics, augmented reality, virtual reality, and robotics.

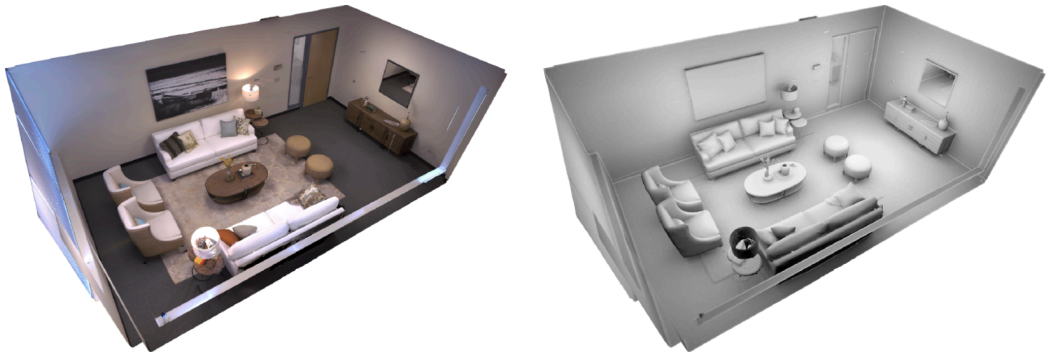


Figure 1.3 – 3D representation of a room

There are several AI approaches that can be utilized in order to solve the problem of 3D Scene Reconstruction and Modeling. One of the possible approach is to use Deep Learning techniques, such as Convolutional Neural Networks (CNNs) and Generative Adversarial Networks (GANs) [4], as they are able to learn a mapping between 2D images and their corresponding 3D models [5]. These techniques can be trained on large datasets of images and 3D models to learn to recognize and reconstruct different types of scenes and objects.

Another possible way to go is to use Structure from Motion (SfM) [6] algorithms (fig. 1.4), which use geometric constraints to evaluate the 3D structure

of an object from a set of 2D images. SfM algorithms can be combined with bundle adjustment techniques to refine the estimated 3D structure and improve the accuracy of the reconstructed scene.

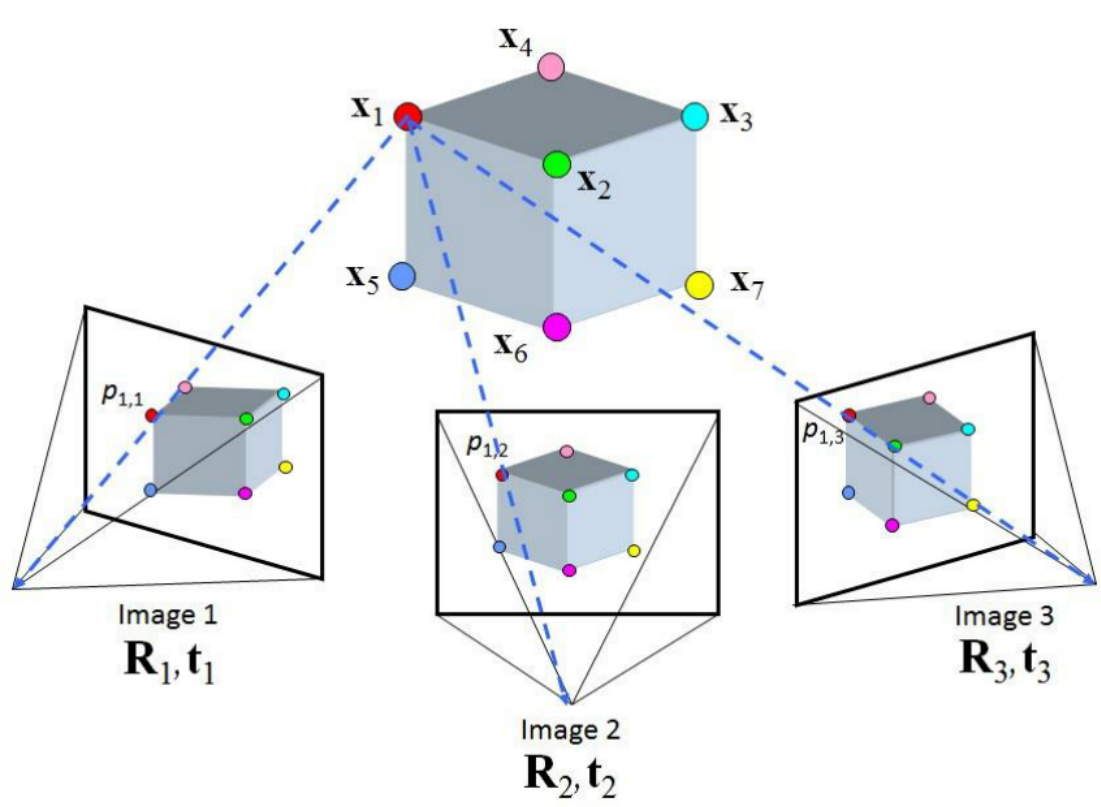


Figure 1.4 – process illustration of Structure from Motion (SfM)

Finally, there are also hybrid approaches that combine Deep Learning techniques with traditional computer vision algorithms such as SfM. These hybrid approaches can leverage the strengths of both techniques to improve the accuracy and robustness of the 3D scene reconstruction.

In the recent years another approach for this task has been invented – NeRF, or Neural Radience Fields [7]. After it was invented it gained a lot of popularity for solving this type of tasks (fig. 1.5). This technology is the main interest of this work so it will be researched more precisely later on.

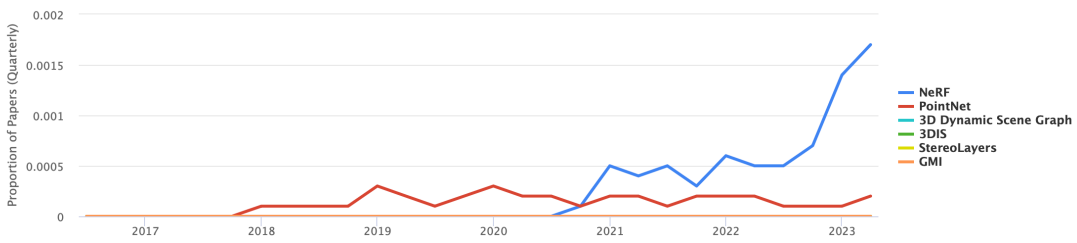


Figure 1.5 – Usage over time of different approaches for 3D representation tasks

The applications of 3D Scene Reconstruction and Modeling are numerous. In computer graphics, 3D scene reconstruction is used to create realistic 3D models of real-world scenes and objects for use in movies, video games, and virtual reality experiences. In robotics, 3D scene reconstruction is used to enable robots to navigate and interact with their environment. In medicine, 3D scene reconstruction is used to create virtual models of organs and tissues for use in surgical planning and medical training.

1.4 Problem statement

The main topic of research of the current work is NeRF technology and its different approaches as well as whole variety of areas that it can be applied to.

In other words the main goal or objective of this work is research what are Neural Radience Fields as a technology, how does it work under the hood, what are different types of this technology and how it can be applied in the real world, what are the applications of NeRF in the real world.

For achieving these types of goals the following tasks were set:

– deeply research NeRF as a technology and understand its principles and concepts;

– research different types and modifications of NeRF technology and compare them to each other;

– investigate areas of application of NeRF technology and understand what real world tasks can it solve;

– choose one of the tasks and use NeRF in practice to solve this task, analyze results and present conclusions.

By following these tasks, one can gain a comprehensive understanding of NeRF, explore its variations, identify suitable applications, and gain hands-on experience with the technology.

This approach is the best in order to critically analyze NeRF's performance, contribute to the existing body of knowledge, and potentially uncover new possibilities for its application.

2 ANALYSIS AND DESCRIPTION OF NERF

NeRF (Neural Radiance Fields) is a 3D scene representation method based on deep learning that can generate highly photorealistic images and animations of complex 3D scenes from a set of input images or a video. It was introduced by Mildenhall et al. in 2020 within the article named NeRF – representation of scenes as Neural Radiance Fields used for synthesizing the view.

A state-of-the-art method is presented for synthesizing novel views of complex scenes. The underlying continuous volumetric scene function is optimized using a sparse set of input views. The scene is represented by a fully-connected (non-convolutional) deep network, which takes a single continuous 5D coordinate that consists of thre spatial coordinates (x, y, z) and direction of the viewing (θ, φ) as input, and outputs the volume density and view-dependent released radiance at that spatial coordinates as a result.

Views are synthesized by using 5D coordinates along camera rays, and classic volume rendering techniques are used to make projection of the output colors and densities into an image. A bunch of pictures with known camera poses is the only input required to optimize the representation, as volume rendering is naturally differentiable.

The traditional approach to generating 3D representations of a scene is to use geometric methods such as point clouds or meshes, which can be time-consuming and require significant manual effort to create. In contrast, NeRF can automatically learn a continuous 3D representation of a scene from a set of input images using a neural network.

The core idea behind NeRF is to model a 3D scene as a continuous function that maps any 3D point in the scene to its corresponding color and opacity (known as «radiance»). The function is represented as a neural network, which is trained on a large set of input images using a rendering loss that compares the output of the network to the ground-truth images.

During training, the network learns to predict the radiance values of a scene

for any 3D point by encoding the scene's geometry and appearance into the network's weights. This allows the network to accurately synthesize novel views of the scene from any viewpoint, even if the viewpoint was not seen during training.

The training data for NeRF typically consists of a set of input images captured from different viewpoints around the scene, along with the corresponding camera parameters and depth maps. The depth maps provide information about the 3D structure of the scene, which is used to help train the network to model the scene's geometry.

Once the network is trained, it can be used to render novel views of the scene from any viewpoint, which makes it useful and applicable for a wide variety of differnet applications, including virtual reality, video games, and special effects in movies.

Overall, NeRF is a powerful approach to 3D scene representation that can generate highly realistic images and animations of complex scenes, and has the potential and ability to completely change the way we create and interact with 3D content.

To summarize, the technical contributions of this approach are as follows:

– approach for representing scenes with complicated geometrical properties and materials as 5D neural radiance fields, with the help of simple MLP networks as parameters;

– differentiable rendering process based on classical volume rendering techniques that uses RGB images to optimize these representations and uses a hierarchical approach for sampling to allocate the capacity of the MLP to visible scene content;

– positional encoding that maps each input 5D coordinate to a higher-dimensional space, thus making it possible to successfuly optimize the neural radiance fields to represent high-frequency scene content.

The resulting NeRF approach is shown to beat state-of-the-art view synthesis and scene recreation methods, including even those that use neural 3D

representations to scenes and train deep convolutional networks to predict sampled representations.

2.1 General oreview of NeRF

A continuous 5D function is used to represent a static scene that emits radiance in every direction (θ,φ) at every coordinates (x,y,z) in space, along with a density that regulates how much radiance is collected by a ray passing through (x, y, z). This approach trains a deep fully-connected neural network (MLP) [8] to model this function, without any convolutional layers, by regressing from a single 5D coordinate (x,y,z,θ,φ) to a single volume density and RGB color that is dependent on the viewing direction. In order to generate the neural radiance field (NeRF) from a certain point of view the first thing is to generate a set of 3D points with the help of marching camera rays through the scene. Next those points and their corresponding 2D viewing directions are utilized as and input to the neural network in order to produce an output set of colors and densities. Finally a classical volume rendering methods are used to combine both those colors and densities into a 2D image.

This entire process is differentiable, that makes it possible to use gradient descent so that optimizing the model by minimizing the discrepancy between each observed image is possible and its corresponding view can be rendered from the representation.

By reducing this error across multiple views, the network is able to learn to create a consistent representation of the scene by assigning accurate colors and high volume densities to the coordinates that have the actual scene content. An overview of the entire pipeline can be seen in figure 2.1.

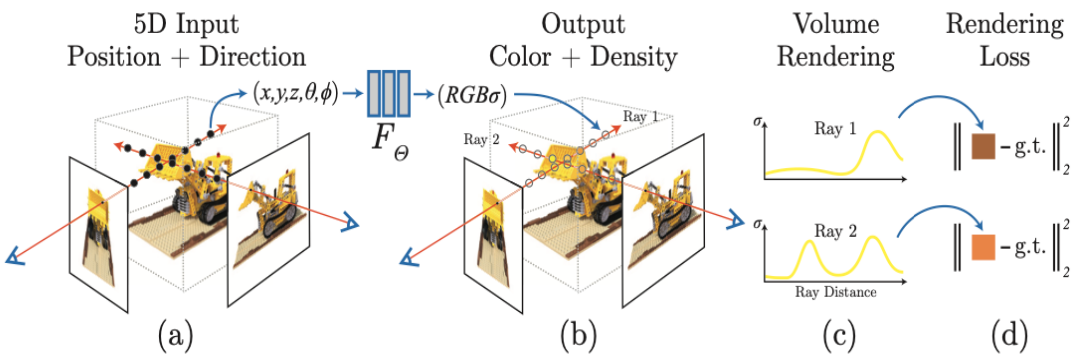


Figure 2.1 – Oreview of NeRF pipeline

An overview of the NeRF scene representation and differentiable rendering procedure is provided on the image above. Images are synthesized by sampling 5D coordinates that include coordinates and viewing direction along camera rays as shown on the left-most picture.

Those coordinates are then used as input into an MLP to generate a color and density of the volume. Volume rendering techniques are employed to combine these values into an image (c).

The rendering function possesses differentiability, what makes it possible to optimize the scene representation by minimizing the residual between synthesized images and observed ground truth images as it shown on segment (d).

It is found that the fundamental method of optimizing a neural radiance field representation for a complicated scene is not that good for creating a highly-detailed representation and it requeres an huge amount of samples per camera ray.

To deal with these complications, it is suggested to use a positional encoding technique that modifies the input 5D coordinates and improves the ability of the MLP to represent functions with higher frequencies.

Additionally, it is strongly recommended to use a hierarchical sampling approach as it significantly helps decreasing the number of queries required for suitably sampling this high-frequency scene representation.

The advantages of this approach are derived from volumetric representations: both are capable of representing complicated and precise

geometry and appearance in the wild and are well suited for projected image-based optimization that implies gradients optimization. It should also be mentioned that this approach addresses the storage challenges associated with discretized voxel grids when modeling high-resolution complex scenes.

2.2 Neural Radiance Field Scene Representation

A continuous scene is represented as a 5D vector-valued function, with an input of 3D coordinates x = (x, y, z) and the direction of the view is 2D (θ, φ), and an output of released color c = (r, g, b) and volume density σ. Direction is expressed as a 3D Cartesian unit vector d in practice. An MLP network FΘ tries to interpolate the representation of continuous 5D scene by only using optimized weights Θ and thus performing the mapping of each input 5D location and the corresponding value of the volume density as well as directional released color.

Multiview consistency is encouraged in the representation by restricting the neural network to output predictions of the volume density σ as a function that takes as input only the coordinates x, at the same time making it possible for the RGB color c to be approximated as a function of both coordinates and direction of the viewing.

In order to deal with this, the MLP FΘ firstly does the processing of the incoming 3D location x using 8 fully-connected layers, and after that infers outputs σ and a feature vector that has dimensionality 256. The resulting vector of features is then combined with the viewing direction of the camera ray's, provided to the last extra fully-connected layer, and generates the view-dependent RGB color as outputs.

Figure 2.2 shows an example of how NeRF method represents non-Lambertian effects by utilizing the input viewing direction. The released radiance that is dependent on the viewing angle is illustrated in this visualization. This approach to neural radiance field represents RGB color as a continuous 5D function that takes into account both the spatial position x and the direction of

viewing d.

As an example, it is showed the directional distribution of colors for two spatial coordinates in the neural representation of the Ship scene. In picrues (a) and (b), it is demonstrated present the visual appearance of two stationary 3D points from two separate camera positions: first one on the side of the ship and another on the water surface (blue insets). This approach is able to predict the altering specular appearance of these two 3D points. On segment(c) it is shown how this behavior extends consistently across the entire hemisphere of viewing directions.

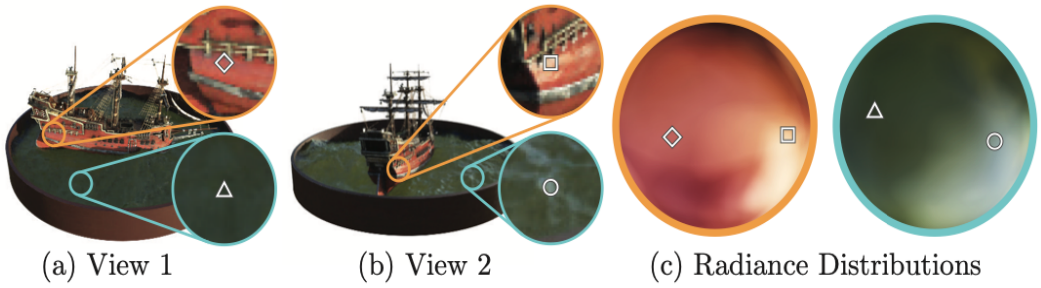


Figure 2.2 – Oreview of NeRF pipeline

Figure 2.3 aslo demonstrates that a model that was trained without view dependence but only with x as an inut fails to reoresent specularities properly.

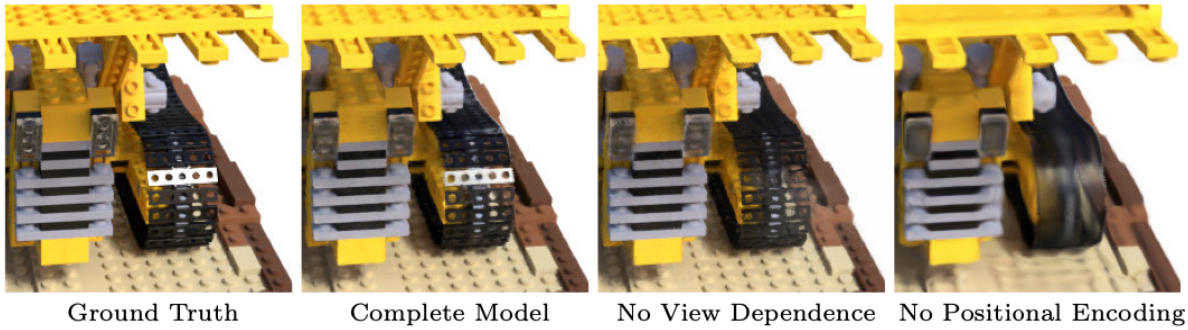


Figure 2.3 – Model predictions without different components

The benefits of incorporating view-dependent released radiance and

positional encoding into the full model can be observed through the following visualization. When view dependence is removed, the model fails to recreate the specular reflection on the bulldozer tread. When the positional encoding is removed, the model's ability to represent high-frequency geometry and texture is severely compromised, leading to an oversmoothed appearance.

2.3 Volume Rendering with Radiance Fields

The scene can be represented by 5D neural radiance field, which indicates the density and released radiance in any direction at each point in space. The color of a ray traversing the scene is generated using classical volume rendering techniques. The volume density σ(x) serves as a differential opacity factor that indicates the likelihood of a ray ending at a particle of negligible size located at x. The anticipated color C(r) of a camera ray r(t) = o + td bounded by tn and tf is:

$$C(\mathrm{r}) = \int_{t_n}^{t_f} T(t)\mathrm{c}(\mathrm{r}(t), \mathrm{d})dt, T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathrm{r}(s))\right). \quad (2.1)$$

The integral C(r) needs to be estimated to render a view from the continuous neural radiance field, which involves calculating the accumulated transmittance T(t) along the ray from tn to t. This means evaluating the probability that the ray will travel from tn to t without colliding with any other particle. The process requires tracing a camera ray through every pixel of the virtual camera that is required for rendering the view.

The continuous integral is estimated numerically using quadrature. To render a discretized voxel grid, deterministic quadrature is typically used, which limits the resolution of the representation as the MLP is only queried at a fixed set of discrete coordinates. To overcome this limitation, I use a stratified sampling method by dividing [tn, tf] into N evenly-spaced bins and drawing one sample randomly and uniformly from each bin:

$$t_i \sim \mathcal{U}\left[t_n + \frac{i-1}{N}(t_f - t_n), t_n + \frac{i}{N}(t_f - t_n)\right]. \quad (2.2)$$

The integral is estimated using a set of discrete samples; however, due to the stratified sampling, it can still be represented a continuous scene. This approach allows the MLP to be evaluated at continuous positions during optimization. The quadrature rule presented in Max's volume rendering review is used to estimate C(r) based on these samples:

$$\hat{C}(\mathrm{r}) = \sum_{i=1}^{N} T_i\left(1 - \exp\left(-\sigma_i \delta_i\right)\right)\mathrm{c}_i, \text{ where } T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right). \quad (2.3)$$

where δi = ti+1 − ti is the distance between different examples. This function can be used for calculating Cˆ(r) from the set of (ci,σi) values, and is easily differentiable and thus can be reduced to normal alpha compositing with alpha values αi = 1 − exp(−σiδi).

2.4 Main ideas and principles of NeRF in simple words

NeRF (Neural Radiance Fields) is a technology that can be used for synthesizing photorealistic 3D representations of real-world scenes. It uses a training of the neural network to predict the radiance (color and intensity) of light rays as they pass through a scene. The general concepts, ideas, and principles behind NeRF include:

Radiance fields: A radiance field is a continuous function that predicts the radiance of a point in 3D space given its position and viewing direction. NeRF learns a radiance field that models the 3D scene by training a deep neural network to predict the radiance of a point given its 3D coordinates.

Volume rendering: NeRF uses volume rendering techniques to generate images from the predicted radiance field. Volume rendering involves integrating

the radiance along the viewing rays to compute the final color and intensity of each pixel in the image.

View synthesis: NeRF can synthesize novel views of a scene by reprojecting the radiance field onto a new viewpoint and using volume rendering to generate a new image. This enables the creation of photorealistic 3D models of scenes from a limited set of 2D images.

Training data: NeRF requires a dataset of images and camera poses to train the neural network. The dataset can be created using a variety of methods, including photogrammetry, laser scanning, or by capturing images with a handheld camera.

Optimization: NeRF uses a differentiable rendering approach that allows the neural network to be optimized end-to-end using gradient-based methods. This enables the network to learn a radiance field that accurately models the scene and can generate photorealistic images (fig. 2.4).

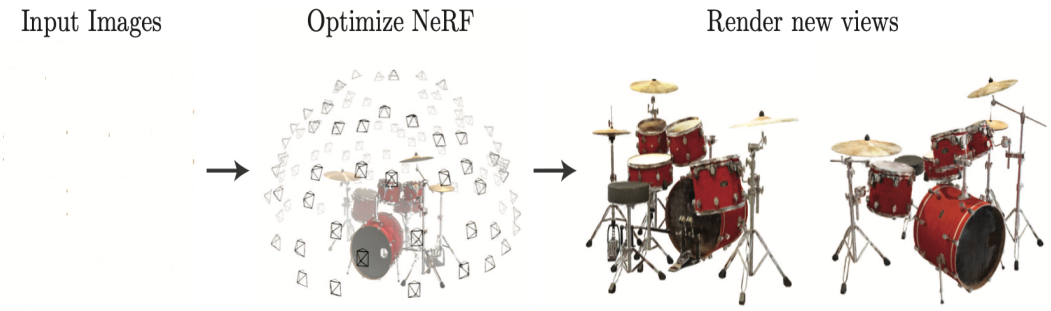


Figure 2.4 – Isutration of NeRF work algorithm

2.5 Positional encoding

Although neural networks are capable of approximating any function, it can be discovered that utilizing the network FΘ to directly operate on xyzθφ input coordinates leads to suboptimal renderings with limited capability to represent high-frequency color and geometry variation. This aligns with recent research by

Rahaman et al, which indicates that deep networks have a tendency to learn lower frequency functions. The study also demonstrates that transforming the inputs to a higher dimensional space by employing high-frequency functions before feeding them to the network allows for improved fitting of data that contains high-frequency variation.

In the domain of neural scene representations, it is capitalized on these discoveries and demonstrate that the performance of FΘ can be enhanced by re-expressing it as a combination of two functions FΘ = FΘ′ ∘ γ, where one of the functions is trained and the other is not. Experimental results (prove that this approach provides a significant boost in performance. The function γ maps R to a higher dimensional space R2L, while FΘ′ remains an ordinary MLP. In terms of formulation, the encoding function can be emplied can be expressed as follows:

$$\gamma(p) = \left(\sin\left(2^0\pi p\right), \cos\left(2^0\pi p\right), \cdots, \sin\left(2^{L-1}\pi p\right), \cos\left(2^{L-1}\pi p\right)\right). \quad (2.3)$$

To enhance the performance of neural scene representations, the concept of a positional encoding is incorpoorated, also known as a mapping function γ(·), as demonstrated in the experiments (fig. 2.3). This function is separately applied to each coordinate value of x, which is normalized to the range of [-1,1], and to the three components of the Cartesian viewing direction unit vector d, which are also within the range of [-1,1]. It is selected L = 10 for γ(x) and L = 4 for γ(d).

The positional encoding used in this method is similar to the one used in the Transformer architecture, where it is utilized to indicate the discrete positions of tokens in a sequence to an architecture that does not consider order.

However, the purpose why it is used for NeRF is different, as it uses these functions in order to perform the mapping of the continuous input coordinates into a higher-dimensional space, allowing the MLP to better approximate high-frequency functions. A similar mapping is used in concurrent research on modeling 3D protein structure from projections.

2.6 Hierarchical volume sampling

Images generation by densely evaluating the neural radiance field network at N query points along each camera ray is not very efficient because the empty free space and occluded areas that do not help with the rendering of an image are still sampled repeatedly.

To increase rendering efficiency, it is proposed a hierarchical representation inspired by early work in volume rendering. Two networks are optimized simultaneously: one «coarse» and one «fine». A set of Nc coordinates is also samplesd using stratified sampling technique, and evaluate the «coarse» network at these coordinates. The output of this «coarse» network is then used to generate informed sampling that contains more information of points along each ray. Samples are biased towards the ROI parts of the volume, which increases efficiency. To achieve this, the alpha composited color from the coarse network Cˆc(r) is rewritten as a weighted sum of all sampled colors ci along the ray:

$$\hat{C}_c(\mathbf{r}) = \sum_{i=1}^{N_c} w_i c_i, w_i = T_i\big(1 - \exp\left(-\sigma_i \delta_i\right)\big). \qquad (2.4)$$

The weights are normalized by calculating wˆ = wi/PNc wj, which generates a PDF that is piecewise-constant along the ray. Then inverse transform sampling is used in order to make a sampling of a second set of Nf coordinates from this distribution. After that, «fine» network is evaluated at the union of both sets of samples and determine the final rendered color of the ray using equation 2.4, but with all Nc+Nf samples. By doing so, it is possible to allocate more samples to regions that are likely to contain visible content.

Although this approach has a similar objective to importance sampling, it is emploied the sampled values as a nonuniform discretization of the entire integration domain instead of treating each of the samples as an independent probabilistic estimate of the integral.

2.7 Results and comparison to other methods

Experimental results are presented on two datasets of synthetic object renderings, namely «Diffuse Synthetic» and «Realistic Synthetic». The dataset that was used is DeepVoxels [9] (fig. 2.5) and it consists of four objects with Lambertian surfaces and simple geometry.

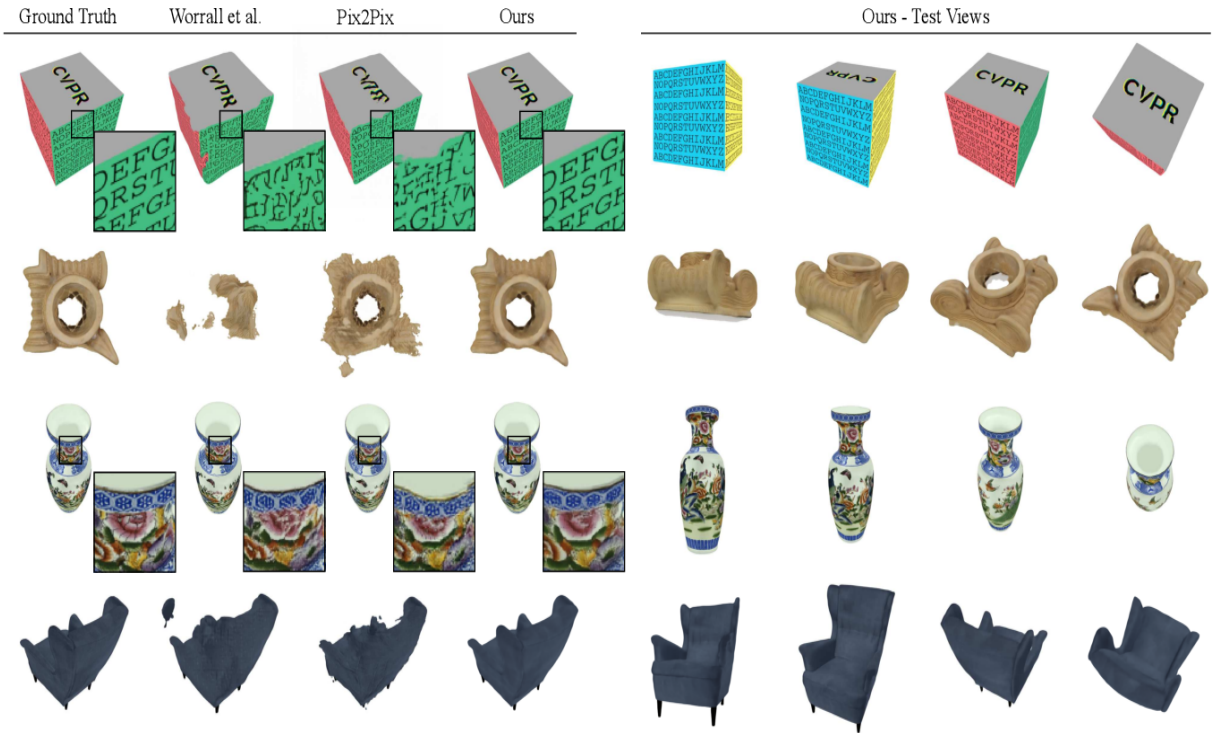


Figure 2.5 – DeepVoxels dataset example

Each object is rendered at a resolution of 512 by 512 pixels from different points of view sampled on the upper hemisphere (480 views for training and 1000 views for testing).

To further evaluate the discussed approach, another dataset was generated containing pathtraced images of eight objects with complex geometry and realistic non-Lambertian materials.

Six of these objects are generated from the points of view that are sampled on the upper hemisphere, and the remaining two are generated from the points of view sampled on a full sphere. For each scene, it was rendered 100 views as input

and 200 views for testing, all at a resolution of 800 × 800 pixels.

Quantitive results of comparison with other methods are demonstrated on figure 2.6. PSNR/SSIM [10] (higher values indicate better performance) and LPIPS [11] (lower values indicate better performance) were used to evaluate discussed method's performance on both synthetic and real image datasets.

The DeepVoxels dataset contained four simple Lambertian objects. Another dataset that was used realistic synthetic dataset included eight complex objects with non-Lambertian materials, rendered using pathtracing techniques. Eight real-world scenes were captured using handheld forward-facing cameras for the real dataset. Due to its limited ability to reconstruct objects within a bounded volume, NV could not be evaluated on this dataset.

	Diffuse Synthetic 360° [41]			Realistic Synthetic 360°			Real Forward-Facing [28]		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
SRN [42]	33.20	0.963	0.073	22.26	0.846	0.170	22.84	0.668	0.378
NV [24]	29.62	0.929	0.099	26.05	0.893	0.160	-	-	-
LLFF [28]	34.38	0.985	0.048	24.88	0.911	0.114	24.13	0.798	**0.212**
Ours	**40.15**	**0.991**	**0.023**	**31.01**	**0.947**	**0.081**	**26.50**	**0.811**	0.250

Figure 2.6 – Quantitive results of comparison

When tested on scenes from the synthetic dataset, which was generated with a physically-based renderer, discussed method shows superior ability to recover intricate details in both geometry and appearance.

For instance, it is capable of restoring Ship's rigging, Lego's gear and treads (fig. 2.7), microphone's stand and mesh of the grille, and material's reflectance (fig. 2.8).

In contrast, LLFF demonstrates the banding artifacts on the microphone stand and material's edges of the object, as well as ghosting artifacts in ship's mast and inside the Lego object. SRN, on the other hand, produces distorted and blurry renderings in all cases. Neural Volumes, meanwhile, fails to capture the

details of the Microphone's grille or Lego's gears, and it cannot recover the geometry of Ship's rigging at all.

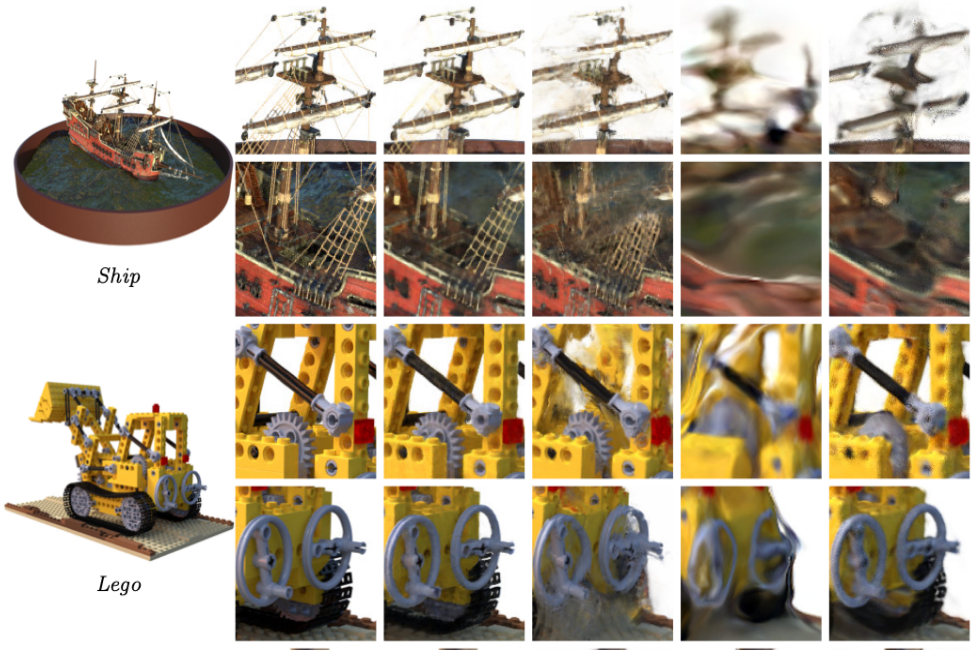


Figure 2.7 – Qualitive(image) results of comparison part 1

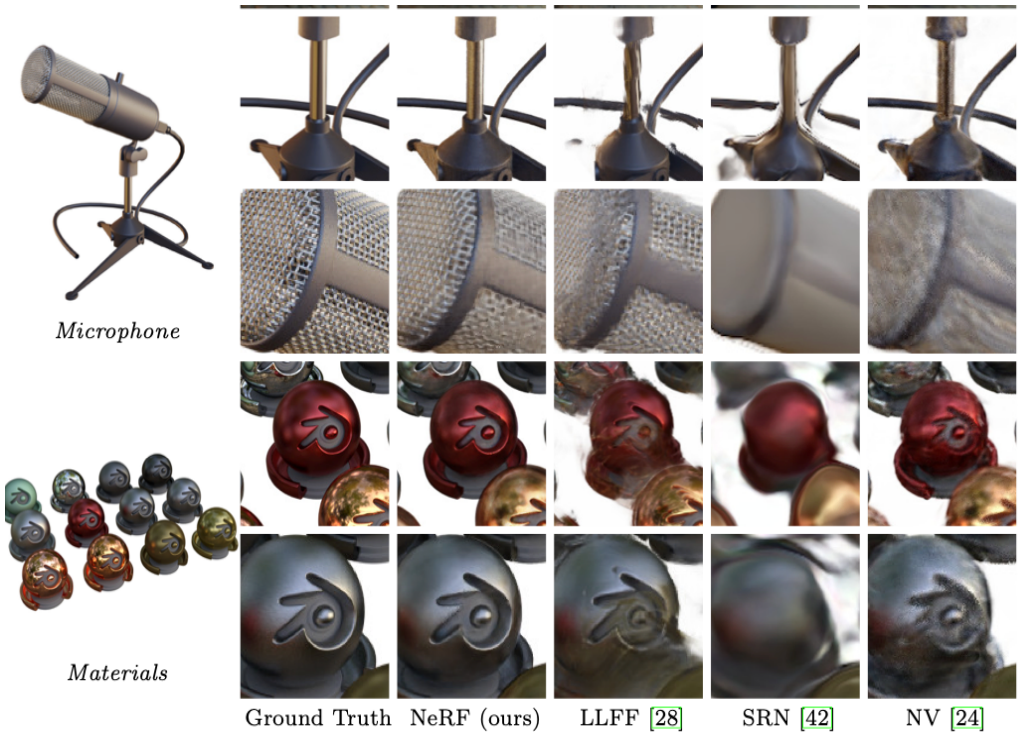


Figure 2.8 – Qualitive(image) results of comparison part 2

2.8 Areas of application of NeRF

Neural Radiance Fields (NeRF) has a range of potential applications, including:

Computer Graphics: NeRF is particularly useful in computer graphics for creating high-quality, photorealistic renderings of complex 3D scenes. By modeling a scene as a continuous function that maps 5D coordinates to color and density values, NeRF can generate images that exhibit fine-grained details, complex lighting effects, and realistic reflections and shadows.

Virtual and Augmented Reality: NeRF can be used in virtual and augmented reality applications to create realistic 3D environments that users can explore and interact with. By using NeRF to generate high-quality renderings of virtual environments, these applications can provide users with a more immersive and engaging experience.

Robotics and Autonomous Systems: NeRF [12] can also be used in robotics and autonomous systems to help robots and other machines navigate and interact with the physical world. By modeling the environment as a continuous function that maps 3D coordinates to density and color values, NeRF can enable robots to perceive and reason about their surroundings more accurately and efficiently.

Medical Imaging: NeRF can be applied in medical imaging to create detailed and accurate 3D models of organs, tissues, and other biological structures. By using NeRF to generate high-quality renderings of medical images, doctors and researchers can gain a better understanding of the underlying structures and functions of the human body, and develop more effective treatments and therapies for various medical conditions.

Entertainment and Media: NeRF can be used in the entertainment and media industries to create immersive and engaging experiences for audiences. By using NeRF to generate high-quality renderings of 3D environments, characters, and objects, movies, TV shows, and video games can provide viewers with a more realistic and compelling visual experience.

On of the latest and the most interested application of NeRF technology was a McDonalds advertisments video (fig. 2.9) that was fully generated with the NeRF technology.

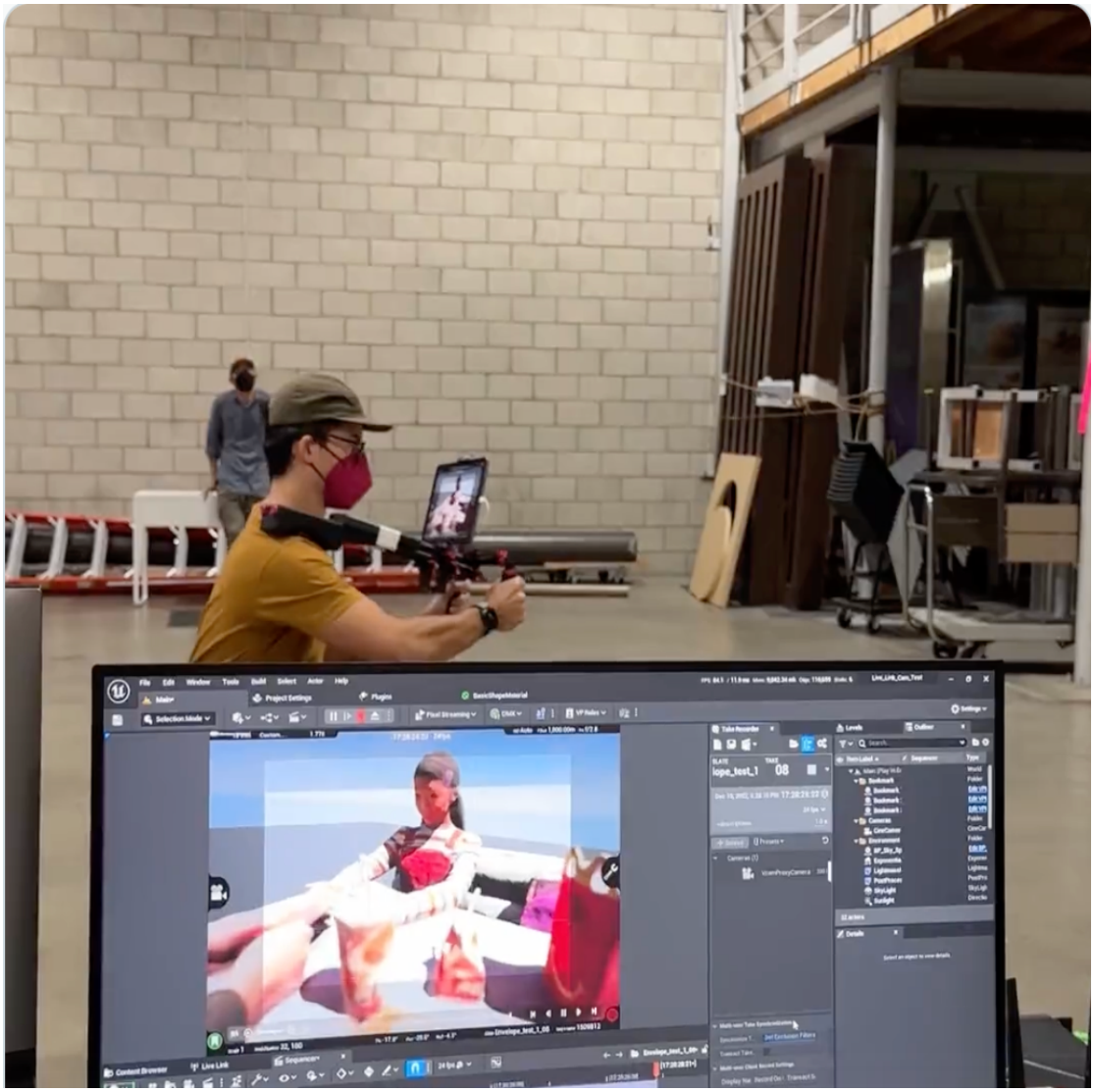


Figure 2.9 – McDonalds commercial generated with NeRF

Initialy only several scenes were filmed, and NeRF made it possible to create views from a variety of different angles like if you were there in real time walking and looking at things from different viewpoints.

3 REVIEW AND ANALYSIS OF NERF MODIFCATIONS

3.1 NeRF++ – overview

This technical report first presents an analysis of potential failure modes in NeRF, followed by an explanation of how NeRF avoids these failures in practice. Additionally, it introduces a new spatial parameterization method called the inverted sphere parameterization, which enables NeRF to work with a new class of unbounded scene captures.

The article has found that when optimizing the 5D function from a set of training images without any regularization [13], it can result in critical degenerate solutions that fail to generalize to new test views. This phenomenon is known as the shape-radiance ambiguity (fig. 3.1) and can be avoided by using a specific MLP structure, which is shown to have an impressive ability to create new views. This analysis provides a new perspective on NeRF's success. According to a theoretical analysis, when no explicit or implicit regularization is present, a set of training images can be fitted to an incorrect scene geometry (Sˆ) instead of the correct geometry (S∗) by using view-dependent radiance to mimic the correct geometry.

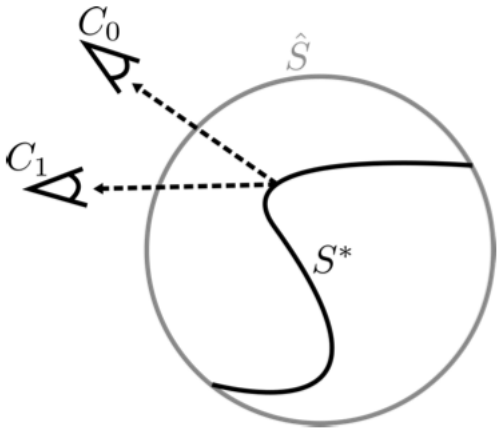


Figure 3.1 – Shape-radiance ambiguity

The report also addresses a spatial parameterization issue that arises when capturing 360◦ views around objects in unbounded environments (fig. 3.2). Using conventional parameterization methods, one of the following issues may arise – only a part of the scene is represented (indicated by the red outline), causing significant distortions in the background elements, or (2) the complete scene is represented (indicated by the orange outline), which results in a reduction in details due to the limited sampling resolution. NeRF assumes that the entire scene can fit into a bounded volume, but this can lead to either missing background elements or lacking detail due to limited sampling resolution. To solve this problem, the report proposes a simple yet effective method that models foreground and background separately using an inverted sphere parameterization. The improved results on real-world captures from the Tanks and Temples dataset and the light field dataset of Yu ̈cer et al. are presented.

In summary, this report presents an analysis of how NeRF avoids failure modes and offers a solution for parameterizing unbounded scenes in the case of 360◦ captures.

Figure 3.2 – Parameterization of unbounded scenes

3.1.1 Math behind NeRF++

An opacity field σ representing a soft shape and a radiance field c representing view-dependent surface texture are reconstructed by NeRF from

posed multi-view images of a static scene. Both σ and c are implicitly represented as MLPs (multi-layer perceptrons). The field of opacity is derived as a function of 3D position x ∈ R3, while the radiance field usingas parameters 2 values – both 3D position and viewing direction d ∈ S2. Moreover, σ(x) is used in order to refer to opacity as a position function, and c(x, d) on the other hand is used to address to radiance as a function of the direction of the view and position.

$$\min_{\sigma,\mathrm{c}} \frac{1}{n}\sum_{i=1}^{n} \left\|I_i - \hat{I}_i(\sigma, \mathrm{c})\right\|_2^2. \tag{3.1}$$

The implicit volumes σ and c are ray-traced in order to generate each pixel of I (σ, c). For a given ray r = o + td, o ∈ R3, d ∈ S2, t ∈ R+, its color is calculated with the help of integral:

$$\mathrm{C(r)} = \int_{t=0}^{\infty} \sigma(\mathrm{o}+t\mathrm{d}) \cdot \mathrm{c}(\mathrm{o}+t\mathrm{d}, \mathrm{d}) \cdot e^{-\int_{s=0}^{t} \sigma(\mathrm{o}+s\mathrm{d})ds} dt. \tag{3.2}$$

A positional encoding γ is utilized by NeRF to compensate for the network's spectral bias and synthesize sharper images. This encoding maps x and d to their Fourier features

$$\gamma^k: \mathrm{p} \rightarrow \Big(\sin(2^0\mathrm{p}), \cos(2^0\mathrm{p}), \sin(2^1\mathrm{p}), \cos(2^1\mathrm{p}), \dots, \sin(2^k\mathrm{p}), \cos(2^k\mathrm{p})\Big). \tag{3.3}$$

where k is a hyper-parameter that specifies the dimensionality of the Fourier feature vector.

3.1.2 Shape-radiance ambiguity

An inherent ambiguity exists between 3D shape and radiance in NeRF due to its ability to model view-dependent appearance. This can lead to degenerate solutions in the absence of regularization. Specifically, for any incorrect shape, it

is possible to find a family of radiance fields that perfectly fit the training images but perform poorly in novel test views.

The uncertainty between 3D object and radiance that can result in degenerate solutions, in the absence of regularization, is illustrated by using a unit sphere to represent the geometry of a given scene. The field of opacity that is used by NeRF is set to be one at the surface of the unit sphere, and zero anywhere else. Then, for each pixel in each training image examples, a ray is intersected through that pixel with the sphere, and the value of radiance at the point of intersection (and along the direction of a ray) can be defined to be the color of that pixel. This solution is to use a valid NeRF reconstruction that perfectly fits the input images. However, the ability of this solution to synthesize novel views is limited as it requires recreating an arbitrarily complicated and dependent on the view function at each point on the surface, which is unlikely to be accurately interpolated unless the density of the training views is extremely high. The shape-radiance ambiguity is depicted in figure 3.3.

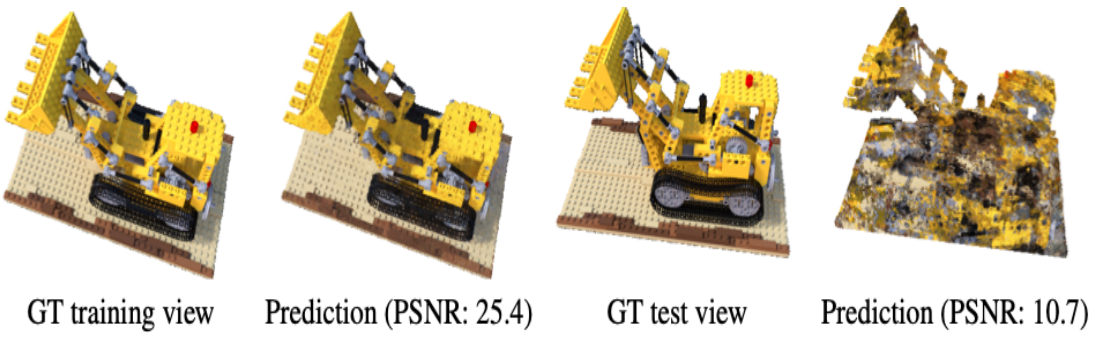


Figure 3.3– Shape-radience ambiguity demonstration

In order to demonstrate the shape-radiance ambiguity, NeRF was pretrained on a synthetic dataset in which the opacity field σ was optimized in order to make a model of a completely wrong 3D shape, while the radiance field c was optimized in a way to be able perform mapping of the training rays' intersection with the sphere and the directions of the view to their pixel color. In this example, 3 MLP

(fig. 3.4) layers were used to model the effects of view-dependence and the model was learned on 50 synthetic differenttraining samples with the points of view randomly distributed on a hemisphere. As a result the obtained incorrect solution was able to generalize the training examles pretty descently (left two images), but was unable to explain to novel test views (right two images).

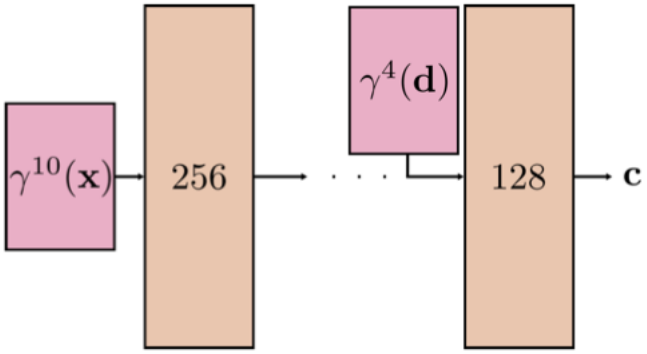


Figure 3.4 – Structure of NeRF's MLP for modeling radiance c.

3.1.3 Why NeRF avoids such degenerate solutions

It is hypothesized that NeRF is aided by two related factors. Firstly higher intrinsic complexity (i.e., much higher frequencies) is forced on the radiance field by incorrect geometry, and secondly a smooth BRDF prior on surface reflectance is implicitly encoded in NeRF's specific MLP structure.

First factor is that when the correct shape deviates from σ, the input images require c to become a high-frequency function with respect to d for reconstruction. The surface light field is generally much smoother for the correct shape, and it is constant for Lambertian materials. Representing the higher complexity required for incorrect shapes is more difficult with a limited capacity MLP.

Second factor is that a smooth surface reflectance function is implicitly favored by NeRF's specific MLP structure, where c is smooth with respect to d at any given surface point x. The MLP structure shown in figure 3.4 treats the scene position x and the viewing direction d asymmetrically, with d injected into the

network closer to the end of the MLP. This results in fewer MLP parameters and non-linear activations involved in creating view-dependent effects. The Fourier features that were used for the sake of encoding the direction of the view only consist of low-frequency components, $\gamma 4(\cdot)$ vs. $\gamma 10(\cdot)$ that are used for encoding d vs. x. That is to say, the radiance component c(x, d) has a very limited expressivity considering the d for a fixed x.

To test this hypothesis, an experiment was performed where the representation of c was changed to a simple MLP that treated x and d in the same way, namely accepting both as inputs to the first layer and encoding both with $\gamma 10(\cdot)$, in order to eliminate any implicit priors involving viewing direction that arose from the network structure. When NeRF was trained from scratch with this alternate model for c, reduced test image quality was observed compared with NeRF's special MLP, as shown in figure 3.5. This result supports the hypothesis that the regularization in the implicit way of reflectance in NeRF's MLP model of radiance c helps to recover correct solutions.

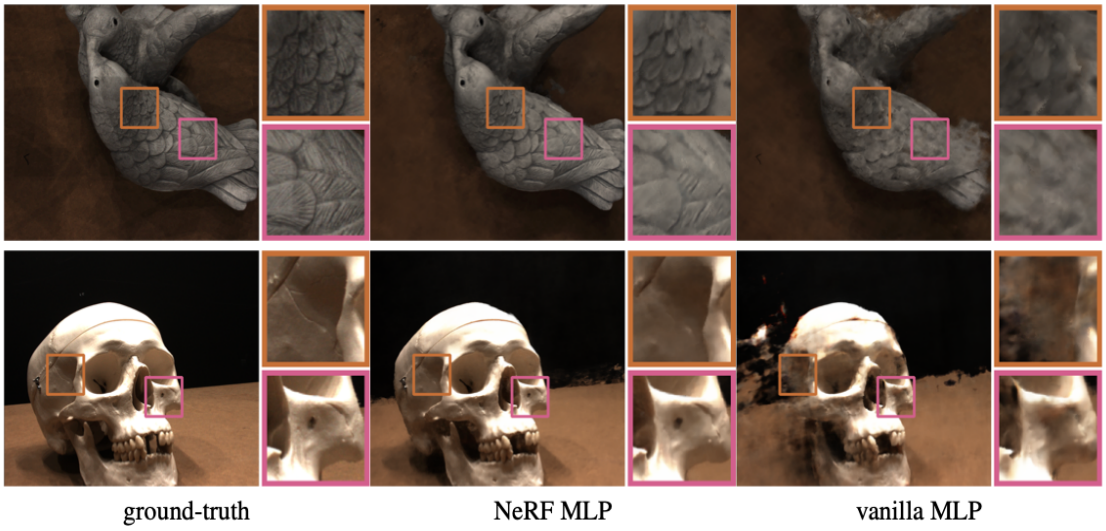


Figure 3.5 – Comparison of NeRF's MLP and vanilla MLP.

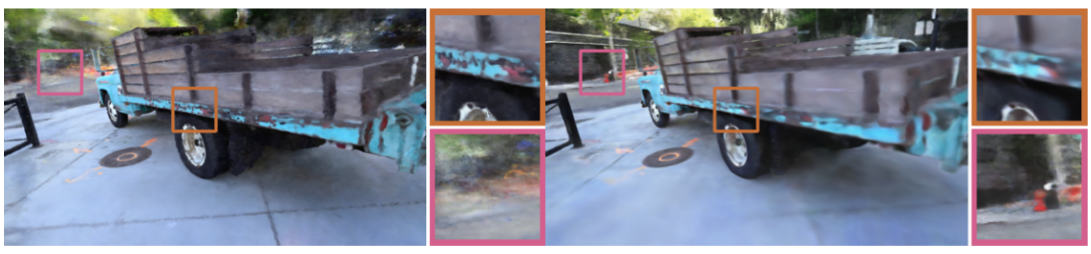
The effect of replacing NeRF's radiance field model c with a vanilla MLP (while keeping the structure of σ unchanged, both fields are trained from scratch.)

is demonstrated in this figure above. NeRF's ability to generalize to novel views is compromised by the vanilla MLP.

3.1.4 Inverted sphere parametrazation

The effect of using a finite number of samples to numerically approximate the volumetric rendering formula, which integrates over Euclidean depth, is demonstrated in figure 3.6. When the dynamic range of the true scene depth is small, the integral can be well-approximated. However, for outdoor, 360◦ captures that encompass both nearby objects and the surrounding environment, the dynamic depth range can be exceedingly large, resulting in resolution issues in NeRF's volumetric scene representation. It happens because the process of generation of the photo-realistic images requires enough resolution in both foreground and background areas. Achieving it by simply sampling points according to a Euclidean parameterization of 3D space can be challenging.

In a more strict scenario, for example if all cameras are forward-facing towards a plane, and thus separating the cameras from the content of the scene, NeRF can address this resolution issue by mapping a subset of the Euclidean space projectively, that is, the view frustum of a reference camera is converted to Normalized Device Coordinates (NDC), and integration is performed in this NDC space.



(a) bounding volume for the truck only (b) bounding volume for the entire scene

Figure 3.6 – Comparison of NeRF and NeRF++.

For 360◦ captures of unbounded scenes, either a portion of the scene is modeled by NeRF's parameterization of space leading to significant artifacts in background elements (a), or the full scene is modeled resulting in an overall loss of detail due to finite sampling resolution (b), as shown in the figure 3.6.

The restriction is addressed through an inverted sphere parameterization that enables unrestricted view synthesis. In this representation, the scene space is divided into two volumes: an inner unit sphere and an outer volume represented by an inverted sphere that covers the complement of the inner volume (as shown in figure 3.7).

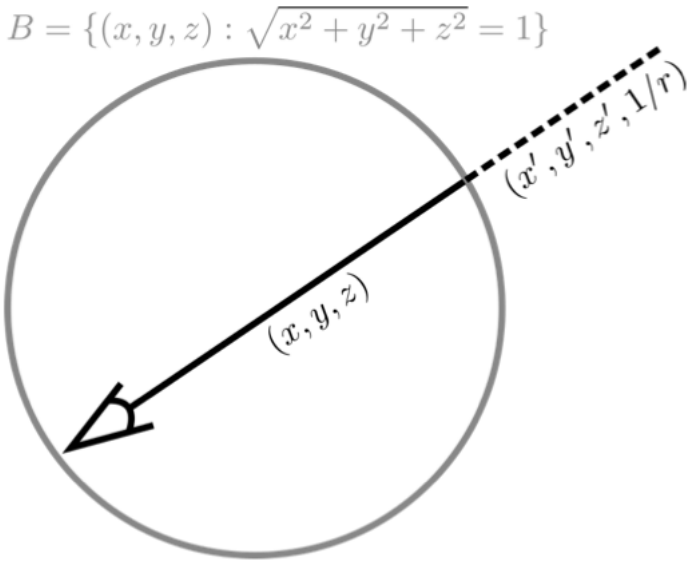


Figure 3.7 – Different parameterizations for scene contents inside and outside the unit sphere

A real-world example of a scene modeled in this manner is depicted in figure 3.8. The foreground and all the cameras are located in the inner volume, while the outer volume contains the remainder of the environment.

Two separate NeRFs are used to model the inner unit sphere and the outer inverted sphere covering the complement of the inner volume, as shown in figure 3.7. To render the color for a ray, each volume is raycasted individually, and then a final compositition is performed. There is no need to re-parameterize the inner

NeRF, as that part of the scene is already bounded. However, for the outer NeRF, an inverted sphere parametrization is applied.

(a) NeRF++ prediction (b) predicted foreground (c) predicted background

Figure 3.8 – Separating the background and foreground modelling

The modelling of foreground and background is separated in NeRF++. The predicted foreground (b) and background (c) are raycasted individually and composited to synthesize the image (a).

3.1.5 Performance comparison with original NeRF

NeRF++ was validated and compared with NeRF on two real-world datasets that were gathered with hand-held cameras: Tanks and Temples (T&T) [14] and the Light Field (LF) dataset. Quantitative metrics such as PSNR, SSIM, and LPIPS were used to measure the quality of the synthesized test images.

The training/testing images and SfM poses were used for the T&T dataset. Four large-scale scenes were captured in 360◦ using hand-held cameras, and the camera poses were estimated by COLMAP SfM. The scene was normalized for NeRF such that all cameras were inside the sphere of radius 18, ensuring that the unit sphere is covering the major part of the background content of the scene (eventhough some of the background geometry still lay outside the bounding unit sphere).

For each camera ray, points were uniformly sampled from the ray origin to its intersection with the unit sphere to numerically compute the volumetric

rendering integral. NeRF++ ray-casts both outer and inner volumes, using twice the number of samples per each camera ray in comparison to a single volume. This way the number of samples that are used by NeRF was doubled. To say it more precisely, NeRF's coarse-level MLP used 128 different uniform samples, while the fine-level MLP used 256 different additional importance samples. NeRF and NeRF++ were trained for 250 thousand iterations with a learning rate of 5e-4, and 2048 camera rays were randomly sampled at each training iteration.

PSNR, SSIM, and LPIPS were used as quantitative metrics to measure the quality of synthesized test images.

For the LF dataset, four scenes (Africa, Basket, Ship, and Torch) were selected. Each of the scenes is densely covered by two to four thousand hand made photos, with camera parameters recovered using SfM. A sparse surrounding capture was constructed by temporally subsampling the images, reducing their frequency by a factor of 40. Specifically, frames 0, 40, 80, ... were used for training, while frames 20, 60, 100, ... were used for testing. The same scene normalization method, number of samples per camera ray, batch size, training iterations, and learning rate were utilized, similar to those employed in the T&T dataset.

As demonstrated in figure 3.9, significant improvement is observed in NeRF++ compared to NeRF in challenging situations where 360◦ captures of objects within large-scale unrestricted scenes are involved.

	Truck			Train			M60			Playground		
	↓LPIPS	↑SSIM	↑PSNR	↓LPIPS	↑SSIM	↑PSNR	↓LPIPS	↑SSIM	↑PSNR	↓LPIPS	↑SSIM	↑PSNR
NeRF	0.513	0.747	20.85	0.651	0.635	16.64	0.602	0.702	16.86	0.529	0.765	21.55
NeRF++	**0.298**	**0.823**	**22.77**	**0.523**	**0.672**	**17.17**	**0.435**	**0.738**	**17.88**	**0.391**	**0.799**	**22.37**
	Africa			Basket			Torch			Ship		
	↓LPIPS	↑SSIM	↑PSNR	↓LPIPS	↑SSIM	↑PSNR	↓LPIPS	↑SSIM	↑PSNR	↓LPIPS	↑SSIM	↑PSNR
NeRF	0.217	0.894	26.16	0.377	0.805	20.83	0.347	0.811	22.81	0.372	0.801	23.24
NeRF++	**0.163**	**0.923**	**27.41**	**0.254**	**0.884**	**21.84**	**0.226**	**0.867**	**24.68**	**0.241**	**0.867**	**25.35**

Figure 3.9 – Metrics comparison of NeRF and NeRF++ on 2 different datasets

In figure 3.10, it can be observed that the fidelity of images synthesized by NeRF++ is significantly higher. In all metrics, NeRF was consistently outperformed by NeRF++ when compared on four different scenes taken from Tanks and Temples dataset: Truck, Train, M60, and Playground, as well as four other scenes taken from LF dataset: Africa, Torch, Ship, and Basket.

Figure 3.10 – Comparison of NeRF and NeRF++.

NeRF++ is qualitatively compared with NeRF on two T&T scenes (Truck, Playground) and two LF scenes (Africa, Torch). Sharper images are produced by NeRF++ compared to NeRF, and the representation of both foreground and background is better.

3.2.1 NeRF-W – oreview

Presented here is NeRF-W [15], a system developed for reconstructing 3D scenes using photo collections taken in uncontrolled settings. In the discussed approach, NeRF was extended by introducing two enhancements that are specifically designed to address the challenges of using unconstrained imagery.

Like the original NeRF, it learns a volumetric density representation Fθ using an unstructured collection of photos {Ii}Ni=1, where the camera parameters are known. NeRF operates under the assumption of consistency in the input views, which states that a point in 3D space viewed from the same position and direction in two different images will have the same intensity. However, this assumption is violated by internet photos (as shown in figure 3.11) due to two distinct phenomena.

Figure 3.11 – NeRF performance on the real-world images

The first one it the illumination of objects in outdoor photography is affected by various factors such as time of day and atmospheric conditions. Photometric inconsistencies can arise due to variations in auto-exposure settings, white balance, and tone-mapping across different photographs caused by

photographic imaging pipelines. These inconsistencies can be further amplified by these factors.

The second one is transient objects, such as moving objects or occlusions, are commonly present in real-world landmarks photography, making their reconstruction challenging. This is particularly the case for tourist photos, which may include human subjects or pedestrians posing around the landmark. To address these issues, it is proposed two model components in the approach. In the discussed approach NeRF was extended by making it possible for appearance that is dependent on image and lighting variations, which enables us to model photometric inconsistencies between images explicitly. It also was further improved by jointly estimating and disentangling the transient objects from a static representation of the 3D world. An overview of the proposed model architecture is presented in figure 3.12.

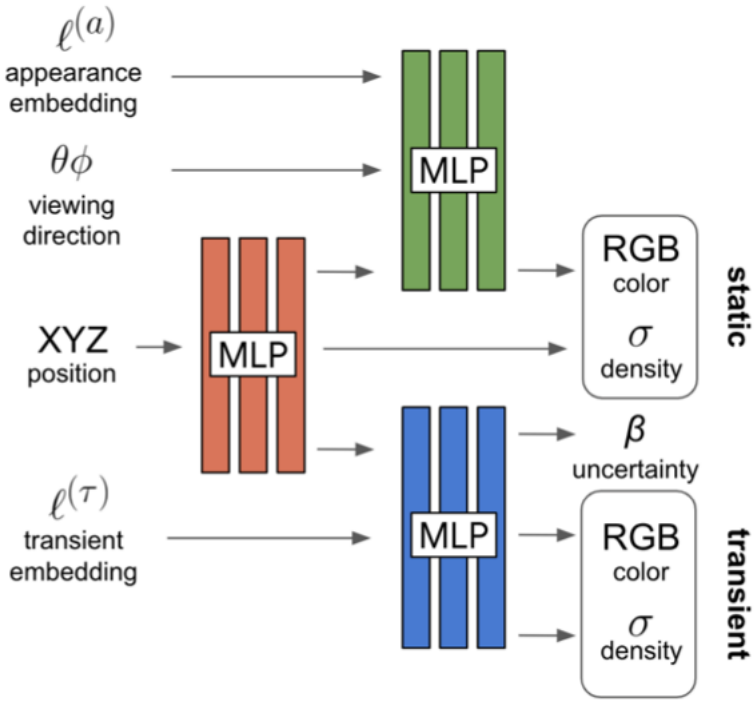


Figure 3.12 – NeRF-W model architecture

NeRF-W generates static and transient colors and densities along with an

estimate of uncertainty for a given 3D position and viewing direction, based on the learned appearance and transient embeddings. It is worth noting that the model first generates the static opacity, prior to being conditioned on the appearance embedding. This guarantees that the static geometry is consistent across all images.

3.2.2 Latent Appearance Modeling

Incorporating variable lighting and photometric post-processing into NeRF requires the implementation of Generative Latent Optimization (GLO) approach [16]. In this approach, a real-valued appearance embedding vector l(a) with a length of n(a) is assigned to each image Ii. The equation (1) is modified by replacing the image-independent radiance c(t) with a radiance ci(t) that is dependent on image, which in turn introduces a dependence on the image index I to the approximated pixel color Ci:

$$\begin{aligned} \hat{\mathrm{C}}_i(\mathrm{r}) &= \mathcal{R}(\mathrm{r}, \mathrm{c}_i, \sigma), \\ \mathrm{c}_i(t) &= \mathrm{MLP}_{\theta_2}\left(\mathrm{z}(t), \gamma_{\mathrm{d}}(\mathrm{d}), \ell_i^{(a)}\right). \end{aligned} \tag{3.4}$$

Optimization of the {l(a)}N embeddings is carried out alongside θ. The input of these appearance embeddings to only the branch of the network that releases color allows the model to have the versatility to configure the released radiance of the scene in a specific image while ensuring that the 3D geometry (previously predicted by MLPθ1) is fixed and shared across all images. A small value of n(a) is set to facilitate optimization in identifying a continuous space where conditions of lighting can be embedded, consequently enabling seamless interpolations between conditions, as presented in figure 3.13.

Figure 3.13 – Interpolations between the appearance embeddings illumination

Renderings are produced by interpolating the appearance embeddings l(a) of two training images (left, right), resulting in seamless transitions between color and illumination while maintaining constant geometry (middle). It should be noted that the renderings do not contain the individuals (left) and luminaires (right) that are present in the training images.

3.2.3 Transient Objects

Two design decisions were made to tackle transient phenomena: Firstly, the color-emitting MLP utilized in NeRF is regarded as the «static» head of the model, and an extra «transient» head is incorporated that generates its own color and density, where the density is permitted to differ across the training images. This feature empowers NeRF-W to construct images that encompass occlusions, without introducing distortions into the static scene representation.

Secondly, instead of making the assumption that all observed pixel colors hold equal reliability, the transient head is allowed to emit an uncertainty field (similar to the existing color and density fields), enabling the model to adjust its reconstruction loss to disregard unreliable pixels and 3D positions that are probable to contain occlusions. Each pixel's color is modeled as an isotropic normal distribution, and its likelihood is maximized. The variance of this distribution is «generated» using the same volume rendering method utilized by NeRF. These two model components empower NeRF-W to differentiate between

static and transient phenomena without explicit supervision.

3.2.4 Optimization

Similar to NeRF, two versions of Fθ are optimized concurrently: A fine model that adopts the model and losses explained previously, and a coarse model that solely employs the latent appearance modeling component. Together with parameters θ, it is optimized per-image appearance embeddings {l(a) }N and transient embeddings {l(τ)}N. The loss function of NeRF-W is then computed as follows:

$$\sum_{ij} L_i(\mathbf{r}_{ij}) + \frac{1}{2}\left\|\mathbf{C}(\mathbf{r}_{ij}) - \hat{\mathbf{C}}_i^c(\mathbf{r}_{ij})\right\|_2^2. \tag{3.5}$$

The set of supplementary hyperparameters for NeRF-W consists of λu, βmin, as well as embedding dimensionalities n(a) and n(τ). Given that optimization solely generates appearance embeddings {l(a)} for images in the training set, the embeddings for test-set images are not specified. To create visualizations for the test-set, it is selected l(a) that closely matches the target image (e.g. figure 3.13) or assign an arbitrary value.

3.2.5 Results and comparisons

A subset of scenes is presented in figure 3.14, showcasing qualitative results for all models and baselines. Checkerboard artifacts that are typical of 2D re-rendering methods are noticeable in NRW's renderings. NRW is also susceptible to 3D geometry errors upstream, as demonstrated by the smaller towers of the church in Prague Old Town. NeRF delivers a consistent 3D geometry, however, considerable portions of the scene contain ghosting artifacts and occlusions that are prominent in the Sacre Coeur and Prague Old Town. Renderings produced by NeRF also exhibit major global color changes comparing

it to the ground truth samples. These artifacts are a direct consequence of NeRF's static-world assumption, where NeRF tries to account for all photometric coombinations and transient occlusion using a single scene representation. This assumption not only impairs NeRF's renderings but also its underlying geometry.

Figure 3.14 – Qualitative results from the inference with the Phototourism dataset

Figure 3.14 also displays the results of the NeRF-A ablation in comparison to NeRF, indicating that NeRF-A [17] produces less hazy renderings.

However, NeRF-A is not able to reconstruct detailed features like the brickwork on Sacre Coeur's dome. Conversely, the NeRF-U ablation excels in capturing fine details but struggles to model varying photometric effects. NeRF-

W combines the advantages of both ablations, thus generating more precise and crisper renderings.

On the other hand, NeRF-W generates precise 3D reconstructions as illustrated in figure 3.15. The expected termination depth of each ray is computed to render depth maps from NeRF and NeRF-W. NeRF's geometry is affected by appearance changes and occlusions, but NeRF-W is resilient to these phenomena and generates precise 3D reconstructions.

Figure 3.15 – Depth maps from NeRF and NeRF-W

Figure 3.16 1 provides a summary of the quantitative outcomes. When optimizing NeRF on image collections taken in the wild, the results are considerably inadequate, failing to compete with NRW. Conversely, across all datasets, NeRF-W shows better results compared to the baselines by PSNR and MS-SSIM metrics. It slightly increases quality of the previous state of the art NRW by an average difference of 4.4dB in PSNR, and with up to 40% progressions in MS-SSIM.

Although NeRF-W trains by minimizing only a per-pixel squared error, it

still is able to improve considering the previous state of the art on LPIPS in three of six scenes and remains competitive in the rest. Since NeRF-W lacks a perceptual loss, it is not motivated to produce the high-frequency textures that are favored by perceptual metrics like LPIPS.

However, the results for NRW exhibit temporal instability. As the camera moves, the generated images can have features flickering and wobbling unrealistically, and this is not captured by the single-image metrics or figures presented in this paper. It is strongly recommended videos of two approaches to observe the temporal instability of NRW as compared to NeRF and NeRF-W.

	Brandenburg Gate			Sacre Coeur			Trevi Fountain			Taj Mahal			Prague			Hagia Sophia		
	PSNR	MS-SSIM	LPIPS	PSNR	MS-SSIM	LPIPS	PSNR	MS-SSIM	LPIPS	PSNR	MS-SSIM	LPIPS	PSNR	MS-SSIM	LPIPS	PSNR	MS-SSIM	LPIPS
NRW [22]	23.85	0.914	0.141	19.39	0.797	0.229	20.56	0.811	0.242	21.24	0.844	**0.201**	19.89	0.803	**0.216**	20.75	0.796	**0.231**
NeRF	21.05	0.895	0.208	17.12	0.781	0.278	17.46	0.778	0.334	15.77	0.697	0.427	15.67	0.747	0.362	16.04	0.749	0.338
NeRF-A	27.96	0.941	0.145	24.43	0.923	0.174	26.24	0.924	0.211	25.99	0.893	0.225	22.52	0.870	0.244	21.83	0.820	0.276
NeRF-U	19.49	0.921	0.174	15.99	0.826	0.223	15.03	0.795	0.277	10.23	0.778	0.373	15.03	0.787	0.315	13.74	0.706	0.376
NeRF-W	**29.08**	**0.962**	**0.110**	**25.34**	**0.939**	**0.151**	**26.58**	**0.934**	**0.189**	**26.36**	**0.904**	0.207	**22.81**	**0.879**	0.227	**22.23**	**0.849**	0.250

Figure 3.16 – Quantitative results for different NeRF versions on the Phototourism dataset

A new method called NeRF-W has been introduced in this work for reconstructing 3D scenes from unstructured internet photo collections. The approach is based on NeRF and utilizes a per-image latent embedding to capture the photometric appearance variations commonly seen in in-the-wild data. By separating the scene into shared and image-dependent components, the model can distinguish transient elements from the static scene. Results of experiments on both real-world and synthetic data show significant improvements in both qualitative and quantitative performance over previous state-of-the-art techniques.

4 DESCRIPTION OF THE PRACTICAL PART

This section of the project will detail the selected model and methodologies that have been employed in the final solution. Additionally, the reasoning behind selecting these specific approaches will be elaborated upon.

4.1 Python

Selecting a programming language was the initial step, and without hesitation, I have decided to adopt Python (as depicted in figure 4.1) as the primary language for carrying out the investigation and developing the desktop version of the implementation.

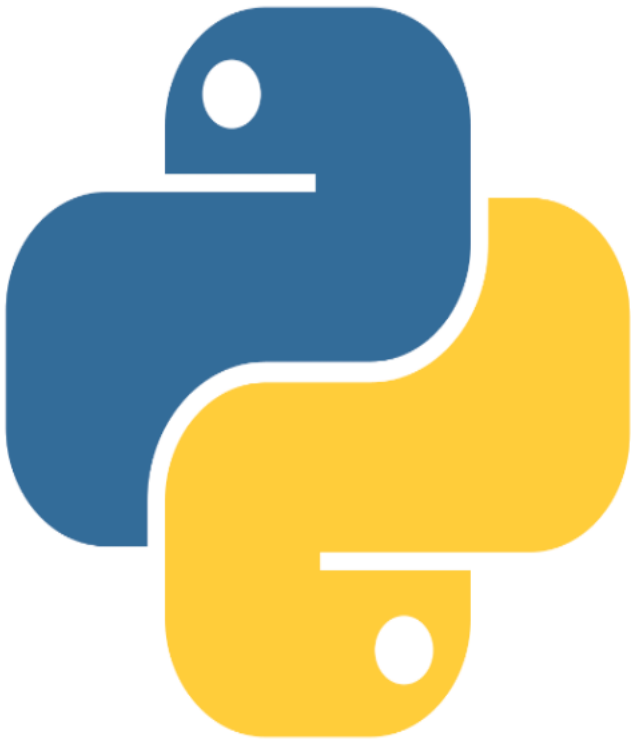


Figure 4.1 – Python programming language logo

Python was chosen as the primary programming language for its unrivaled popularity and ease of use. Its simple syntax makes it accessible to individuals

without advanced computer science skills, contributing to its widespread use among researchers from diverse fields.

The large and rapidly evolving community surrounding Python is a testament to its simplicity and versatility, with a constant influx of projects, research, and investigations. As a result, Python remains a top choice for conducting research and implementing practical solutions (fig. 4.2).

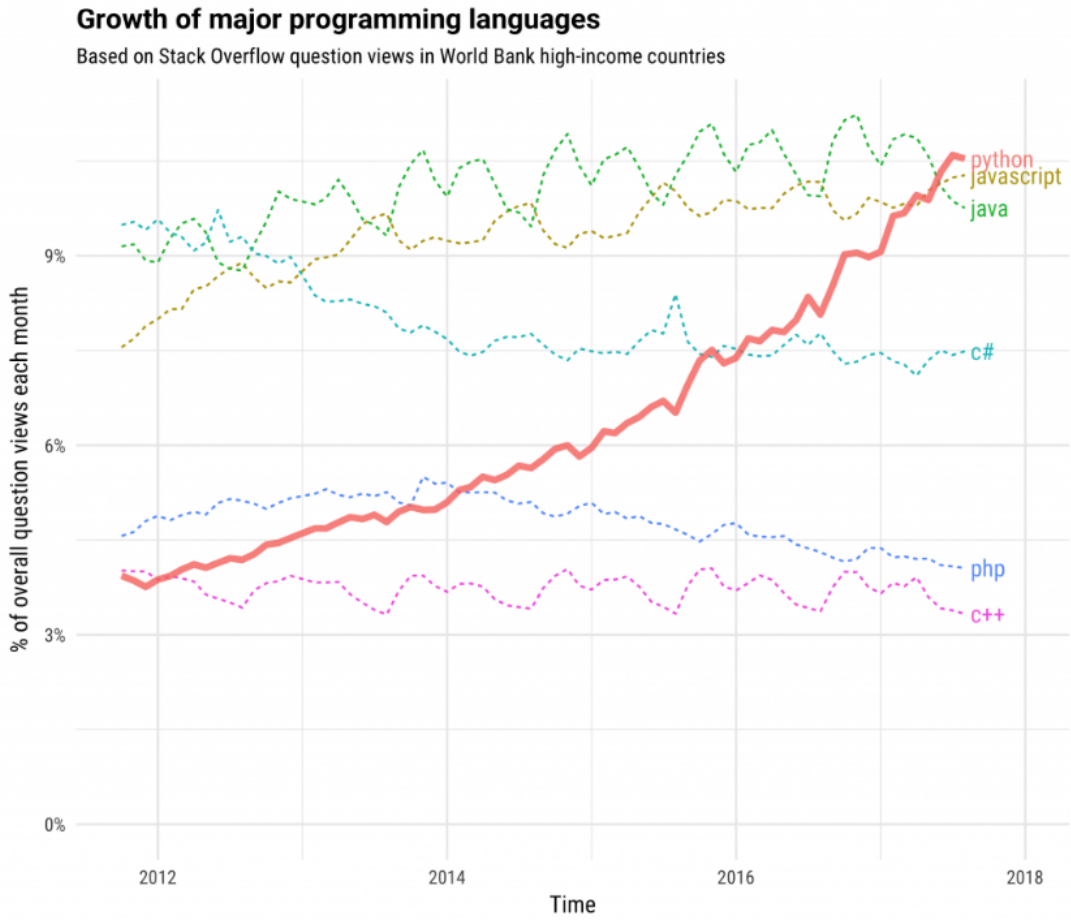


Figure 4.2 – Demonstration of continuously increasing Python popularity

Python has a significant advantage in such areas as Data Science, Machine Learning, and Artificial Intelligence. It has virtually monopolized these domains, with most new research and papers having a corresponding Python implementation.

This feature makes it incredibly convenient as you do not need to learn

complex languages to keep up with the latest state-of-the-art models, technologies, or approaches. Python's widespread use in these fields has created a situation where you can simply use it because most people are already using it, making it a convenient choice.

4.2 General frameworks

During this work a lot of different frameworks were used. I will highlight them most important ones of them and briefly describe each of them, why it was used and what it is used for.

NumPy (fig. 4.3) is a Python library that stands for «Numerical Python» [18]. It is a popular package for scientific computing in Python and makes it possible to work with the large multi-dimensional arrays and matrices, along with a wide range of mathematical functions to operate on these arrays.

Figure 4.3 – NumPy logo

One of the main advantages of NumPy is that it provides efficient numerical operations on large arrays of data, which makes it well-suited for data manipulation, numerical simulation, and scientific computing tasks. Additionally, NumPy has many built-in functions for mathematical operations like linear algebra, Fourier transform, and random number generation.

NumPy is widely used in the data science community and is a fundamental

tool in many scientific fields, including physics, biology, and engineering. Its easy-to-use interface and powerful capabilities make it an essential tool for many data-driven applications.

OpenCV (fig. 4.4) is a popular open-source library of programming functions that are mainly aimed at real-time computer vision tasks, such as image processing and analysis, object detection and recognition, and machine learning.

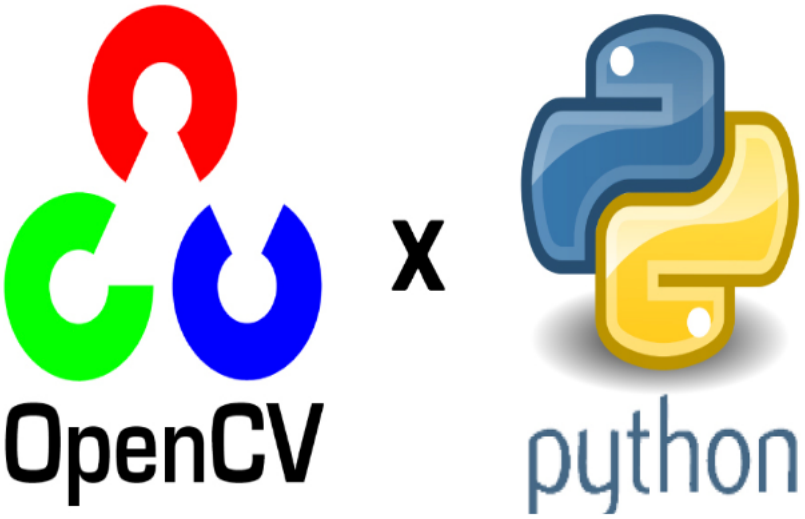


Figure 4.4 – OpenCV logo

OpenCV provides a wide range of algorithms and techniques for various tasks, including image and video input/output, filtering and transformation, feature detection and extraction, object tracking, and machine learning-based object recognition and classification. It supports many different programming languages such as C++, Python, and Java, making it accessible to a broad range of developers.

The library also offers a range of utilities and tools to help developers visualize and debug their computer vision applications, such as the ability to display images and videos, draw on them, and track events and metrics. Additionally, OpenCV has a very large and active community of supporters who contribute to the library and provide support through forums, and tutorials.

4.3 Deep Learning frameworks

PyTorch (fig. 4.5) is an open-source machine learning library that is widely used for developing various types of deep learning models [19]. It was developed by Facebook's AI Research (FAIR) team and released in October 2016. PyTorch provides a dynamic computational graph, which makes it easy to write and debug code for complex neural networks.

Figure 4.5 – PyTorch logo

One of the key benefits of PyTorch is its ease of use and flexibility. It allows for efficient processing of large amounts of data using GPUs and supports various operations such as linear algebra, signal and image processing, and neural network training.

PyTorch also provides a huge amount of pre-trained weights fo models that can be used for a bunch of different tasks, such as computer vision and natural language processing. It has a large and active community that contributes to its development and provides resources for learning and support.

Overall, PyTorch has become a popular choice among researchers and practitioners in the machine learning community due to its ease of use, flexibility, and powerful capabilities for deep learning model development.

TensorFlow (fig. 4.6) is an open-source software library developed by

Google [20] for data preparation and management tasks, as well as programming application for tasks such as machine learning, deep learning, and neural networks. It was released in 2015 and has since become one of the most famous and widely used machine learning frameworks.

Figure 4.6 – Tensorflow logo

TensorFlow is based on the concept of a computational graph, where nodes in the graph contain inside themselves different mathematical operations and edges represent the flow of data between those operations. It allows users to create complex machine learning models, including deep neural networks, using a high-level API.

TensorFlow also provides a suite of tools and libraries for data manipulation, model visualization, and deployment to various platforms, including mobile devices and the web. Additionally, TensorFlow has a large and active community that provides support and resources for users at all levels of expertise.

TensorFlow and PyTorch are two popular deep learning frameworks used for developing and training machine learning models. Here are some key differences between the two:

– ease of use: PyTorch is considered to be more user-friendly than TensorFlow. It offers a more pythonic approach to building models, making it

easier to debug and write custom code. TensorFlow, on the other hand, has a steeper learning curve and is more complex to use.

– dynamic vs static graph computation: PyTorch uses dynamic computational graphs, which means the graph is built on the fly as the code is executed. TensorFlow, on the other hand, uses static computational graphs, which means the graph must be defined before the code is executed. This makes PyTorch more flexible and easier to debug.

– performance: TensorFlow is known to have better performance than PyTorch, especially when it comes to large-scale distributed training. TensorFlow also has more mature tools for production deployment, such as TensorFlow Serving.

– community: Both TensorFlow and PyTorch have large and active communities. However, TensorFlow has been around longer and is used by more companies, which means it has a larger developer base and more resources available.

– ecosystem: TensorFlow has a more extensive ecosystem with more pre-trained models, libraries, and tools available. PyTorch has a smaller ecosystem, but it is growing quickly.

Overall, the choice between TensorFlow and PyTorch often is a question of personal preference of a user and the specific requirements of the project. PyTorch is often preferred for research and development due to its ease of use and flexibility, while TensorFlow is often used for production environments and large-scale distributed training.

4.4 Dataset description

To the best of knowledge, no standardized dataset or metrics have been put forth for assessing the performance of systems that remove static objects from videos. During this work a new RGB-D dataset comprising real-life scenes was used, specifically curated for evaluating the effectiveness of object removal

techniques. The dataset, consists of two distinct variants. These variants serve different purposes and are utilized in diverse ways for benchmarking the performance of such systems.

4.4.1 Real objects dataset

A collection of 17 scenes, consisting of real-world objects, is included in this dataset. These scenes encompass a mix of indoor and outdoor environments, with a variety of orientations, such as landscape and portrait. The focus is on a specific area within each scene that contains at least one object, with one particular object designated as the labeled object of interest. The dataset encompasses scenes with varying levels of difficulty, characterized by factors such as background texture, object size, and scene complexity.

For each scene, it was obtained two sequences: one sequence includes the object that is intended to remove, while the other sequence does not. These sequences were captured utilizing ARKit on an iPhone 12 Pro equipped with Lidar technology. They comprise RGB-D images and corresponding camera poses. The RGB images were resized to 192 × 256 pixels, matching the resolution of the depth maps. Both sequences within a scene share the same camera coordinate system. The length of the sequences varies, ranging from 194 to 693 frames.

As mentioned earlier, the masks are acquired by annotating a 3D bounding box around the object of interest and then refining them for all scenes. In each scene, the sequence containing the object along with its corresponding masks is utilized for training the NeRF model, while the sequence without the object is used for testing. The inclusion of real objects facilitates the assessment of how the systems handle actual shadows, reflections, and the synthesis of novel views. This approach allows for a more comprehensive evaluation of the system's performance in realistic scenarios.

4.4.2 Synthetic objects dataset

The 'Real objects' dataset cannot be evaluated fairly by most video and image inpainting method, due to their inability to perform novel view synthesis. Hence, a separate variant of the dataset, augmented with synthetic elements, is introduced. The scenes in this variant are the same as those in the 'Real objects' dataset, but only the sequence without the object is utilized. Subsequently, a 3D object mesh from ShapeNet [21] is manually placed in each scene.

The object is positioned in a plausible coordinates and size, such as a laptop on a table. The masks are derived by projecting the mesh into the input images, with this being the sole utilization of the 3D object mesh. Regarding this synthetic dataset, every 8th frame is utilized for testing, while the remaining frames are used for training the NeRF model.

4.4.3 ARKitScenes dataset

ARKitScenes [22] stands out as the first RGB-D dataset that has been captured using a depth sensor that is now widely available. Furthermore, it holds the distinction of being the largest collection of indoor scene understanding data ever amassed. Alongside the raw and processed data, ARKitScenes includes meticulously acquired high-resolution depth maps, which were obtained using a stationary laser scanner.

Additionally, a comprehensive set of 3D oriented bounding boxes has been manually annotated for a diverse range of furniture categories. To facilitate research in the field, it have been also included helper scripts designed for two specific sub tasks, namely RGB-D guided upsampling and 3D object detection.

Overall it can be said that ARKitScenes is probably the biggest 3D dataset as it contains about 5000 captures of over 1700 different unique scenes.

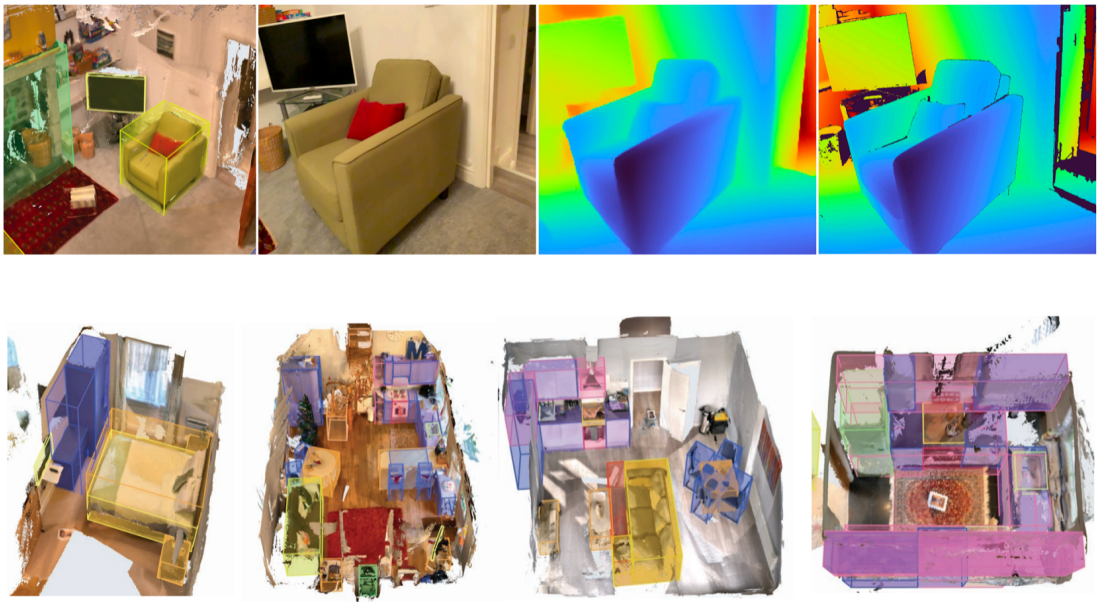


Figure 4.7 – ARKitScenes data samples

By providing this dataset, the aim is to propel the advancement of state-of-the-art techniques while presenting new challenges that more accurately reflect real-world scenarios.

The validation of the approach is further carried out qualitatively on ARKitScenes. In this dataset, consisting of 1,661 scenes, depth information was captured using iPhone Lidar.

4.4 Model architecture an method details

Fisrts step was to implement a model following the schema (fig. 4.8) from the original paper [23]. Two color heads, F_θ^c and F_θ^{MV}, are included in the NeRF formulation, with F_θ^{MV} not taking the view direction as input. Multi-view consistency is encouraged by this, and the scene and uncertainty of the 2D inpaintings for the automated view selection are jointly modeled using uncertainty variables U_n as demonstrated on the schema below.

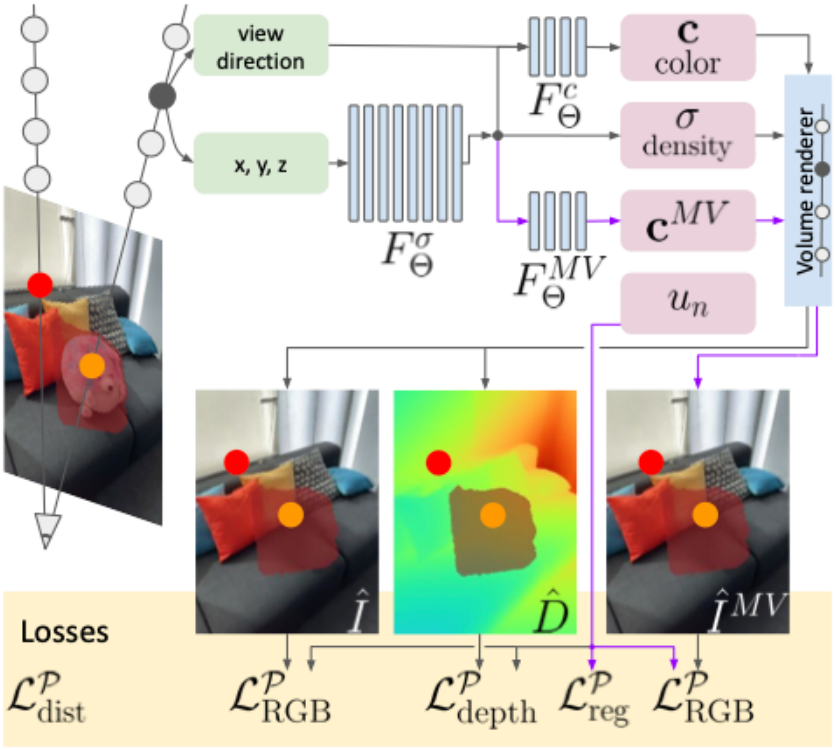


Figure 4.8 – Model arcitecture

The proposed method is implemented based on RegNeRF [24] and Mip-NeRF [25] architectures. The shared MLP F_θ^σ is composed of eight 256-wide layers, while the branches F_θ^c and F_θ^{MV} consist of four layers that are 128-wide each. The resulting output is then activated using softplus function for density, ReLU for uncertainty, and sigmoid for the color channel. The terms in the loss function are weighted with $\lambda_{RGB} = \lambda_{depth} = \lambda_{dist} = 1$, and $\lambda_{reg} = 0.005$. Optimization is performed using optimizer called Adam which uses initial learning rate value of lr = 0.0005, and the scheduler from RegNeRF is utilized. A filtering step is conducted, where low confidence images are removed every $K_{grad} = 50{,}000$ steps, resulting in $K_{outer} = 4$ filtering steps.

It is also worth mentioning the Mask refinement process. In practice, it is observed that masks that were gathered from the annotated 3D bounding boxes may be quite rough and contain large parts of the background. The inpainting quality is negatively affected by large masks.

Therefore, a mask refinement step is proposed to obtain masks that are more precise around the object. This step is not required for already precise input masks. Intuitively, in this mask refinement step, parts of the 3D bounding box that represent empty space are removed.

All points inside the 3D bounding box in the reconstructed 3D point cloud are taken as a starting point. The mask of a better quality is then obtained by generating these points into each image and then performing an easy comparison with the depth map to check for occlusions in the current image. Any pixel leaks caused by sensor noise are dilated and eroded using binary dilation to clean up the resulting mask. The effect of the mask refinement process is demonstrated in figure 4.9. As a result we get smaller masks and inpainting results of a better quality.

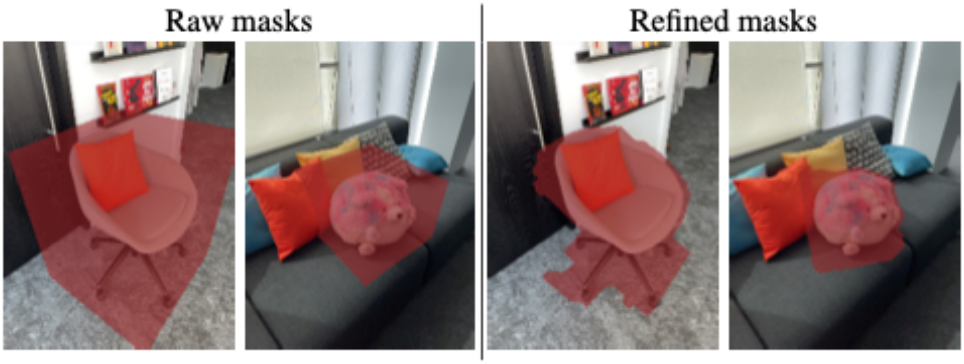


Figure 4.9 – Mask refinement

4.6 Metrics

To assess the object removal and inpainting quality, the system's output image is compared with the ground truth sample for each test image sample in the available dataset. All metrics used for evaluation are computed solely within the masked region. Metrics for the entire image were omitted.

The three standard metrics for NeRF evaluation were utilized. They are PSNR that stands for peak signal-to-noise ratio, SSIM that on the other hand

stands for structural similarity index measure, and LPIPS – Learned Perceptual Image Patch Similarity.

To evaluate the geometric completion, the L1 and L2 loss functions between the generated and ground-truth depth maps within the masked regions are computed. The metrics are then averaged across all frames of a sequence and further averaged across all sequences.

4.7 Experiments results analysis

In table 4.1 and table 4.1, a comparison is made between described approach and alternative methods for object removal, specifically focusing on methods that utilize an underlying NeRF representation.

Table 4.1 – Comparison of different approaches on real objects dataset

Approach	PSNR↑	SSIM↑	LPIPS↓
LaMa	27.999	0.898	0.060
Masked NeRF	26.126	0.882	0.093
PixelSynth	25.481	0.887	0.116
CompNVS	17.389	0.823	0.171
Current	29.437	0.916	0.078

Table 4.2 – Comparison of different approaches on synthetic objects dataset

Approach	PSNR↑	SSIM↑	LPIPS↓
Masked NeRF	21.644	0.815	0.142
PixelSynth	25.438	0.851	0.152
Current	25.271	0.859	0.125

For image and video inpainting baselines, a comparison is conducted with a state-of-the-art method – LaMa [26]. Their reference implementation and provided trained network are utilized for this comparison. It should be noted that

neither of these methods supports novel view synthesis, therefore their evaluation is limited to the synthetic objects dataset.

NeRF-based ablations are compared, focusing on different training approaches for a baseline NeRF model. The architecture of Masked NeRF [27] baseline involves training a NeRF model using the complete input RGB-D data, but with pixels and depths within the masked regions disregarded in the NeRF losses. The Inpainted Images approach trains a NeRF model using all inpainted images, excluding the inpainted depth maps. In contrast, the Inpainted Images + Inpainted Depth approach utilizes both the inpainted images and inpainted depth maps.

3D scene completion is compared with several published works. It should be noted that none of these baselines specifically focus on inpainting, and thus the results should be interpreted in light of this fact. The publicly available implementations of these works are used for the comparison.

PixelSynth [28] and CompNVS [29] were originally proposed for scene outpainting. Their objective is to complete a scene based on one or a few frames, enabling novel view synthesis. Both approaches are dependent from a generative model of indoor scenes and do not require test-time optimization.

As depicted in Table 1, the suggested method outperforms other baselines for novel-view synthesis across most appearance and depth metrics. Additionally, in comparison to the single image inpainting method LaMa , described method achieves close-to-multi-view consistency, effectively reducing inter-frame flickering.

In figure 4.10, it is demonstrated that the selected object is successfully removed by the proposed method in comparison to the baselines. Masked NeRF is observed to fail in completing large holes, while Inpainted NeRF suffers from inadequate inpaintings in the training set.

However, the suggested method is capable of leveraging the 2D inpaintings while avoiding the integration of artifacts by removing the corresponding input frames.

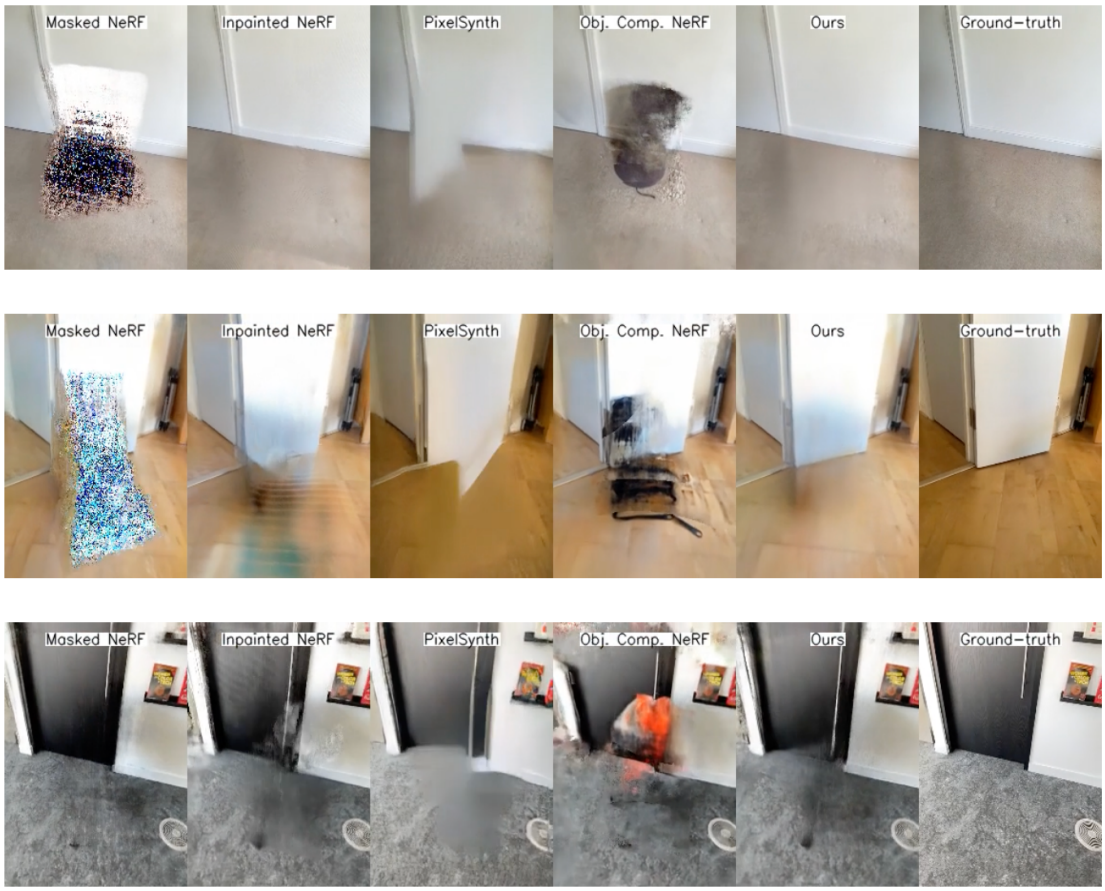


Figure 4.10 – Qualitive comparison with other approaches

Furthermore, the use of inpaintings in the desxribed method assists in mitigating the appearance of artifacts below the surface of the object, which distinguishes it from Object compositional NeRF. When compared to and, the described method exhibits a greater ability to generate plausible scene completions. Additionally, results from the ARKitScenes dataset are showcased in figure 4.11.

Figure 4.11 – Results on ARKitScenes dataset

Through experimental validation, it has been demonstrated that the proposed method enhances novel-view synthesis from 3D inpainted scenes in comparison to existing approaches.

One the disadvantages and limitations of the proposed approach it the fact that when all frames fail in the 2D inpainting method, such as when the mask covers a large portion of the image, the described method is unable to recover. Additionally, if the shadows of the removed object are not included in the object mask, they are retained by the described method.

CONCLUSIONS

During the research that was conducted while working on the current qualification work with the topic «Research on Neural Radiance Field approaches and applications» the relevance of the proposed topic was proved. The introduction to the topic of Neural Radiance Filed models was obtained as well as an experience of working with 3D data. Analysis and comparison of the existing solutions was conducted.

It was conducted analysis of such models and approaches as original NeRF, NeRF++, NeRF in the wild, RegNeRF and Mip-NeRF, Shap-e (the most recent paper that was published in the beginning of May, and uses NeRF to generated 3D animated images from text prompt), LaMa, Masked NeRF, PixelSynth and other methods. Based on experience with each of them the suggested approach for objext removal was implemented.

After the research several of the architectures were implemented using Python programming language as well as deep learning frameworks such as PyTorch and TensorFlow. After training the choosen models and using both pretrained weights a comparison was conducted using both real world objects and synthetic ones as well as ARKitScenes datasets.

The described approach showed slightly better results than most of other approaches both in quantitive comparison and qualitive one. Models were successfully able to remove highlighted (masked) objects from the pictures and do the required inpainting.

In future this model definitely requires the further improvements considering both quality of output results and optimization of performance and stability. The other desired improvement is to train the model on «in the wild» images dataset to make it robust towards wide variety of scenes with different lightning/illumination.

Overall, during this qualification and skills of working with Deep Learning models were improved. A lot of work was done towards the research of

architectures and variations of NeRF models available, as well as general experience of working with 3D data was obtained. In addition practical research was conducted, and during that research analysis and investigation of different architectures such ad Masked NeRF was done as well as task of removing objects from image using NeRF was solved.

REFERENCES

1. Hu Z., Bodyanskiy Y., Tyshchenko O., Kulishova N. A Multidimentional Extended Neo-Fuzzy Neuron for Facial Expression Recognition. Int. Journal of Intelligent Systems and Applications (USA). 2017. Vol. 9, No. 9. P. 29–36.

2. Bodyanskiy Y., Kulishova N., Chala O. The Extended Multidimensional Neo-Fuzzy System and Its Fast Learning in Pattern Recognition Tasks, Multidisciplinary Digital Publishing Institute. 2018. Vol. 3, no. 4. P. 63.

3. Bodyanskiy Y., Zaychenko Y., Hamidov G., Kulishova N. Multilayer GMDH-neuro-fuzzy network based on extended neo-fuzzy neurons and its application in online facial expression recognition, Системні дослідження та інформаційні технології. 2020. Vol. 3. P. 66–78.

4. Jakub Langr, Vladimir Bok. GANs in Action: Deep learning with generative adversarial networks, 2019.

5. Alankrita Aggarwal, Mamta Mittal , Gopi Battineni. Generative adversarial network: An overview of theory and applications. URL: https://doi.org/10.1016/j.jjimei.2020.100004 (дата звернення: 14.04.2023).

6. Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, Ren Ng, NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, 3 Aug, 2020, URL: https://arxiv.org/pdf/2003.08934.pdf (Last accessed: 23.03.2023).

7. Sudheendra Vijayanarasimhan, Susanna Ricco, Cordelia Schmid, Rahul Sukthankar, SfM-Net: Learning of Structure and Motion from Video, 25 Apr, 2017, URL: https://arxiv.org/abs/1704.07804 (Last accessed: 23.04.2023).

8. Deep 2d-neural network and its fast learning / Y. Bodyanskiy et al. 2018 IEEE second international conference on data stream mining & processing (DSMP), Lviv, Ukraine, 21–25 August 2018. 2018. URL: https://doi.org/10.1109/dsmp.2018.8478578 (date of access: 25.03.2022).

9. Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, DeepVoxels: Learning Persistent 3D Feature Embeddings, 2 Dec, 2018, URL: https://arxiv.org/pdf/1812.01024.pdf (Last accessed: 26.04.2023).

10. Alain Horé, Djemel Ziou, Image Quality Metrics: PSNR vs. SSIM, 27 Oct, 2010, URL: https://ieeexplore.ieee.org/document/5596999 (Last accessed: 21.04.2023).

11. Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, Oliver Wang, The Unreasonable Effectiveness of Deep Features as a Perceptual Metric, 11 Jan, 2018, URL: https://arxiv.org/abs/1801.03924 (Last accessed: 29.04.2023).

12. Matthew Tancik, Ethan Weber, Evonne Ng, Justin Kerr, Alexander Kristoffersen, David McAllister, A Modular Framework for Neural Radiance Field Development, 8 Feb, 2023, URL: https://arxiv.org/pdf/2302.04264v1.pdf (Last accessed: 27.03.2023).

13. Kai Zhang, Gernot Riegler, Noah Snavely, Vladlen Koltun, NeRF++: Analyzing and Improving Neural Radiance Fields, 15 Oct, 2020, URL: https://arxiv.org/abs/2010.07492 (Last accessed: 11.05.2023).

14. Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, Vladlen Koltun, Tanks and Temples: Benchmarking Large-Scale Scene Reconstruction, 1 Oct, 2017, URL: https://www.tanksandtemples.org (Last accessed: 06.05.2023).

15. Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, Daniel Duckworth, NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections, 6 Jun, 2021, URL: https://arxiv.org/pdf/2008.02268v3.pdf (Last accessed: 28.04.2023).

16. Piotr Bojanowski, Armand Joulin, David Lopez-Paz, Arthur Szlam, Optimizing the Latent Space of Generative Networks, 18 Jul, 2017, URL: https://arxiv.org/abs/1707.05776 (Last accessed: 28.04.2023).

17. Shih-Yang Su, Frank Yu, Michael Zollhoefer, Helge Rhodin, A-NeRF: Articulated Neural Radiance Fields for Learning Human Shape, Appearance, and Pose, 11 Feb, 2021, URL: https://arxiv.org/abs/2102.06199 (Last accessed: 27.04.2023).

18. NumPy documentation. URL: https://numpy.org/doc/stable/ (Last accessed: 12.05.2023).

19. PyTorch documentation. URL: https://pytorch.org/docs/index.html (Last accessed: 12.05.2023).

20. TensorFlow API framework documentation. URL: https://www.tensorflow.org/api_docs/python/tf (Last accessed: 12.05.2023).

21. Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, ShapeNet: An Information-Rich 3D Model Repository, 9 Dec, 2015, URL: https://arxiv.org/abs/1512.03012 (Last accessed: 27.04.2023).

22. Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Tal Dimry, Yuri Feigin, Peter Fu, ARKitScenes: A Diverse Real-World Dataset For 3D Indoor Scene Understanding Using Mobile RGB-D Data, 17 Nov, 2021, URL: https://arxiv.org/pdf/2111.08897v3.pdf (Last accessed: 05.05.2023).

23. Silvan Weder, Guillermo Garcia-Hernando, Aron Monszpart, Marc Pollefeys, Removing Objects From Neural Radiance Fields, 22 Dec, 2022, URL: https://arxiv.org/pdf/2212.11966.pdf (Last accessed: 26.03.2023).

24. Michael Niemeyer, Jonathan T. Barron, Ben Mildenhall, Mehdi S. M. Sajjadi, RegNeRF: Regularizing Neural Radiance Fields for View Synthesis from Sparse Inputs, 1 Dec, 2021, URL: https://arxiv.org/abs/2112.00724 (Last accessed: 05.05.2023).

25. Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields, 24 Mar, 2021, URL: https://arxiv.org/abs/2103.13415 (Last accessed: 05.05.2023).

26. Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Resolution-robust Large Mask Inpainting with Fourier Convolutions, 15 Sep, 2021, URL: https://arxiv.org/abs/2109.07161 (Last accessed: 08.05.2023).

27. Daniel Rho, Byeonghyeon Lee, Seungtae Nam, Joo Chan Lee, Masked Wavelet Representation for Compact Neural Radiance Fields, 18 Dec, 2022,

URL: https://arxiv.org/abs/2212.09069 (Last accessed: 09.05.2023).

28. Chris Rockwell, David F. Fouhey, Justin Johnson, PixelSynth: Generating a 3D-Consistent Experience from a Single Image, 21 Aug, 2021, URL: https://arxiv.org/abs/2108.05892 (Last accessed: 10.05.2023).

29. Zuoyue Li, Tianxing Fan, Zhenqiang Li, Zhaopeng Cui, Yoichi Sato, CompNVS: Novel View Synthesis with Scene Completion, 23 Jul, 2022, URL: https://arxiv.org/abs/2207.11467 (Last accessed: 10.05.2023).

ANNEX A

List of qualification work

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	77 с.
	Інші документи	
2.	Презентаційні матеріали	__ плакатів

.	.	Прізвище та ініціали.	Підп.	Дата		
Розробив		Гура О.А.			Шифр групи	Код напр./спец.
Перевірив		Кулішова Н.Є.			СШІм-21-2	122
Н.контр.		Малєєва І.А.			ХНУРЕ кафедра ШІ	
Затв.		Філатов В.О.				

Дослідження підходу поля нейронного випромінювання та його застосунків