


# Generación de datos sintéticos en una escena pseudoaletoria utilizando modelos NeRF

Grado en Ingeniería Informática

María Naranjo Almeida

Supervisado por:
Adrián Peñate Sánchez
Fernando Rivas Manzaneque

Julio 2023

# Agradecimientos

*A mi familia, en especial a mis padres y mi hermano por su apoyo y paciencia.*

*A mis tutores Adrián y Fernando por su implicación, ayuda y confianza.*

*A mis amigos y compañeros por acompañarme en esta travesía.*

# Resumen

El modelado 3D de objetos translúcidos y transparentes es un reto para la fotogrametría tradicional, un claro ejemplo de este tipo de productos son los que aparecen en supermercados donde el envoltorio es translúcido para permitir ver el producto interior. Esta limitación introduce una gran problemática para los comercios que utilizan estas técnicas para la comprobación del estado de las mercancías, la visualización de su stock en plataformas digitales o simplemente poder hacer detección automática de los mismos.

La utilización de herramientas basadas en Inteligencia Artificial está siendo revolucionaria en la creación de modelos 3D de objetos con reflejos especulares, como pueden ser los Neural Radiance Fields (NeRF). Esta propuesta de TFG propone la creación de datos sintéticos tridimensionales basados en modelos NeRF, acabando con las limitaciones comentadas anteriormente, y dispuestos en una escena donde éstos puedan interactuar de manera pseudoaleatoria con otros objetos. Este sistema de generación de datos sintéticos complejos permitirá una multitud de aplicaciones.

En resumidas cuentas, este TFG permitirá crear modelos NeRF de múltiples objetos complejos, hacerlos interactuar en un simulador de física y generar datos sintéticos a partir de las interacciones de los mismos. Esto creará escenas complejas que posibilitarán el entrenamiento de otros modelos de Deep Learning como aquellos que realizan la detección de objetos a partir de datos, que, aunque siendo sintéticos, modelan de una forma más rica las posibles escenas a encontrar en casos reales de supermercados online entre otros.

# Abstract

Modelling 3D translucent or transparent objects could challenge traditional photogrammetry. As an example of this type of product, we have everyday objects found in supermarkets that are translucent to make the product's interior visible. This limitation introduces a big problem in businesses which use this technique to check good's status, stock visualization, or even automatic detection.

Using tools based on Artificial Intelligence has been revolutionary in creating 3D models of objects with specular reflections, such as Neural Radiance Fields (NeRF). This project proposes generating synthetic tridimensional data based on NeRF models, breaking down the constraints mentioned above, and displaying them in a scene where these objects could interact with each other with a pseudo-random behavior. This generation system of complex synthetic data could be used in several applications.

To sum up, this project allows the creation of NeRF models using multiple complex objects that interact in a physical simulator and generate synthetic data from the interaction with other entities. This will create complex scenes, allowing the training of other Deep Learning models, such as those that detect the objects from data that, although they are synthetic, model the possible scenes in a more prosperous way to find actual cases in online supermarkets, among others.

# Índice general


**1. Introducción** **1**
1.1. Objetivos del estudio . . . . . 2

**2. Planificación del proyecto** **3**
2.1. Metodología . . . . . 3
2.2. Tecnologías utilizadas . . . . . 3
2.2.1. Lenguajes de programación . . . . . 3
2.2.2. Bibliotecas . . . . . 4
2.2.3. Aplicaciones software . . . . . 4

**3. Estado del arte** **5**
3.1. Introducción . . . . . 5
3.2. Redes neuronales artificiales . . . . . 6
3.2.1. Neurona artificial . . . . . 6
3.2.2. Perceptrón multicapa . . . . . 6
3.3. Campo neuronal o *neural field* . . . . . 6
3.4. NeRF: Neural Radiance Fields . . . . . 7
3.5. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections 8
3.6. MipNeRF y MipNeRF-360 . . . . . 10
3.7. NeRF– – (Neural Radiance Fields Without Known Camera Parameters) . . . 12
3.8. Ref-NeRF . . . . . 12
3.9. Instant-NGP . . . . . 12
3.10. Nerfacto . . . . . 13

**4. Competencias específicas cubiertas** **15**

**5. Desarrollo** **16**
5.1. Generación de modelos NeRF . . . . . 16
5.1.1. Análisis . . . . . 17
5.1.2. Adaptaciones . . . . . 18
5.2. Composición de escenas . . . . . 23
5.2.1. Un único NeRF y otros objetos de Blender . . . . . 25
5.2.2. Múltiples NeRF . . . . . 32
5.3. Generación de una escena pseudoaleatoria y aplicación de física . . . . . 37

5.4. Captura de objetos . . . . . 41
5.5. Proceso de generación de datos sintéticos en escena pseudoaleatoria con los modelos NeRF de los objetos capturados . . . . . 43

**6. Generación de la base de datos de imágenes 47**
6.1. Objetos capturados . . . . . 47
6.1.1. Botella de lejía . . . . . 47
6.1.2. Caja de cereales . . . . . 49
6.1.3. Lata de refresco . . . . . 51
6.1.4. Rollo de papel . . . . . 53
6.1.5. Cacao en polvo . . . . . 55
6.1.6. Botella de champú . . . . . 57
6.2. Escenas . . . . . 60
6.2.1. Escena 1 . . . . . 60
6.2.2. Escena 2 . . . . . 61
6.2.3. Escena 3 . . . . . 62
6.2.4. Escena 4 . . . . . 63
6.2.5. Escena 5 . . . . . 64
6.2.6. Escena 6 . . . . . 65
6.2.7. Escena 7 . . . . . 66
6.2.8. Escena 8 . . . . . 67
6.2.9. Escena 9 . . . . . 68
6.2.10. Escena 10 . . . . . 69

**7. Aportaciones 71**

**8. Conclusiones y trabajo futuro 72**

# Índice de figuras


1.1. Reconstrucción de un vaso utilizando fotogrametría tradicional. Foto: Mikolas Zuza [30] . . . . . 1

3.1. Escena icónica «Garden» del dataset de Mip-NeRF 360 en un estudio cinematográfico. Foto: Fernando Rivas Manzaneque, Neural Radiance Fields [8] . . 5
3.2. Perceptrón multicapa . . . . . 7
3.3. Visión general de la representación de escenas NeRF y el procedimiento de renderizado diferenciable [10] . . . . . 8
3.4. Arquitectura de NeRF-W [9] . . . . . 9
3.5. Comparación del muestreo de NeRF y de Mip-NeRF[1] . . . . . 10
3.6. Comparación entre las arquitecturas de MipNeRF (esquema superior) y MipNeRF-360 (esquema inferior) [2] . . . . . 11
3.7. Comparación del espacio con y sin contracción [24] . . . . . 11
3.8. Comparación de núcleos sin fusión y con fusión. . . . . 13
3.9. Estructura de Nerfacto [23] . . . . . 14

5.1. Etapas del desarrollo . . . . . 16
5.2. Renderizado de Lego con Nerfacto sin las adaptaciones necesarias (izquierda) e imagen de referencia (derecha). . . . . 19
5.3. Distribución de densidad en la escena según la distancia de los planos. . . . . 20
5.4. Colisionador de caja a partir de las posiciones normalizadas. . . . . 21
5.5. Colisionador de caja a partir de las poses de cámara. . . . . 21
5.6. Resultado cualitativo del entrenamiento de Lego con las modificaciones de Nerfacto . . . . . 23
5.7. Complemento de Nerfstudio importado y activado en Blender . . . . . 24
5.8. Nueva sección creada por el complemento en el apartado de «Propiedades de procesamiento» . . . . . 24
5.9. Opciones para importar en Blender . . . . . 25
5.10. Malla de Lego obtenida con Poisson en Blender . . . . . 26
5.11. Escena con un NeRF y dos objetos de Blender . . . . . 26
5.12. Malla del Lego con el procesamiento deshabilitado . . . . . 27
5.13. Nodos de composición. A la izquierda la configuración por defecto. A la derecha la configuración para renderizar la profundidad . . . . . 28
5.14. Configuración de las propiedades de la salida. . . . . 28
5.15. Generación del fichero JSON de la cámara. . . . . 29

5.16. Escena en Blender. . . . . 30
5.17. Renderizado del color de la escena de Blender sin procesar el Lego. . . . . 31
5.18. Renderizado del color de Lego con Nerfstudio. . . . . 31
5.19. Mapa de segmentación de la escena compuesta. El color rosa representa que lo que está delante son los elementos de Blender. El color amarillo representa que Lego está delante. . . . . 32
5.20. Resultado final de la composición. . . . . 32
5.21. NeRF del objeto Hotdog con Nerfacto. . . . . 33
5.22. Escena en Blender con dos NeRFs (Lego y Hotdog) y dos objetos de Blender (cubo y mono) . . . . . 34
5.23. Escena en Blender. . . . . 34
5.24. Renderizado del color de la Escena de Blender sin procesar Lego ni Hotdog. . 35
5.25. Renderizado del color de Lego con Nerfstudio. . . . . 35
5.26. Renderizado del color de Hotdog con Nerfstudio. . . . . 36
5.27. Mapa de segmentación de la escena compuesta. El color rosa representa que lo que está delante son los elementos de Blender. El color amarillo representa que Lego está delante. El color verde muestra que Hotdog está delante. El color azul indica fondo. . . . . 36
5.28. Resultado final de la composición. . . . . 37
5.29. Opción de Aleatorizar transformaciones . . . . . 37
5.30. Menú de los posibles parámetros a aleatorizar . . . . . 38
5.31. Configuración de la escena para agregar la física en Blender . . . . . 39
5.32. Configuración de un cuerpo rígido activo. . . . . 40
5.33. Simulación de la caída de los objetos en una caja con posición inicial pseudo-aleatoria. . . . . 41
5.34. Set de captura. . . . . 42
5.35. Salida del comando ns-process-data . . . . . 43
5.36. Malla de un bote de cacao en polvo. Resaltado en un rectángulo rojo se encuentra el centro de masa del objeto, el cual está desplazado . . . . . 44
5.37. Malla de un bote de cacao en polvo con el origen en el centro de la superficie (malla padre). Resaltado en un rectángulo azul se encuentra el centro de masa del objeto, el cual está centrado . . . . . 45
5.38. Asociación de la malla padre (CacaoP) y de la malla hijo (CacaoH) . . . . . 45
5.39. Resultado final de la asociación. En el rectángulo rojo se muestra el origen desplazado de la malla hijo y en el rectángulo azul se indica el origen centrado de la malla padre. . . . . 46

6.1. Pila de imágenes de la botella de Lejía. . . . . 48
6.2. Comparación de la imagen de referencia con el NeRF obtenido. A la izquierda la imagen de referencia (*Ground truth*) y a la derecha el NeRF. . . . . 48
6.3. NeRF de la lejía obtenido tras el entrenamiento con las adaptaciones realizadas con Nerfacto. . . . . 49
6.4. Pila de imágenes de la caja de cereales. . . . . 50
6.5. Comparación de la imagen de referencia con el NeRF obtenido. A la izquierda la imagen de referencia (*Ground truth*) y a la derecha el NeRF. . . . . 50

6.6. NeRF de la caja de cereales obtenido tras el entrenamiento con las adaptaciones realizadas con Nerfacto. . . . . 51
6.7. Pila de imágenes de la lata de refresco. . . . . 52
6.8. Comparación de la imagen de referencia con el NeRF obtenido. A la izquierda la imagen de referencia (*Ground truth*) y a la derecha el NeRF. . . . . 52
6.9. NeRF de la lata de refresco obtenido tras el entrenamiento con las adaptaciones realizadas con Nerfacto. . . . . 53
6.10. Pila de imágenes del rollo de papel. . . . . 54
6.11. Comparación de la imagen de referencia con el NeRF obtenido. A la izquierda la imagen de referencia (*Ground truth*) y a la derecha el NeRF. . . . . 54
6.12. NeRF del rollo de papel obtenido tras el entrenamiento con las adaptaciones realizadas con Nerfacto. . . . . 55
6.13. Pila de imágenes del cacao en polvo. . . . . 56
6.14. Comparación de la imagen de referencia con el NeRF obtenido. A la izquierda la imagen de referencia (*Ground truth*) y a la derecha el NeRF. . . . . 56
6.15. NeRF del cacao en polvo obtenido tras el entrenamiento con las adaptaciones realizadas con Nerfacto. . . . . 57
6.16. Pila de imágenes de la botella de champú. . . . . 58
6.17. Comparación de la imagen de referencia con el NeRF obtenido. A la izquierda la imagen de referencia (*Ground truth*) y a la derecha el NeRF. . . . . 58
6.18. NeRF de la botella de champú obtenido tras el entrenamiento con las adaptaciones realizadas con Nerfacto. . . . . 59
6.19. Secuencia de la primera escena . . . . . 61
6.20. Secuencia de la segunda escena . . . . . 62
6.21. Secuencia de la tercera escena . . . . . 63
6.22. Secuencia de la cuarta escena . . . . . 64
6.23. Secuencia de la quinta escena . . . . . 65
6.24. Secuencia de la sexta escena . . . . . 66
6.25. Secuencia de la séptima escena . . . . . 67
6.26. Secuencia de la octava escena . . . . . 68
6.27. Secuencia de la novena escena . . . . . 69
6.28. Secuencia de la décima escena . . . . . 70

# Índice de cuadros

5.1. Comparación de NeRF, Mip-NeRF y Nerfacto con el dataset de presentado en Mip-NeRF 360 [2] . . . . . 18
5.2. Valores medios de los resultados obtenidos tras el entrenamiento de Lego de 22000 iteraciones. . . . . 23
5.3. Valores medios de los resultados obtenidos tras el entrenamiento de Hotdog de 22000 iteraciones. . . . . 33

6.1. Métricas obtenidas de los NeRFs de los distintos artículos. Además, se ha destacado en negrita aquellos valores que sean los mejores para cada métrica. 59

# Capítulo 1

# Introducción

La fotogrametría es muy utilizada en las cadenas de abastecimiento para realizar diversas tareas e incrementar la eficiencia. Ésta se puede utilizar para realizar automáticamente un seguimiento de los productos o para analizar el estado y detectar defectos de los mismos. Sin embargo, este método presenta ciertas deficiencias, ya que es dependiente de la iluminación y no es capaz de representar adecuadamente objetos que presentan transparencias o reflejos especulares.

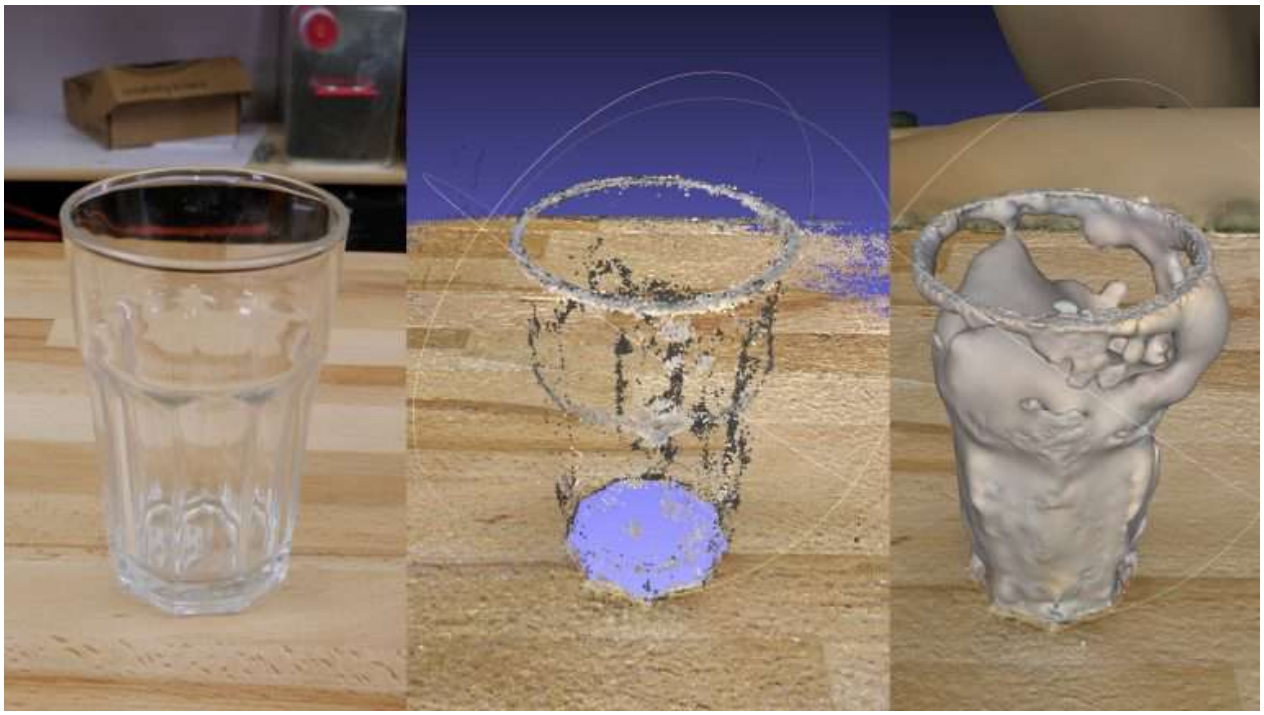

Ilustración 1.1: Reconstrucción de un vaso utilizando fotogrametría tradicional. Foto: Mikolas Zuza [30]

La motivación inicial para realizar este estudio proviene por diversas razones. Una de ellas ha sido la constante presencia de problemas en relación a esta área de investigación. Diversos

grupos de investigación han intentado profundizar en el tema y buscar soluciones a los retos y dificultades que se presentan.

En la actualidad, una de las tecnologías más relevante y revolucionarias en este ámbito es la aplicación de modelos de Inteligencia Artificial y Deep Learning. Estas herramientas se han convertido en una gran oportunidad para abordar de manera efectiva los problemas y desafíos que se presentan.

Uno de los modelos que más repercusión está teniendo y, tanto él como sus variantes, son los que están constituyendo el estado del arte son los basados en Neural Radiance Field (NeRF) [10].

## 1.1. Objetivos del estudio

Los objetivos que se plantean en este Trabajo de Final de Grado son los siguientes:

1. Generación de un dataset de artículos que se puedan encontrar en un supermercado.
2. Análisis de las distintas tecnologías y técnicas disponibles para la generación de volúmenes utilizando campos neurales.
3. Aplicación de la tecnología/técnica seleccionada
4. Renderizado y composición de una escena física pseudoaleatoria
5. Análisis de los resultados obtenidos.

# Capítulo 2

# Planificación del proyecto

## 2.1. Metodología

Para la realización de este trabajo de fin de grado se ha utilizado una metodología de desarrollo ágil, en concreto se ha usado el marco de trabajo denominado SCRUM. Éste se basa en la mejora continua, la colaboración y la adaptabilidad. Su objetivo es la entrega de resultados parciales de alta calidad de manera incremental y en ciclos de tiempo cortos (*sprints*). Además, se centra en el uso de buenas prácticas.

De esta manera, se realizaron diversas reuniones semanales para realizar una supervisión del proyecto. Por un lado, se tuvieron reuniones grupales con el tutor académico y los otros alumnos tutorizados. Por otro lado, se realizaron sesiones de seguimiento con el tutor de empresa. En ambas se mostraba los avances realizados y se fijaban nuevos objetivos para la siguiente semana.

Este proyecto se abordó de manera incremental, primero realizando las tareas, pruebas y experimentos más básicos con un dataset utilizado como referencia (*benchmark*), denominado *nerf_synthetic*[10]. Una vez alcanzados los objetivos mínimos, se procedió a la captura de objetos que se pueden encontrar en un supermercado y la posterior creación del dataset.

## 2.2. Tecnologías utilizadas

Para el desarrollo de este proyecto se ha hecho uso de distintas herramientas y tecnologías que han permitido el progreso eficaz del mismo, obteniendo resultados exitosos.

### 2.2.1. Lenguajes de programación

**Python [16].** Se trata de un lenguaje de programación de código abierto, interpretado, de alto nivel y de código abierto. Éste se utiliza en distintos campos, desde la creación de

aplicaciones de web hasta inteligencia artificial debido a su simpleza y legibilidad. La razón fundamental para el uso de este lenguaje en el proyecto se debe a que cuenta con una gran variedad de bibliotecas relacionadas con las redes neuronales e inteligencia artificial.

### 2.2.2. Bibliotecas

**PyTorch** [17] Es una librería de código abierto para Python centrada en el aprendizaje profundo, capaz de utilizar de manera eficiente las GPUs o CPUs del equipo en el que se ejecute. Se basa en el concepto de tensores (matriz multidimensional), ofreciendo un conjunto de abstracciones de alto nivel, facilitando el desarrollo de redes neuronales y de modelos de aprendizaje profundo.

**NumPy** [13] Se refiere a una biblioteca para Python especializada en computación.

**OpenCV**[14] Se trata de una librería de código abierto, cuyo fin es de ofrecer herramientas sencillas y útiles para tareas de visión por computador y aprendizaje automático.

**Nerfstudio** [24] Esta biblioteca ha sido fundamental para el desarrollo de este proyecto. Ofrece implementaciones interpretables de distintos modelos NeRF y proporciona un entorno sencillo para crear, entrenar, exportar y visualizar NeRFs.

**COLMAP** [19, 20, 21] Corresponde a una librería para fotogrametría y reconstrucción 3D, entre otras muchas más funciones. En el caso particular de este proyecto, se ha utilizado para poder obtener las poses de cámara de las imágenes tomadas para la generación del dataset.

### 2.2.3. Aplicaciones software

**Blender** [3] Es un motor gráfico de código abierto y multiplataforma. Se ha utilizado en este trabajo de final de grado con el propósito de generar las escenas pseudoaleatorias y simular la física de los objetos.

**PhotoRoom** [15] Es un programa de edición de imágenes que ofrece diversas funcionalidades tanto gratuitas como de pago. Se ha usado la versión gratuita con el fin de realizar un preprocesamiento de las imágenes capturadas para crear el dataset.

**Weights & Biases** [27] Se trata de una herramienta para el seguimiento, evaluación y visualización de experimentos de inteligencia artificial.

# Capítulo 3

# Estado del arte

## 3.1. Introducción

Como se comentó anteriormente, las técnicas basadas en NeRF se encuentran en auge y están consiguiendo diversos hitos en el renderizado de modelos tridimensionales. Actualmente, se emplean en la industria cinematográfica, así como en la creación de videojuegos, entre otros diversos ámbitos de aplicación.

Ilustración 3.1: Escena icónica «Garden» del dataset de Mip-NeRF 360 en un estudio cinematográfico. Foto: Fernando Rivas Manzaneque, Neural Radiance Fields [8]

Dado que varios grupos de investigación se están centrando en este campo, en los últimos meses se han desarrollado diversos modelos que utilizan los fundamentos de los Neural Ra-

diance Fields para generar sus propios avances en la materia. Sin embargo, en este TFG se va profundizar el estudio en aquellos más relevantes para el proyecto.

## 3.2. Redes neuronales artificiales

Las redes neuronales artificiales son el medio por el cual se pueden crear sistemas inteligentes. Como su propio nombre indica, pretenden imitar el comportamiento del cerebro humano, inspirándose en el concepto de las neuronas y su interconexión. La función principal de una neurona es recibir señales o impulsos nerviosos de otras neuronas o células sensoriales, procesar esta información y luego enviar señales electroquímicas a otras células, incluyendo otras neuronas, músculos o glándulas. En otras palabras, las neuronas permiten la comunicación entre diferentes partes del cuerpo y son la base fundamental del procesamiento de información en el sistema nervioso.

### 3.2.1. Neurona artificial

Siguiendo el mismo concepto, una neurona artificial pretende imitar las funciones básicas de unas neuronas biológicas. Siendo la unidad básica de procesamiento de una red neuronal. Acepta parámetros de entrada, realiza diversos cálculos sobre ellos y se obtiene una salida. Cada entrada está asociada con un peso, el cual indica la importancia de cada una para la salida. Además, cada neurona tiene otro parámetro denominado sesgo y se trata de una constante que permite que la red sea más flexible y se adapte a los patrones de la entrada.

### 3.2.2. Perceptrón multicapa

Existen diversas estructuras de redes neuronales, sin embargo, la más predominante en los NeRFs es el perceptrón multicapa o *MLP*. Está formada por una capa de entrada de variables, seguida de un número diverso de capas denominadas capas ocultas y, finalmente, una capa de salida que devuelve los resultados de la red. Cabe destacar que las neuronas de una capa se conectan con todas las neuronas de la siguiente capa.

## 3.3. Campo neuronal o *neural field*

Un campo es una cantidad definida por todas las coordenadas espaciales y/o temporales. En otras palabras, es una cantidad que puede variar en diferentes posiciones del espacio o tiempo. El término campo neural o *neural field* hace referencia a una red neuronal que está basada en coordenadas para parametrizar propiedades físicas. Esto permite a las redes neuronales codificar de manera eficiente y precisa una señal en cualquier resolución y dimensión [28].

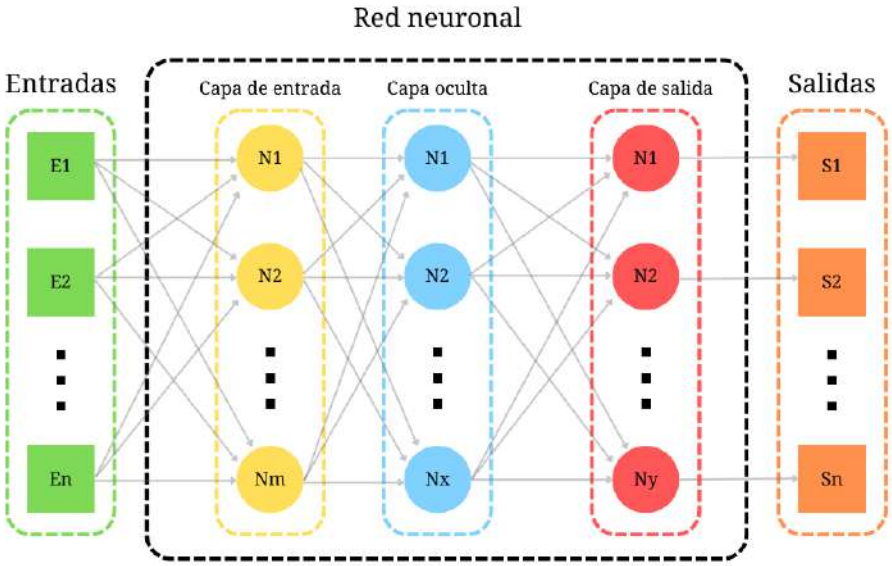


Ilustración 3.2: Perceptrón multicapa

Tras los avances realizados en el ámbito del aprendizaje automático, se han aplicado los campos neurales para resolver problemas relacionados con la visión por computador utilizando la inteligencia artificial. En el contexto de este proyecto, es esencial conocer este concepto, ya que es aplicado en los modelos utilizados, empleando las imágenes para establecer un campo que cuantifica la intensidad del color en cada píxel.

## 3.4. NeRF: Neural Radiance Fields

El modelo que se describe a continuación se trata de la primera publicación sobre este ámbito, la cual fue un hito importante en la investigación de los gráficos por computador, ya que propuso una nueva técnica para generar modelos 3D fotorrealistas a partir de fotografías [10]. En 2020 se establecía como modelo que mejoraba el estado del arte existente, obteniendo destacables resultados en la generación de nuevas perspectivas y vistas que la red no había visto con anterioridad.

En este caso, se utiliza un campo neural para expresar la **radiancia**[1]. Para ello, emplea una red profunda multicapa para representar la escena. Para alimentar dicha red, es necesario introducir como entrada un vector de cinco dimensiones. Este consiste en las coordenadas en el espacio del punto $(x, y, z)$ y la dirección de la vista $(\theta, \phi)$. Una vez se tienen estos datos,

[1]Cantidad de luz emitida por un objeto.

son convertidos en una codificación posicional (*positional encoding*, en inglés), es decir, se convierten las coordenadas en características de mayor dimensión. Esta técnica permite a la red representar las funciones de alta frecuencia (zonas de la imagen donde existe un cambio drástico de color o geometría). El uso de *positional encodings* es necesario, ya que las redes neurales están sesgadas para aprender señales de baja frecuencia [18]. Esta técnica aproxima dichas funciones a señales sinusoidales, también conocidas como características de Fourier o *Fourier features* [22].

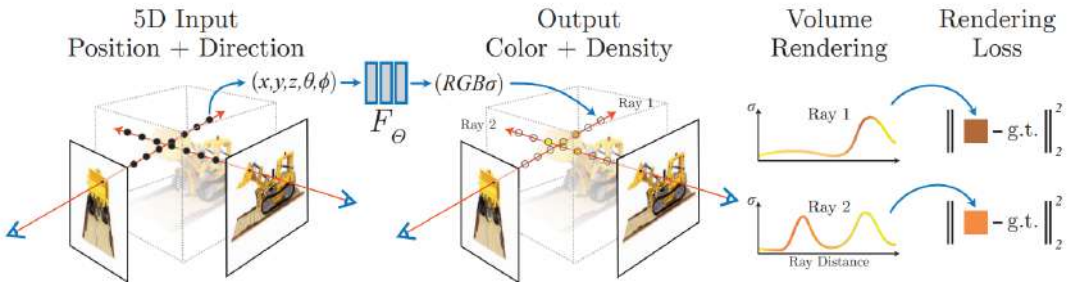


Ilustración 3.3: Visión general de la representación de escenas NeRF y el procedimiento de renderizado diferenciable [10]

La salida obtenida de NeRF son la densidad y el color dependiente de la vista y de la localización $(x, y, z)$. Este método permite optimizar la red multicapa sin ningún tipo de convolución, ya que todo el proceso de renderizado es diferenciable y, por lo tanto, es posible utilizar el descenso por el gradiente.

Para entrenar esta red es necesario proporcionarle una serie de imágenes con sus poses de cámara. El proceso de renderizado es el siguiente:

1. Dada una imagen, se trazan rayos a lo largo de las escena para tomar muestras a lo largo del rayo para generar una serie de puntos 3D.
2. Se utilizan esos puntos con las direcciones de vista como entrada de la red neuronal para obtener los colores y las densidades.
3. Se acumulan los colores y las densidades utilizando técnicas clásicas de renderizado volumétrico para tener un único valor de color y densidad en el rayo trazado.

## 3.5. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections

Una de las desventajas que muestra el modelo anterior es la falta de control de las condiciones de luz. Es decir, la escena es completamente dependiente de las condiciones lumínicas con las que se tomaron las imágenes, siendo incapaz de modelar otro tipo fenómenos del mundo real. Este proyecto ha recogido imágenes de diversos monumentos históricos tomadas por turistas

para analizarlas y reconstruir toda la escena tridimensionalmente [9]. Es por ello, que este método presenta mejoras con respecto a NeRF en dos aspectos fundamentales:

**Variaciones fotométricas.** Una misma escena puede presentar en distintas imágenes variaciones en las condiciones atmosféricas, condiciones lumínicas e, incluso, existencia de distintos tipos de cámara, tiempo de exposición, filtros, etc.

**Objetos transitorios.** Se trata de la presencia de objetos en la escena que pueden estar en movimiento o que pueden ocluir otras partes de la misma.

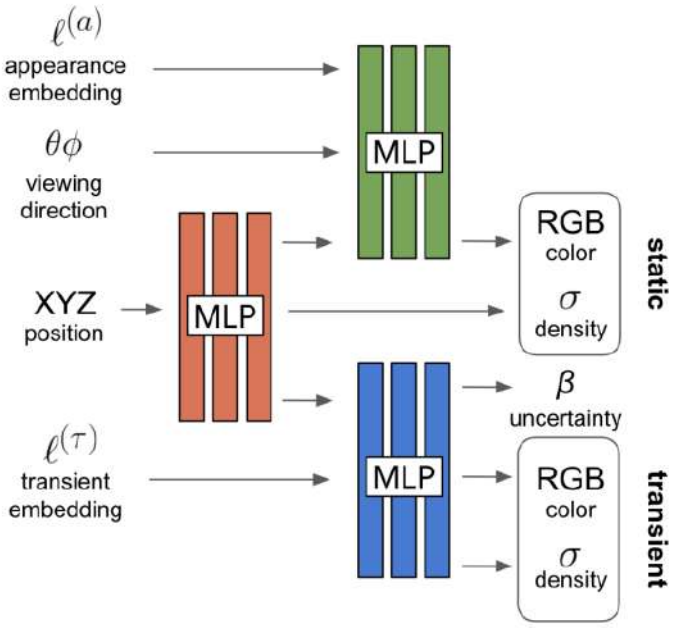


Ilustración 3.4: Arquitectura de NeRF-W [9]

Para alcanzar dichos objetivos, NeRF in the Wild (NeRF-W) presenta un entrenamiento en 3 fases:

1. En una primera estancia se calcula la densidad de la escena.
2. A continuación, con la información de la densidad, se obtiene el color de la zona estática de la escena.
3. Finalmente, se calcula el color y la densidad de la parte transitoria, y la incertidumbre. Esta incertidumbre permite que la red se centre en aquellos elementos que sean estáticos. Toda esta información transitoria se almacena en lo denominado *vector de embeddings*[2]. Además, para manejar las variaciones lumínicas se codifica la información de la apariencia en una estructura denominada embedding de apariencia.

[2]Representación de un objeto en un espacio vectorial

## 3.6. MipNeRF y MipNeRF-360

El artículo de MipNeRF [1] mejora la publicación original de NeRF introduciendo frustums cónicos[3] en cada píxel para evitar que el renderizado sea borroso o se produzca *aliasing* o efecto escalera. Además, se mejora la calidad, captando detalles más pequeños, y disminuye el tiempo de renderizado comparado con NeRF.

El nombre del modelo está inspirado en la técnica de gráficos por ordenador para evitar *aliasing*, denominada *mipmapping*. Esta permite el modelado de escenas en múltiples escalas haciendo uso de pirámides de imágenes.

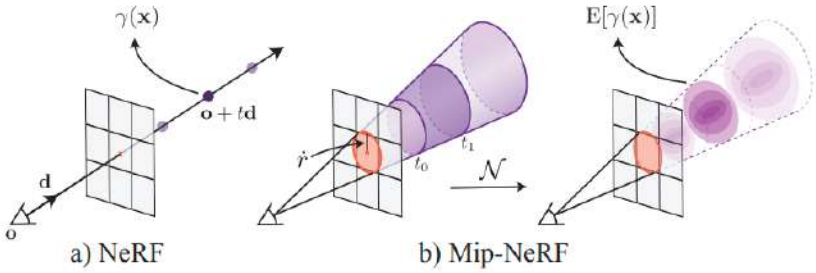


Ilustración 3.5: Comparación del muestreo de NeRF y de Mip-NeRF[1]

Un tiempo después de la publicación de MipNeRF, los autores presentaron una mejora con respecto al trabajo anterior, llamado MipNeRF-360 [2]. Este es capaz de modelar escenas no acotadas[4] (*unbounded*). Algunas características que se introdujeron y que han sido un gran aporte a la mejora de la calidad han sido:

***Proposal sampler*** Se trata de un perceptrón multicapa que permite consolidar las localizaciones de las muestras tomadas de la imagen a las regiones de la escena en las que más contribuyen (en la mayoría de los casos se trata de la primera superficie de intersección), ya que cuando las escenas son muy extensas, empiezan a aparecer ciertas ambigüedades.

**Contracción del espacio** Establece una esfera de radio 2, en la cual los rayos que se salgan de esta esfera son contraídos. Es decir, se envuelve el espacio a un volumen específico.

[3]Porción visible de un objeto en una escena, el cual se encuentra dentro de un cono de visión.

[4]Escena en la que no se delimita el espacio en el que puede moverse una cámara y en el que el contenido puede estar a cualquier distancia de la cámara.

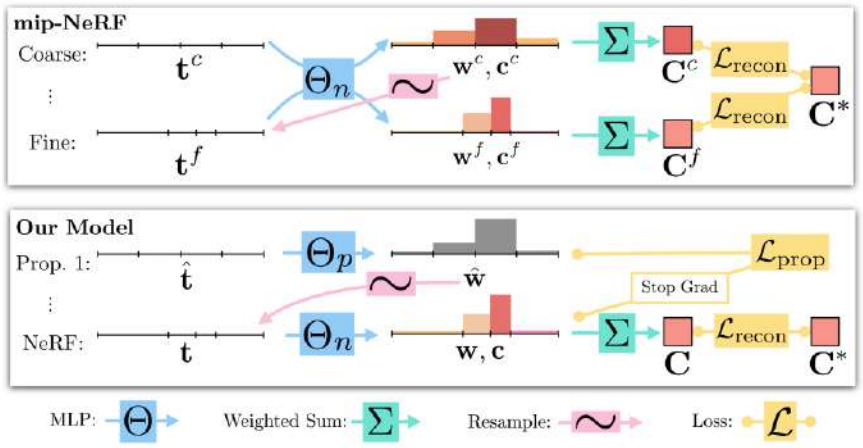


Ilustración 3.6: Comparación entre las arquitecturas de MipNeRF (esquema superior) y MipNeRF-360 (esquema inferior) [2]

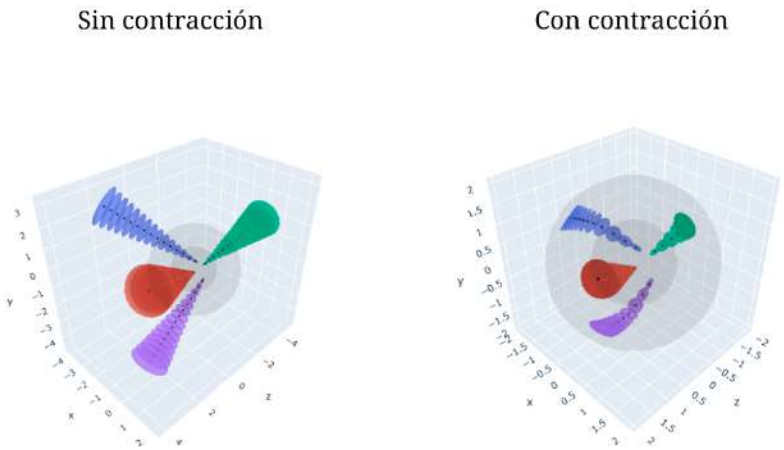


Ilustración 3.7: Comparación del espacio con y sin contracción [24]

## 3.7. NeRF−− (Neural Radiance Fields Without Known Camera Parameters)

Este artículo aborda uno de los desafíos más significativos que enfrentan los modelos basados en NeRF: los errores en la estimación de las poses y los parámetros intrínsecos de las cámaras. Reconociendo esta problemática, los autores han propuesto e implementado una solución para optimizar de manera eficiente los parámetros de las cámaras [26].

La optimización de los parámetros de las cámaras es crucial para lograr resultados precisos y realistas al generar representaciones 3D. Al ajustar correctamente los parámetros intrínsecos, como la distancia focal, la apertura del diafragma y los coeficientes de distorsión, así como los parámetros extrínsecos, como la posición y orientación de la cámara, se puede mejorar la precisión y la calidad de las representaciones generadas.

La implementación de esta optimización de parámetros de cámaras proporciona una solución efectiva para mejorar la precisión y la fidelidad de los modelos basados en NeRF.

## 3.8. Ref-NeRF

Las técnicas basadas en NeRF han demostrado ser altamente efectivas en la representación de estructuras geométricas detalladas y complejas en escenas 3D. Pueden capturar con precisión las formas y los detalles sutiles de objetos pequeños, así como las variaciones en la iluminación y la textura de la superficie.

Sin embargo, una de las limitaciones que se ha observado es que los modelos basados en NeRF pueden tener dificultades para representar de manera óptima superficies altamente reflectantes o brillantes. Estas superficies, como espejos, metales pulidos o materiales especulares, presentan un desafío debido a su comportamiento de reflectancia complejo y altamente direccional.

Para abordar esta limitación, este artículo presenta una sustitución de la manera en que NeRF representa la radiancia saliente con una radiancia reflejada y la regularización de los vectores normales [25].

## 3.9. Instant-NGP

El modelo *Instant Neural Graphic Primitives* o Instant-NGP [11] fue desarrollado por la empresa NVIDIA, fue nombrado como «El mejor invento de 2022» por la revista TIME [7] y premiado como «Mejor artículo» de la conferencia SIGGRAPH en 2022 [4]. Surge como mejora a ciertas limitaciones que presentaban los modelos anteriores, como son el tiempo de entrenamiento, ya que para poder entrenar una escena con modelos como NeRF o MipNeRF era necesario emplear horas o días.

Instant-NGP mejora el estado del arte establecido por los modelos descritos anteriormente, gracias a las siguientes medidas:

**Reducción del coste computacional** Se realiza una nueva manera de codificar la entrada que permite utilizar redes más pequeñas sin afectar la calidad del resultado. Esto se logra mediante el uso de tablas *hash* multirresolución que contiene los vectores de características. Esta estructura evita las colisiones *hash*.

**Aumento del paralelismo de las tareas** De manera general, cada núcleo de la GPU realiza una operación. Para evitar operaciones innecesarias y aumentar el paralelismo, este proyecto implementa lo que se denomina «núcleos CUDA completamente fusionados» («*Fully-fused CUDA kernels*»). Esto permite que, cuando se quieren realizar 2 operaciones distintas y no interesa el valor intermedio, se puedan fusionar dos núcleos de la GPU. De esta forma se obtienen resultados de calidad en segundos.

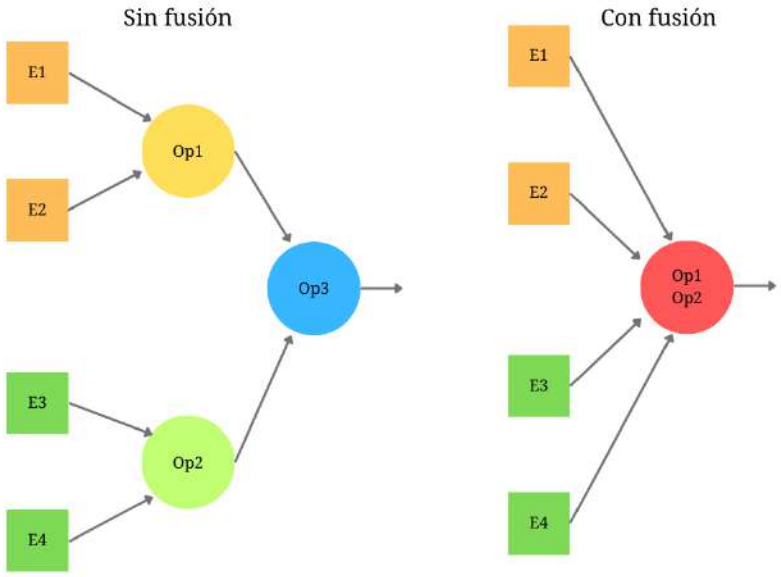


Ilustración 3.8: Comparación de núcleos sin fusión y con fusión.

## 3.10. Nerfacto

Se trata de un modelo desarrollado por los autores de Nerfstudio, no está publicado en ningún artículo, pero está inspirado en modelos como Instant-NGP, MipNeRF-360, NeRF-W y NeRF- -. Sus objetivos consisten en lograr una capacitación rápida, obtener resultados de alta calidad y prescindir de equipos de computación de gran capacidad.

Como se puede observar en la Ilustración 3.9, el modelo que se presenta en esta sección hace uso de varios de los artefactos y técnicas explicadas anteriormente. Estos elementos

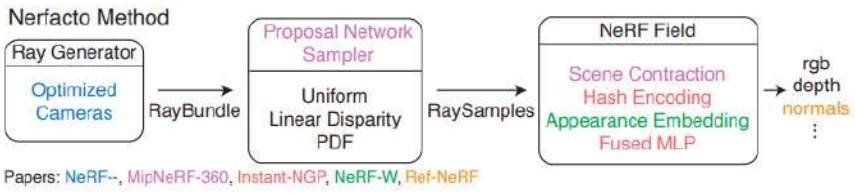


Ilustración 3.9: Estructura de Nerfacto [23]

son utilizados de manera conjunta para mejorar la calidad y la precisión de los resultados obtenidos.

El proceso de funcionamiento del modelo es el siguiente:

1. Generación de rayos. Es usual que la predicción de las poses de las cámaras contengan errores, generando artefactos indeseados en la escena y un empeoramiento de la calidad. Es por esta razón que un paso esencial para evitar los problemas comentados sea realizar una optimización y refinamiento de las poses. Tras llevar acabo dicho procedimiento, se genera un haz de rayos (*Ray Bundle*).

2. Muestreo a trozos. Este permite que se tomen muestras de manera uniforme hasta una cierta distancia establecida. A partir de dicha distancia, las muestras restantes son distribuidas de manera que, con cada muestra tomada, el espacio de esta con la siguiente incrementa.

3. Proposal Sampler. Una vez se tienen las muestras, estas son introducidas en un *Proposal Sampler*, descrito en MipNeRF-360. Esta técnica permite mejorar significativamente la calidad.

4. Campo de densidad. A continuación, se utilizan ciertas características de Instant-NGP, como una codificación hash y un perceptrón multicapa fusionado para que el proceso sea más eficiente y rápido.

5. Contracción del espacio. Como se describió en MipNeRF-360, se utiliza esta técnica para acotar el espacio, contrayendo la escena a un volumen fijo. Sin embargo, una diferencia significativa con respecto al otro método es que Nerfacto utiliza un cubo de lado 4 para contraer la escena, en vez de una esfera de radio 2.

6. Campo NeRF (*NeRF Field*). Finalmente, se utiliza este campo para obtener la densidad, el color y, adicionalmente, las normales (utilizando técnicas desarrolladas por Ref-NeRF) a partir de las coordenadas $(x, y, z)$ del punto, la dirección de vista y un *embedding* de apariencia, propuesto en NeRF-W.

# Capítulo 4

# Competencias específicas cubiertas

Según se define en la Memoria del Plan de Estudios del Grado en Ingeniería Informática[5], en este trabajo de final de grado se han abarcado dos competencias específicas.

**CI15.** «Conocimiento y aplicación de los principios fundamentales y técnicas básicas de los sistemas inteligentes y su aplicación práctica.» Para poder realizar este proyecto ha sido necesario conocer los fundamentos de los sistemas inteligentes, ya que los modelos y herramientas utilizados se han basado en la inteligencia artificial.

**TFG.** «Ejercicio original a realizar individualmente y presentar y defender ante un tribunal universitario, consistente en un proyecto en el ámbito de las tecnologías específicas de la Ingeniería en Informática de naturaleza profesional en el que se sinteticen e integren las competencias adquiridas en las enseñanzas.» El desarrollo del presente trabajo avala que esta competencia ha sido abarcada.

# Capítulo 5

# Desarrollo

En este capítulo se relata el desarrollo seguido en este proyecto. Este se ha dividido en 4 etapas distintas.

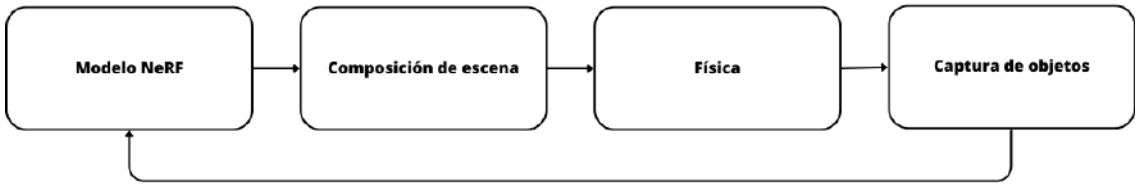


Ilustración 5.1: Etapas del desarrollo

Como se puede ver en la Ilustración 5.1, la primera etapa consiste en la generación de modelos NeRF. En un comienzo se utilizó el dataset *nerf_synthetic* [10] para la realización de distintos experimentos. Una vez se obtuvieron los resultados deseados, se realizó la composición de escenas, que a su vez fue dividida en distintas fases, como se describirá en secciones siguientes. Seguidamente, se realizó la simulación de la física de los objetos. Finalmente, se realizó la captura de objetos de supermercado y se volvió a realizar el proceso para dichos artículos.

## 5.1. Generación de modelos NeRF

Para poder generar los NeRF, lo primero que se realizó fue un análisis de los modelos disponibles en Nerfstudio.

### 5.1.1. Análisis

Para poder realizar una comparación de los distintos métodos disponibles en Nerfstudio, es necesario utilizar el mismo criterio. En el campo de la calidad de imagen y la evaluación de modelos de procesamiento de imágenes, existen varias métricas utilizadas para medir la similitud y la calidad perceptual entre imágenes. Entre las métricas más comunes se encuentran: PSNR (Peak Signal-to-Noise Ratio), SSIM (Structural Similarity Index) y LPIPS (Learned Perceptual Image Patch Similarity). Estas desempeñan un papel fundamental en la evaluación cuantitativa de los resultados obtenidos por los algoritmos de procesamiento de imágenes.

**PSNR.** Representa la relación entre el valor máximo de una señal y el ruido que afecta a la calidad de la imagen reconstruida o procesada. En otras palabras, mide la diferencia entre la imagen de referencia y la imagen procesada en términos de distorsión. Un valor alto implica que hay poca distorsión y viceversa.

$$\text{PSNR} = 20 \cdot \log_{10} \left( \frac{\text{MAX}_f}{\sqrt{\text{MSE}}} \right) \tag{5.1}$$

Donde MAX es el valor máximo de la señal que existe en la imagen original y MSE es el error cuadrático medio.

$$\text{MSE} = \frac{1}{m \cdot n} \sum_{0}^{m-1} \sum_{0}^{n-1} \|f(i,j) - g(i,j)\|^2 \tag{5.2}$$

Donde $m$ es el número de filas, $n$ el número de columnas, $f(i,j)$ representa el píxel de coordenadas $i, j$ de la imagen original y $g(i,j)$ representa el píxel de coordenadas $i, j$ de la imagen procesada.

Sin embargo, esta métrica representa una cierta limitación. El PSNR únicamente es una medida de la diferencia de intensidad entre la imagen original y la procesada, sin tener en cuenta las características visuales o perceptuales. Es por esta razón que es necesario utilizar otras métricas que sí consideren estos aspectos, como el SSIM.

**SSIM.** Esta métrica evalúa la similitud entre dos imágenes, en términos de luminancia, contraste y estructura. Los valores obtenidos van desde $[-1, 1]$, donde un valor cercano a 1 significa que hay mucha similitud estructural y viceversa [12].

El componente de luminancia compara el brillo entre las imágenes. Además, tiene en cuenta que el brillo no es lineal. El contraste mide la diferencia de amplitud entre las estructuras. Analiza cómo se perciben los cambios en el contraste local, lo cual es esencial para la calidad. La estructura examina la relación entre las estructuras de las imágenes.

**LPIPS.** Esta métrica, en comparación a las anteriores, utiliza una red neuronal para medir la similitud perceptual entre dos imágenes. Se basa en realizar un análisis de parches o fragmentos de la imagen, que son procesados para extraer características perceptuales

relevantes. A continuación, se calcula la distancia entre las características de los parches de las dos imágenes para obtener una medida de la similitud. Mientras menor sea esta distancia, mayor será la similitud y viceversa [29].

Una vez que se han definido y explicado las métricas que se utilizarán para evaluar los modelos, se puede proceder a realizar una comparación entre ellos, utilizando el mismo dataset. Los métodos que ofrece Nerfstudio son:

**NeRF.** Está basado en el artículo original [10]. Como los propios autores de Nerfstudio comentan, no es recomendable utilizar este modelo ya que, aunque se obtienen buenos resultados, el entrenamiento puede durar horas o, incluso, días.

**Mip-NeRF.** Mejora al modelo anterior, aumentando ligeramente la calidad.

**Nerfacto.** Como ya se ha comentado anteriormente, este modelo mejora a los dos anteriores en cuestión del tiempo empleado para entrenar.

| Modelo | PSNR ↑ | SSIM ↑ | LPIPS ↓ | Tiempo (horas) |
|---|---|---|---|---|
| NeRF | 23.85 | 0.605 | 0.451 | 4.16 |
| Mip-NeRF | 24.03 | 0.607 | 0.455 | 4.59 |
| Nerfacto | 26.75 | 0.748 | 0.307 | 0.5 |

Cuadro 5.1: Comparación de NeRF, Mip-NeRF y Nerfacto con el dataset de presentado en Mip-NeRF 360 [2]

**Instant-NGP.** Este modelo es mucho más rápido y se obtiene una calidad superior con respecto a los anteriores. Sin embargo, no ha podido ser utilizado en este proyecto debido a la capacidad computacional que requiere.

Como se ha podido observar en el Cuadro 5.1, el modelo que muestra un desempeño superior y requiere menos tiempo de entrenamiento es Nerfacto. Por consiguiente, se ha seleccionado este método como la base para este proyecto.

Una vez se ha seleccionado el modelo, lo primero que se realizó fue un entrenamiento con el objeto Lego del dataset *nerf_synthetic*.

Como se puede apreciar en la Ilustración 5.2, los resultados obtenidos exhiben un nivel de calidad deficiente. Por este motivo, se vio la necesidad de realizar diversas modificaciones al modelo para ajustarse a los datos utilizados en este trabajo.

### 5.1.2. Adaptaciones

Como ya se ha comentado anteriormente, Nerfacto está desarrollado para obtener resultados exitosos en escenas no acotadas y con variaciones de iluminación. Sin embargo, los parámetros de configuración utilizados no son los más óptimos para las escenas utilizadas en este proyecto. Es por ello que es necesario que se haga una adaptación del modelo.

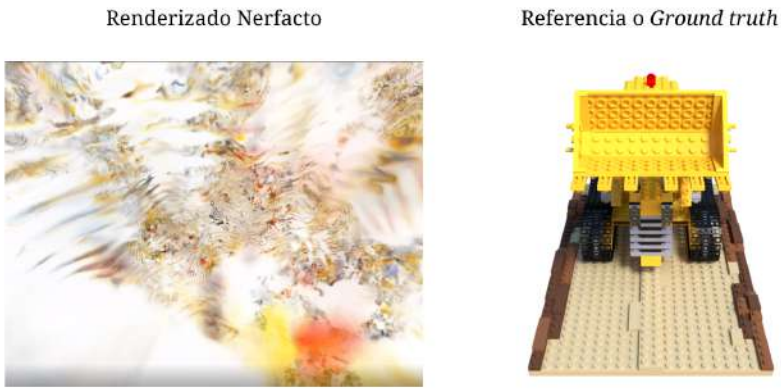


Ilustración 5.2: Renderizado de Lego con Nerfacto sin las adaptaciones necesarias (izquierda) e imagen de referencia (derecha).

Para poder comparar los resultados obtenidos de las adaptaciones, se ha utilizado como referencia el objeto Lego del dataset *nerf_synthetic*. Para obtener dichos datos, se ha utilizado la interfaz de línea de comandos ofrecida por Nerfstudio. En concreto, la orden:

```
$ ns-download-data blender
```

Una vez se tiene el dataset, se puede entrenar el modelo utilizando la orden específica para ello, especificando cuáles son los datos de entrada y el tipo (en este caso, blender-data):

```
$ ns-train nerfacto  --data data/blender/lego blender-data
```

Existen diversos parámetros optativos que se pueden utilizar para especificar distintas configuraciones adicionales. Algunos de ellos se verán en siguientes apartados donde se muestran las modificaciones realizadas.

#### 5.1.2.1. Colisionador

Por defecto, Nerfacto utiliza un colisionador denominado «*NearFarCollider*». Este establece dos planos (uno cercano y otro lejano) donde comienzan y terminan los rayos trazados que son muestreados. De manera predeterminada, el plano cercano está establecido a una distancia de 0.05 unidades y el lejano a 1000 unidades.

Al tener una escena sintética, acotada y de tamaño pequeño, provoca que haya una concentración desproporcionada de puntos de muestra en áreas específicas de la escena, lo que generar una distribución de densidad no uniforme.

Esta problemática se soluciona fácilmente cambiando la distancia de los planos. Unos valores razonables son: para el plano cercano 2 unidades y para el lejano 6. No obstante, sigue presentando ciertos problemas porque los rayos pueden comenzar detrás de las cámaras, las cuales son zonas de la imagen donde no se tiene información y se genera ruido o artefactos

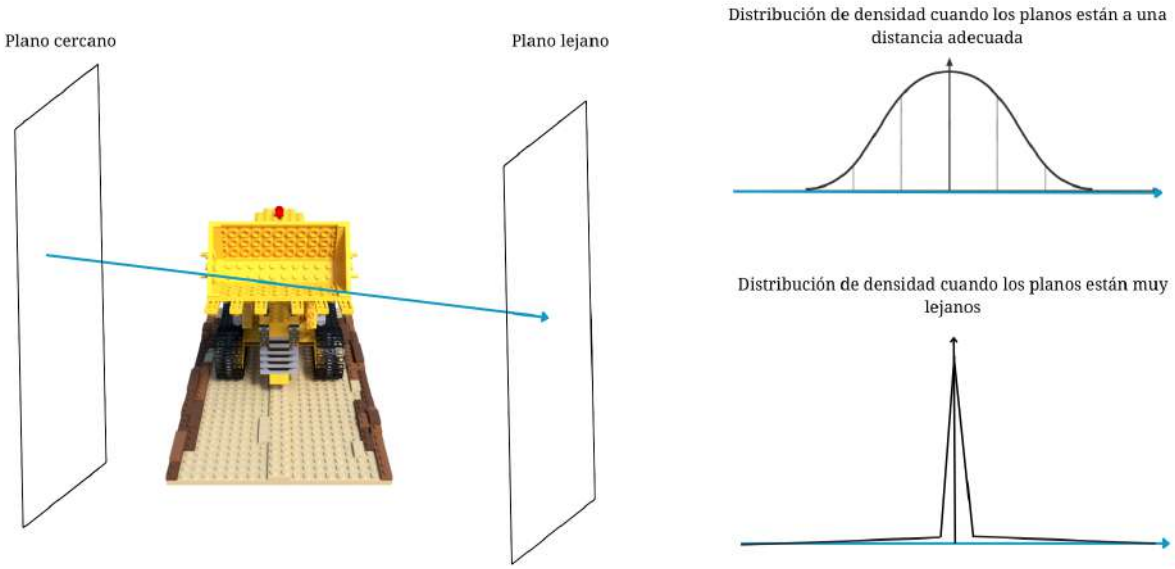


Ilustración 5.3: Distribución de densidad en la escena según la distancia de los planos.

indeseados. Una alternativa a la modificación de los planos podría ser cambiar el tipo de colisionador.

Nerfstudio ofrece otro colisionador, denominado *AABBBoxCollider*. Este establece un cubo o caja definido por la clase *SceneBox*. En esta aproximación únicamente se muestrea cuando el rayo se encuentra dentro de la caja. Existen diversas maneras de configurar este cubo, pero los más útiles para este proyecto son:

**Obtención de las posiciones normalizadas.** Devuelve las posiciones normalizadas en un rango de $[0, 1]$ basado en los límites de la caja. Para establecer los límites se puede hacer de la siguiente manera:

**A partir de los frustums.** Por defecto, el frustum es de tamaño 1. Es decir, se origina un cubo de tamaño 1.

**Ingresando manualmente los valores.** Dado que la escena es acotada, se sabe de antemano su tamaño. Por lo tanto, es posible introducir los límites de la caja manualmente. En este caso, es conocido que los límites del dataset de *nerf_synthetic* son [1.5,1.5,1.5] y [-1.5,-1.5,-1.5].

Debido a las circunstancias particulares que se presentan en este proyecto, se ha determinado que el uso de cualquiera de las dos técnicas descritas para definir la caja es equivalente. Esto significa que ambas técnicas proporcionan resultados similares o comparables en términos de los objetivos y requisitos del proyecto. Otras situaciones pueden requerir una evaluación diferente y pueden dar lugar a conclusiones distintas.

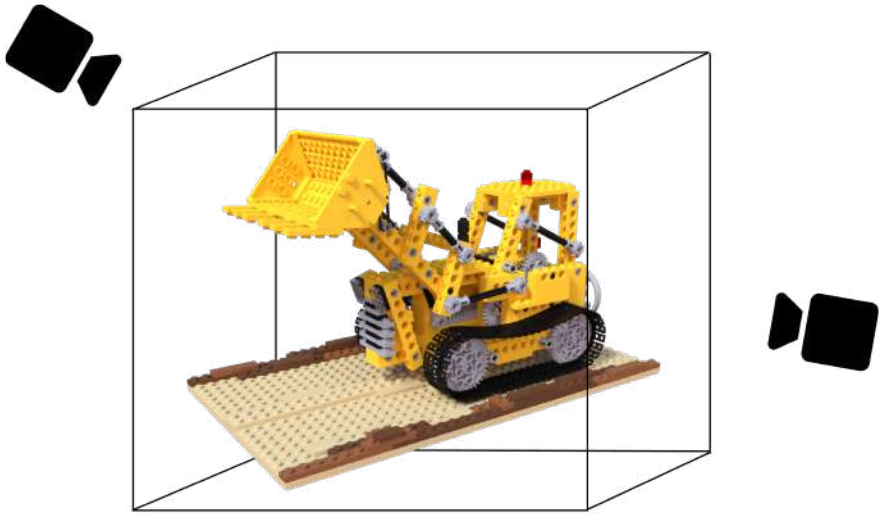

Ilustración 5.4: Colisionador de caja a partir de las posiciones normalizadas.

**Establecimiento de los límites a partir de las poses de cámara.** Configura la caja de manera que envuelva las poses de las cámaras. Esta opción es la menos recomendable ya que al generar rayos detrás de las cámaras, es muy probable que se tenga ruido en la escena.

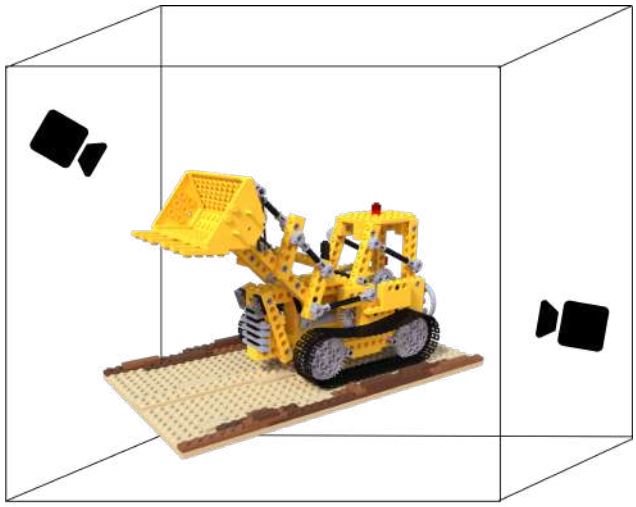

Ilustración 5.5: Colisionador de caja a partir de las poses de cámara.

#### 5.1.2.2. Contracción del espacio

Como ya se comentó en el Estado del Arte, Nerfacto es un modelo que está preparado para escenas reales sin límites. Es por ello que es necesario tener una noción de las fronteras que define dónde empieza y acaba el rayo. Un artefacto muy útil para solventar esta problemática es la contracción del espacio.

Sin embargo, en este proyecto únicamente se va a tratar con escenas acotadas de datasets sintéticos. Por dicha razón, el efecto que presenta la contracción del espacio no es necesario ni deseado, ya que, al no tener fondo, provoca la aparición de artefactos.

#### 5.1.2.3. Color de fondo

Dado que este proyecto únicamente trata imágenes sin fondo, se desea que cuando se realice el renderizado, el color de fondo sea blanco. Por defecto, Nerfacto está configurado para que el color de fondo sea del color de la última muestra tomada. Esta modificación es simple, ya que lo único que hay que realizar es la modificación de una variable en el modelo.

#### 5.1.2.4. Optimización de las cámaras

La optimización de las cámaras resulta de gran utilidad cuando las escenas utilizadas son reales y sus parámetros son calculados con distintos algoritmos. No obstante, con el dataset que está siendo utilizado no es necesario, ya que las poses de las cámaras no necesitan ser optimizadas y, si se hiciese, podría ocurrir un sobre-ajuste de sus parámetros, afectando a la calidad del resultado. Este parámetro se puede desactivar directamente desde la línea de comandos cuando se desea entrenar el modelo.

```
1    $ ns-train nerfacto --pipeline.datamanager.camera-optimizer.mode off
2    --data data/blender/lego blender-data
```

#### 5.1.2.5. Embedding de apariencia

Al igual que la modificación antes descrita, los embedding de apariencia son muy útiles en escenas reales con variación en las condiciones lumínicas. Sin embargo, debido a que en nuestras escenas prácticamente no existen cambios de luz, esta técnica puede provocar un deterioro en el resultado. Al igual que en el caso anterior, se puede especificar que no se utilice el embedding:

```
1    $ ns-train nerfacto --pipeline.model.use-average-appearance-embedding False
2    --data data/blender/lego blender-data
```

Finalmente, tras realizar las adaptaciones mencionadas, se logró entrenar exitosamente el objeto Lego del conjunto de datos. La orden utilizada para este propósito fue la siguiente:

```
1    $ ns-train nerfacto --pipeline.datamanager.camera-optimizer.mode off --
   pipeline.model.use-average-appearance-embedding False
2    --data data/blender/lego blender-data
```

Para poder evaluar si el resultado obtenido es exitoso, se han calculado las métricas previamente mencionadas. Como se puede observar en la tabla 5.2, se han obtenido valores bastante altos de PSNR y SSIM, lo cual indica una alta similitud entre las imágenes generadas y las imágenes de referencia. Además, se ha obtenido un valor bajo de LPIPS, lo que indica una baja distancia perceptual entre las imágenes generadas y las imágenes de referencia.

Ilustración 5.6: Resultado cualitativo del entrenamiento de Lego con las modificaciones de Nerfacto

Es importante destacar que estos valores favorables de las métricas se han logrado en un tiempo de ejecución considerablemente bajo. Esto demuestra la eficiencia y efectividad del método implementado.

Estos resultados indican que el modelo ha logrado generar imágenes de alta calidad y fidelidad en comparación con las imágenes de referencia, lo que sugiere un buen rendimiento del modelo en términos de similitud visual y percepción humana.

En conclusión, los resultados obtenidos muestran un buen desempeño del modelo en términos de las métricas utilizadas, lo cual es prometedor para su aplicación en este proyecto.

| PSNR ↑ | SSIM ↑ | LPIPS ↓ | Tiempo (**minutos**) |
|---|---|---|---|
| 33.33 | 0.970 | 0.015 | 24 |

Cuadro 5.2: Valores medios de los resultados obtenidos tras el entrenamiento de Lego de 22000 iteraciones.

## 5.2. Composición de escenas

Una vez obtenido el modelo NeRF del Lego, es posible proceder con el renderizado y composición de escenas tanto de un NeRF con varios objetos como de varios NeRFs. Sin embargo,

esto no se puede hacer directamente desde la librería de Nerfstudio, sino que es necesario el uso de otras herramientas. Para ello, se ha utilizado Blender. Se ha decido utilizar dicho motor gráfico porque los autores de Nerfstudio han creado un complemento que permite componer los NeRFs obtenidos de Nerfstudio, utilizando las mallas o nubes de puntos de los objetos, obteniendo las coordenadas de la cámara relativas a las transformaciones de los objetos.

Estas mallas o nubes únicamente sirven como referencia porque, como se verá a continuación, la calidad de ellas no es tan buena como la de la representación de NeRF. Por dicha razón, a estas mallas son las que se les va a aplicar la física y las transformaciones físicas y, posteriormente, serán sustituidas por el correspondiente NeRF.

Como se especifica en la documentación de Nerfstudio, lo primero que hay que realizar para hacer la composición es la descarga del código de la extensión, disponible en la documentación e importarlo en Blender.

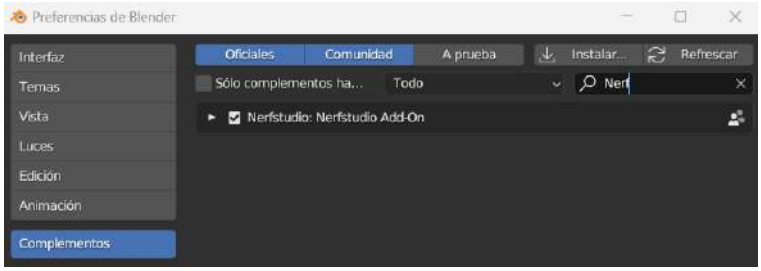


Ilustración 5.7: Complemento de Nerfstudio importado y activado en Blender

Esto provoca que en el apartado de «Propiedades de procesamiento» aparezca la siguiente sección:

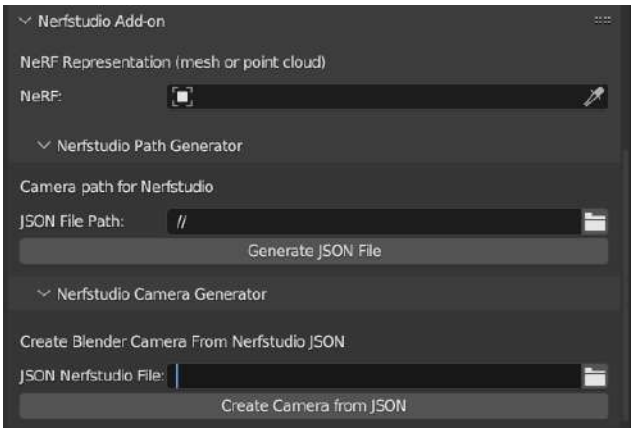


Ilustración 5.8: Nueva sección creada por el complemento en el apartado de «Propiedades de procesamiento»

El siguiente paso a realizar consiste en obtener la malla o la nube de puntos del NeRF. Para ello, se ha utilizado de nuevo la interfaz de línea de comandos que ofrece Nerfstudio.

```
1    $ ns-export {poisson, tsdf, pointcloud} --load-config CONFIG.yml --
   output-dir OUPUT_DIR)
```

Una vez se ha obtenido una malla o nube de puntos, se procede a importarla en Blender. El formato de importación dependerá del tipo de representación seleccionado, ya sea TSDF, Poisson o nube de puntos.

Si se ha utilizado el método de representación TSDF (Truncated Signed Distance Function), la malla se importará en formato de malla triangular (.obj). Este formato representa la superficie mediante una colección de triángulos definidos por sus vértices y caras.

En el caso de haber utilizado el método de representación de Poisson, la malla se importará en formato de malla triangular también (en esta ocasión .ply), ya que el algoritmo de Poisson genera una superficie suave basada en una representación implícita de la función de distancia.

Por otro lado, si se ha generado una nube de puntos, se puede importar en formato .ply. Estos formatos representan la nube de puntos como una colección de coordenadas tridimensionales sin conectividad de vértices.

Al importar la malla o la nube de puntos en Blender, se podrá visualizar y manipular el objeto en un entorno tridimensional, lo que permitirá realizar diferentes operaciones y modificaciones en función de los requerimientos del proyecto.

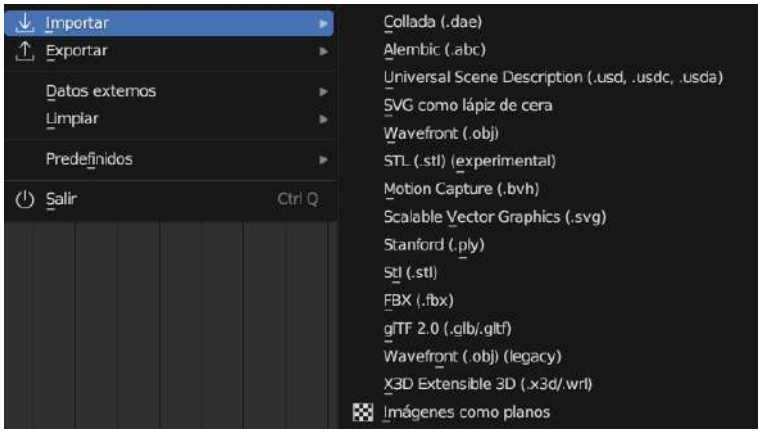


Ilustración 5.9: Opciones para importar en Blender

### 5.2.1. Un único NeRF y otros objetos de Blender

Una vez alcanzado este punto, se puede comenzar con el proceso de composición. Para ello, es necesario configurar la escena según las preferencias deseadas, lo cual implica agregar otros objetos, además del generado con NeRF. En este caso en particular, se han incorporado dos mallas predefinidas en Blender: un cubo y un mono. Estos objetos adicionales se añaden con el objetivo de enriquecer la escena y lograr la composición deseada. La incorporación de estos elementos adicionales, no solo contribuye a la estética y la narrativa visual de la composición, sino que también brinda mayor versatilidad y flexibilidad para la creación de la escena deseada. Además, permite combinar elementos generados por NeRF con objetos predefinidos en Blender u otros recursos disponibles, ampliando así las posibilidades creativas y la calidad visual del resultado final.

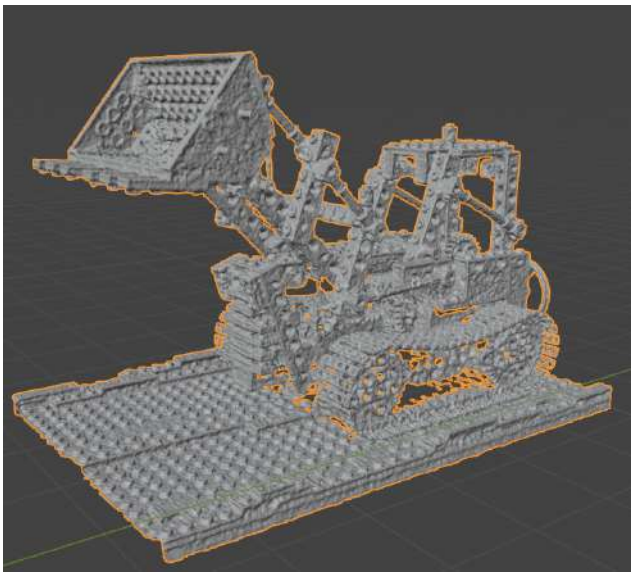

Ilustración 5.10: Malla de Lego obtenida con Poisson en Blender

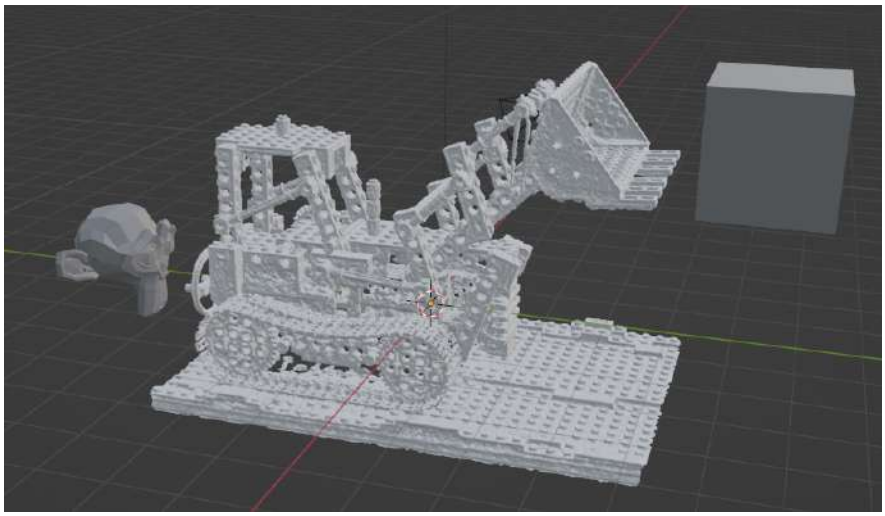

Ilustración 5.11: Escena con un NeRF y dos objetos de Blender

Uno de los puntos claves para hacer la composición de escenas es conocer qué objetos están delante y cuáles detrás para poder calcular las oclusiones. Para ello, se ha utilizado un tipo de dato ampliamente usado en gráficos por ordenador. Se trata de ***z-buffer***, empleado para representar la información de profundidad de una escena tridimensional desde un punto de vista concreto. Su estructura es similar a una cuadrícula, con las mismas dimensiones que la imagen. En cada posición de dicha cuadrícula se guarda un valor que representa la distancia o profundidad de un objeto en relación con la cámara en ese píxel específico. Este valor indica qué tan lejos o cerca está el objeto de la cámara en cada punto de la imagen.

Por lo tanto, para poder hacer la composición es imprescindible tener los siguientes elementos:

**Un vídeo o imágenes de la escena de Blender sin que aparezca la malla del NeRF.** Para renderizar la escena sin que lo haga el Lego, hay que desactivar su procesamiento (Ilustración 5.12). Seguidamente, se escoge la configuración de la salida, es decir, el formato, el tamaño del vídeo o imágenes... Finalmente, se procesa la animación.


Ilustración 5.12: Malla del Lego con el procesamiento deshabilitado

**Z-buffer de la escena de Blender sin la malla del NeRF.** Para obtener el Z-buffer en lugar del color en la composición, es necesario realizar modificaciones en los nodos de composición en Blender. Los nodos de composición son una serie de elementos que se utilizan para manipular y combinar diferentes canales de la imagen, como el color, la transparencia, la profundidad, entre otros. Para obtener el z-buffer, lo único que hay que realizar es especificar que la salida sea la profundidad y no la imagen propiamente dicha. En la Ilustración 5.13 se puede observar como se ha realizado este cambio.

Además, para poder renderizar el z-buffer en lugar del color, es necesario utilizar un formato de archivo que sea capaz de almacenar múltiples canales y datos de alta precisión, como el formato OpenEXR. A diferencia de formatos de imagen como PNG, JPEG o BMP, que almacenan únicamente la información de color de cada píxel, OpenEXR puede guardar información adicional como el canal de profundidad. Es imprescindible que, a parte de modificar el formato de salida, hay que indicar que se almacene el z-buffer, como se muestra en la Ilustración 5.14.

**Un vídeo o imágenes del NeRF obtenidas con Nerfstudio.** La librería ofrece una orden específica para renderizar imágenes o vídeos de un NeRF. Tiene diversos paráme-

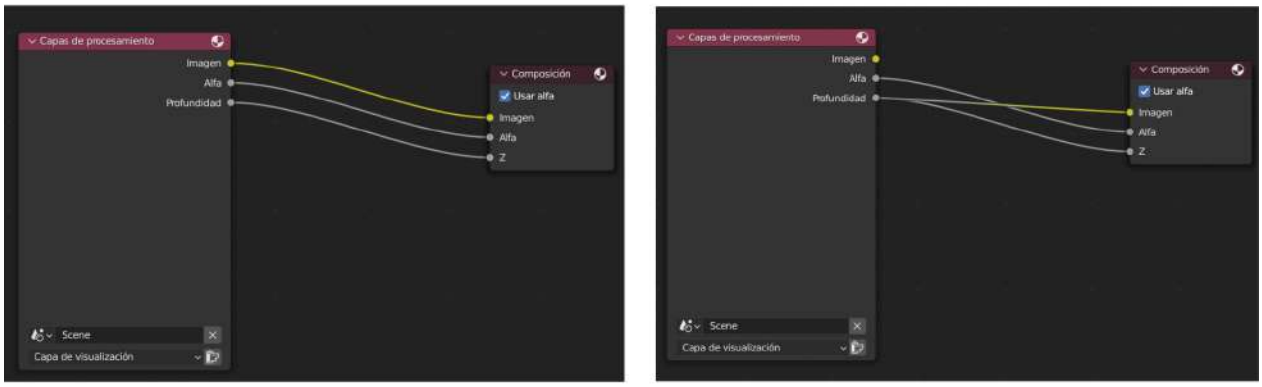


Ilustración 5.13: Nodos de composición. A la izquierda la configuración por defecto. A la derecha la configuración para renderizar la profundidad


Ilustración 5.14: Configuración de las propiedades de la salida.

tros, pero uno de los más interesantes es la trayectoria de la cámara. Esta trayectoria es un fichero JSON con una estructura concreta. En este caso, la trayectoria de la cámara tiene que ser la cámara utilizada en la animación de Blender. Para obtener dicho fichero, se utiliza la extensión de Blender de Nerfstudio comentada anteriormente.

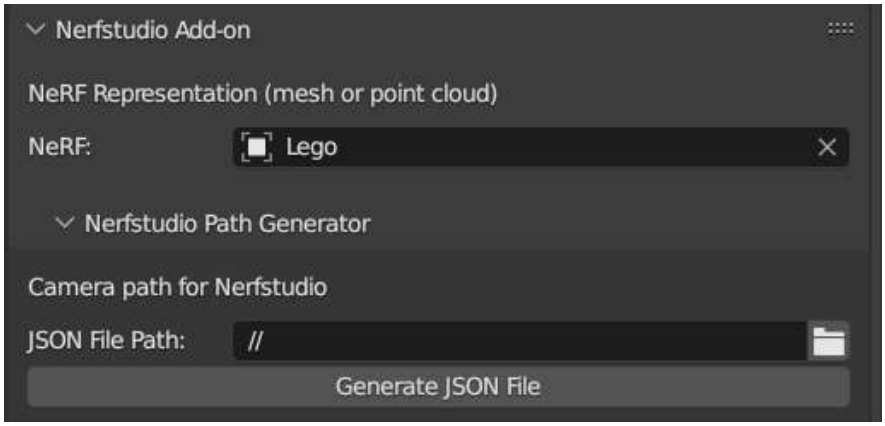


Ilustración 5.15: Generación del fichero JSON de la cámara.

Lo primero que hay que realizar es seleccionar la malla del NeRF. Seguidamente, se pulsa en el botón «Generate JSON File». Esto permite que la extensión calcule las transformaciones relativas de la malla con respecto a la cámara, tomando como referencia, el origen del objeto y genere el fichero deseado, denominado *camera_path_blender.json* en la ubicación que se haya especificado.

A continuación, se puede ir a la interfaz de línea de comandos e introducir la siguiente orden para obtener las imágenes o vídeo utilizando la trayectoria de la cámara de Blender:

```
1         ns-render {images, video} --load-config CONFIG.yml --traj
    filename --camera-path-filename camera_path_blender.json  --output-path
     renders/
2
```

**Z-buffer del NeRF.** Para obtener el z-buffer, se va a utilizar la misma orden anterior, pero especificando el parámetro *–rendered-output-names*. Este permite indicar qué se quiere renderizar. Entre los diversos valores que puede tomar, existe uno de ellos denominado *depth*, que procesa la profundidad del NeRF. Esta opción únicamente obtendrá imágenes de la profundidad, sin embargo, lo que se necesita son los valores del z-buffer. Esto significa que es necesario almacenar en algún fichero los datos con los que se calcula la profundidad.

Para poder obtener dicha información, se ha modificado el fichero de configuración de renderizado de la librería (*render.py*) para que, justo antes de obtener las imágenes, se guarde la información en un fichero NumPy (*.npy*). La orden para realizar dicha acción es la siguiente:

```
1         np.save(output_image_dir / f"{camera_idx:04d}.npy",
    output_image)
```

```
2
```

Donde *np.save* se utiliza para guardar matrices o arreglos de datos en formato binario en el disco duro, utilizando la extensión ".npy". Dicha función toma dos argumentos principales: el primero es el nombre del archivo de salida donde se guardarán los datos, y el segundo es la información que se desea almacenar.

Una vez se tiene toda la información necesaria, se puede proceder a la realización de la composición. Para ello, es necesario leer, para cada fotograma, los elementos comentados anteriormente. A continuación, se itera sobre los ficheros del z-buffer de Blender y de NeRF y se compara la profundidad de cada píxel, quedándose con la profundidad menor (objeto más cercano a la cámara). Esta comparación es la que permite conocer qué elemento está delante y cuál detrás. Seguidamente, se almacena en una nueva matriz el color del píxel más cercano. En las siguientes imágenes se muestra un ejemplo de lo descrito en esta sección.

Primero se tiene la escena en Blender con la malla del Lego y los elementos propios del motor gráfico (Ilustración 5.16).

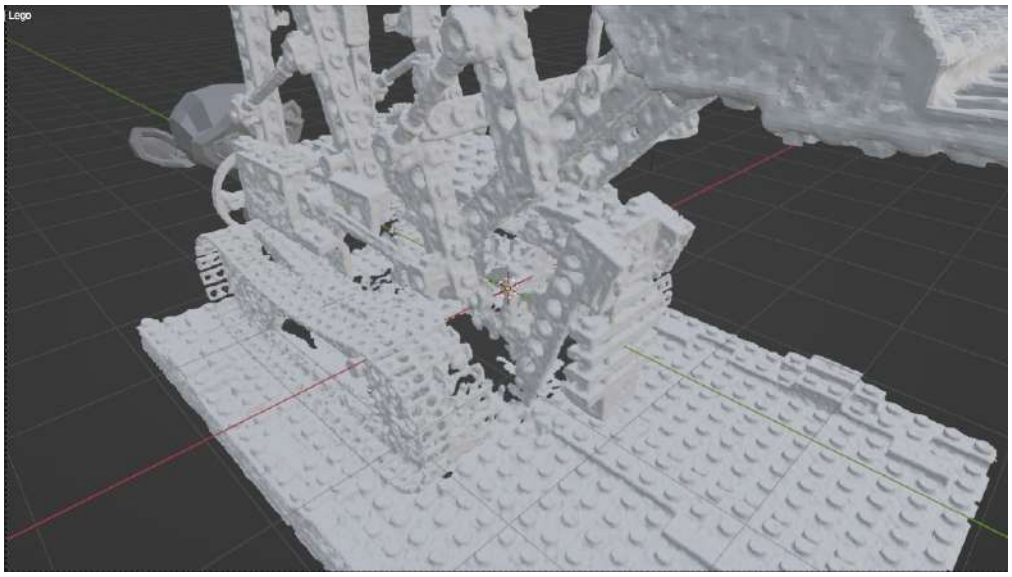


Ilustración 5.16: Escena en Blender.

A continuación, se renderiza el color y la profundidad de Blender sin procesar el Lego (Ilustración 5.17).

Ilustración 5.17: Renderizado del color de la escena de Blender sin procesar el Lego.

Seguidamente se renderiza el color y la profundidad de Lego con Nerfstudio (Ilustración 5.18).

Ilustración 5.18: Renderizado del color de Lego con Nerfstudio.

Finalmente, se realiza la composición, obteniendo un mapa de segmentación y la imagen final (Ilustraciones 5.19 y 5.20, respectivamente).

Ilustración 5.19: Mapa de segmentación de la escena compuesta. El color rosa representa que lo que está delante son los elementos de Blender. El color amarillo representa que Lego está delante.

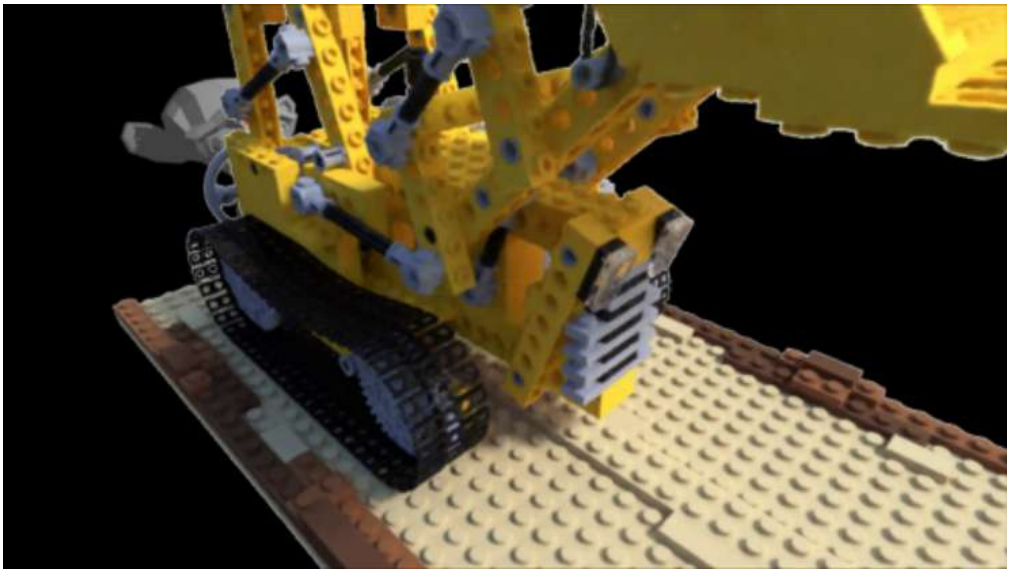

Ilustración 5.20: Resultado final de la composición.

### 5.2.2. Múltiples NeRF

Una vez realizada la composición con un NeRF y diversos elementos de Blender, se puede proceder a tener varios NeRF y volver a realizar la composición. Para esta parte, se ha tenido que entrenar otro NeRF del dataset *nerf_synthetic*. En este caso se ha utilizado el objeto denominado «Hotdog». Con las adaptaciones especificadas en secciones anteriores, se ha entrenado el modelo Nerfacto con este nuevo objeto, obteniendo los siguientes resultados:

Ilustración 5.21: NeRF del objeto Hotdog con Nerfacto.

| PSNR ↑ | SSIM ↑ | LPIPS ↓ | Tiempo (**minutos**) |
|---|---|---|---|
| 34.11 | 0.972 | 0.029 | 19 |

Cuadro 5.3: Valores medios de los resultados obtenidos tras el entrenamiento de Hotdog de 22000 iteraciones.

Como se puede apreciar en el Cuadro 5.3, los resultados obtenidos son aún más satisfactorios que los alcanzados con Lego. Los valores de PSNR y SSIM son notablemente más altos, lo que indica una mayor calidad y similitud con la imagen de referencia. Además, el tiempo de entrenamiento requerido para obtener estos resultados también es menor, lo que demuestra una mejora significativa en la eficiencia del proceso. Estos resultados refuerzan la efectividad y utilidad del enfoque utilizado en este Trabajo de Final de Grado.

El siguiente paso consiste en obtener su respectiva malla e importarla a una escena, junto con el Lego y diversos elementos de Blender, al igual que se hizo anteriormente.

Para realizar la composición de esta escena hay que realizar el mismo procedimiento seguido en la Composición de un único NeRF, con la diferencia que hay que obtener dos ficheros de trayectorias de las cámaras: uno con la malla de Lego como referencia y otro con la malla Hotdog como referencia. Este servirá para obtener las imágenes del color y los z-buffers de cada uno de ellos.

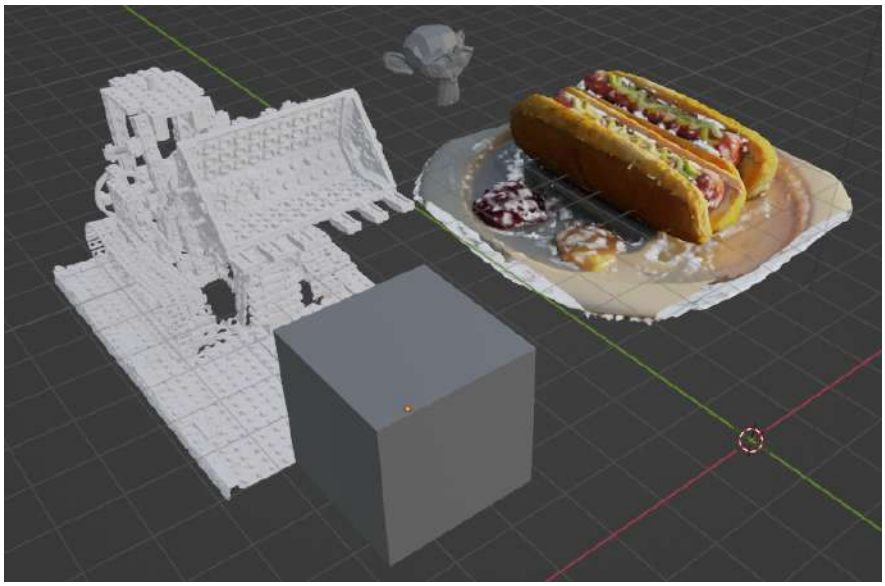

Ilustración 5.22: Escena en Blender con dos NeRFs (Lego y Hotdog) y dos objetos de Blender (cubo y mono)

Finalmente se vuele a hacer la composición como se explicó anteriormente, teniendo en cuenta que ahora se tiene un elemento más en la comparación, el cual es Hotdog. Además, se ha realizado una mejora con respecto al caso previo. Se ha conseguido también distinguir otro elemento en la composición: el fondo. Como se muestra en la Ilustración 5.27, se tiene un nuevo elemento de color azul que representa el fondo. Esta modificación permite cambiar el color del fondo, añadir una imagen... En el ejemplo que se observa en la Ilustración 5.28, el fondo ha sido establecido de color blanco.

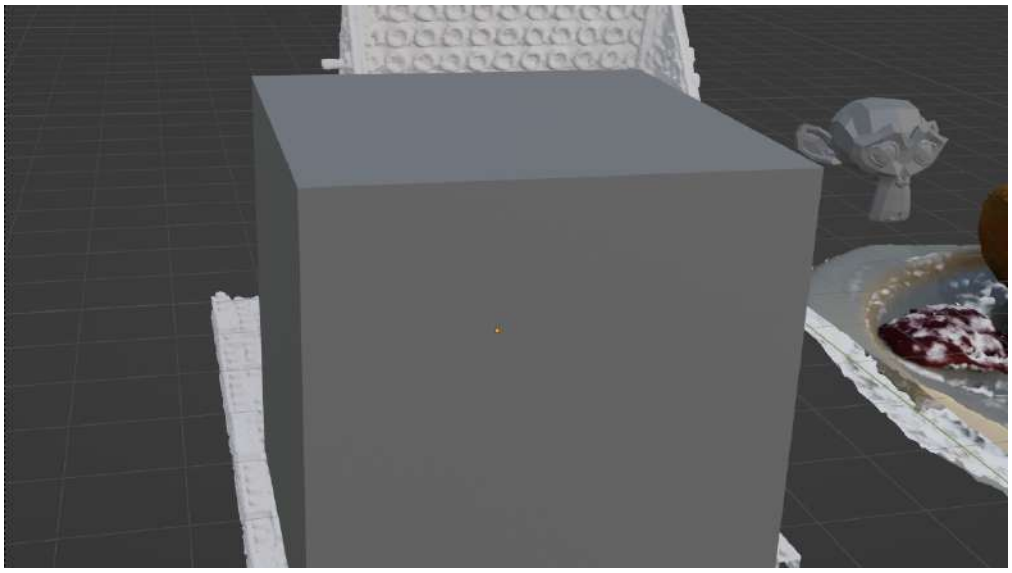

Ilustración 5.23: Escena en Blender.

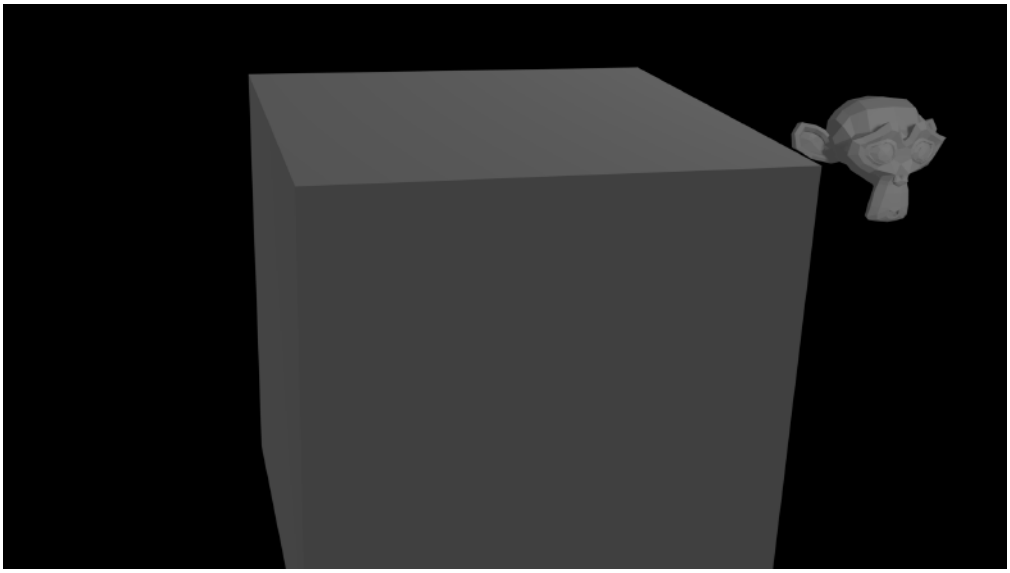

Ilustración 5.24: Renderizado del color de la Escena de Blender sin procesar Lego ni Hotdog.

Ilustración 5.25: Renderizado del color de Lego con Nerfstudio.

Ilustración 5.26: Renderizado del color de Hotdog con Nerfstudio.

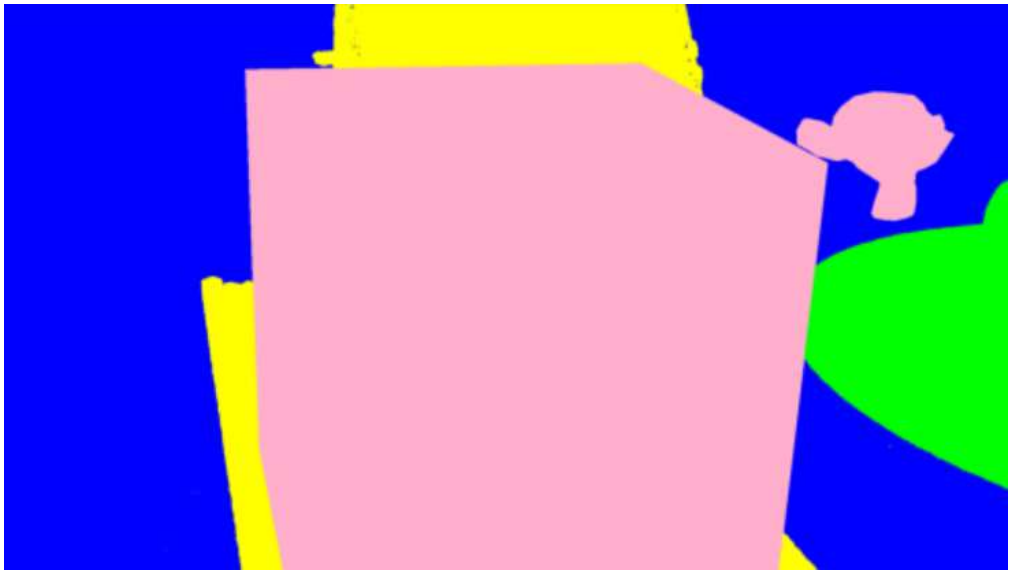

Ilustración 5.27: Mapa de segmentación de la escena compuesta. El color rosa representa que lo que está delante son los elementos de Blender. El color amarillo representa que Lego está delante. El color verde muestra que Hotdog está delante. El color azul indica fondo.

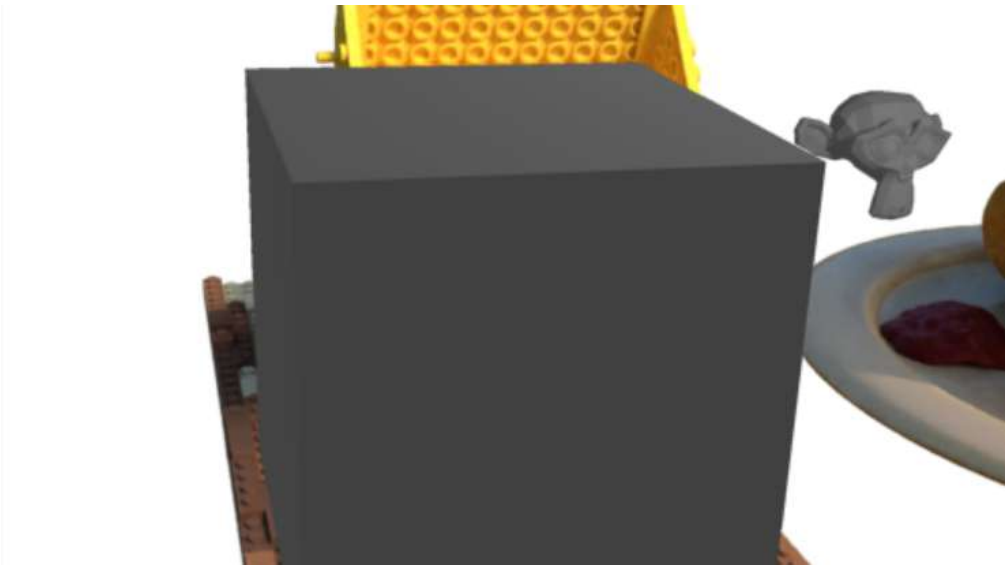

Ilustración 5.28: Resultado final de la composición.

## 5.3. Generación de una escena pseudoaleatoria y aplicación de física

La siguiente fase consiste en disponer los objetos en la escena pseudoaleatoriamente. Se trata de una escena pseudoaletoria y no aleatoria porque un ordenador es una máquina determinista y no es capaz de generar secuencias verdaderamente aleatorias.

Para esta función, el motor gráfico ofrece una función específica que permite realizarle transformaciones pseudoaleatorias a los objetos, seleccionando una semilla.

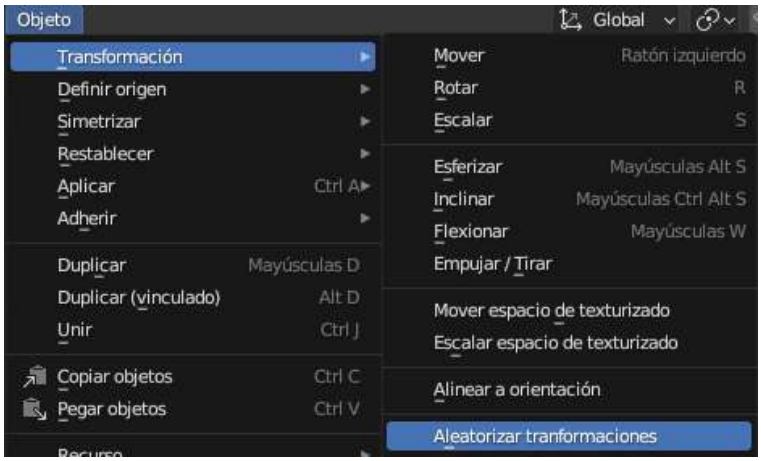


Ilustración 5.29: Opción de Aleatorizar transformaciones

Cuando se selecciona dicha opción, aparece un menú que permite seleccionar un rango de transformación tanto en la posición, rotación y escala, además de la semilla aleatoria (Ilustración 5.30). En este proyecto no se utilizará el factor de escala como una variable aleatoria.

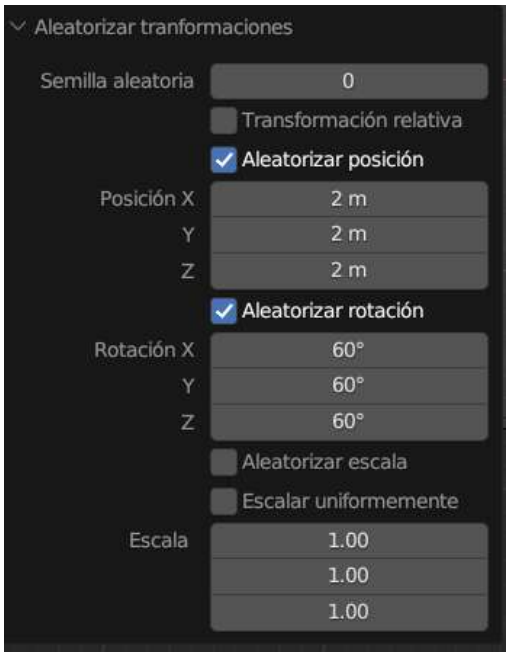


Ilustración 5.30: Menú de los posibles parámetros a aleatorizar

Al realizar modificaciones en los diferentes parámetros disponibles, es posible generar un número ilimitado de escenas con apariencia pseudoaleatoria. Al ajustar variables como la posición o la rotación, se pueden obtener resultados visualmente diferentes y únicos.

La etapa siguiente consiste en añadir la física. Para ello, se ha ideado una escena en la que los objetos caen desde su posición pseudoaleatoria inicial a una caja de cartón.

Ilustración 5.31: Configuración de la escena para agregar la física en Blender

Blender ofrece varios tipos de simulaciones físicas que pueden ser utilizadas en diferentes situaciones, como física de partículas o de fluidos. Sin embargo, la más interesante para el presente proyecto es la física de cuerpos rígidos, ya que permite simular objetos sólidos que no se deforman. Esta se basa en la dinámica de cuerpos rígidos y proporciona colisiones, gravedad y restricciones para simular interacciones realistas entre los objetos. Además, presenta una diversidad de parámetros configurables, como el tipo de cuerpo rígido (activo o pasivo), masa del objeto, entre otros.

En el caso concreto que se ha mostrado en la Ilustración 5.31, tanto las mallas de los NeRFs como el mono y el cubo han sido configurados como cuerpos rígidos activos, mientras que la caja es un cuerpo pasivo. La diferencia entre ambos es que los pasivos no se ven afectados por la gravedad, mientras que los activos sí. Esto permite que los objetos caigan dentro de la caja y colisionen con las paredes de la misma y entre ellos y que la caja se comporte como un objeto estático.

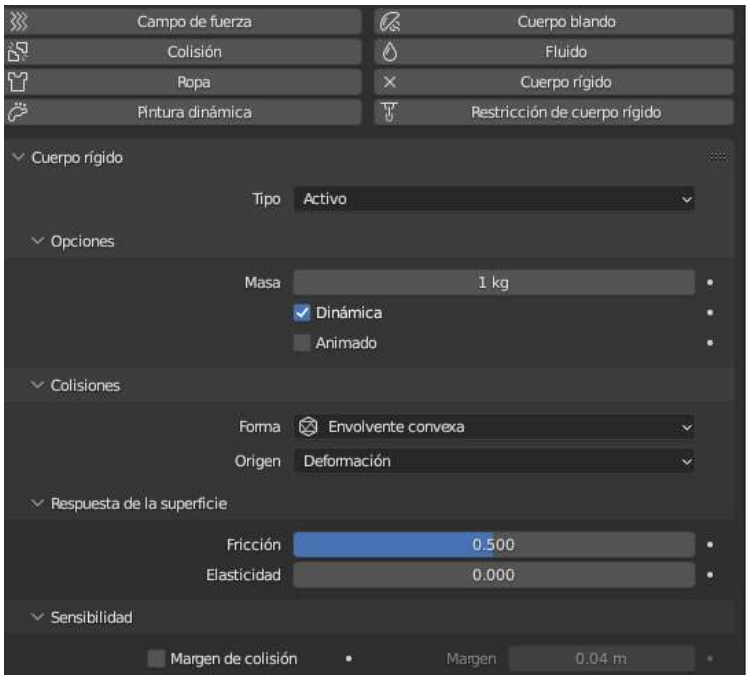


Ilustración 5.32: Configuración de un cuerpo rígido activo.

Una vez se han definido los parámetros y configuraciones deseadas, es posible generar una escena pseudoaleatoria que incluya elementos de física. Finalmente, se puede proceder a la etapa de composición vista anteriormente.


Ilustración 5.33: Simulación de la caída de los objetos en una caja con posición inicial pseudoaleatoria.

## 5.4. Captura de objetos

Una vez se han conseguido las etapas anteriores, se puede repetir el proceso con objetos capturados. Dada la naturaleza de este trabajo, los artículos que se van a utilizar son aquellos

que se puedan encontrar en un supermercado.

La primera tarea consiste en preparar un set de captura adecuado. Para ello, se ha utilizado una mesa blanca como superficie para colocar los objetos. Esta elección de color ayuda a crear un fondo neutro y a destacar los detalles y colores de los artículos. Además, se ha utilizado una pared blanca como fondo para evitar distracciones y mantener el enfoque en los objetos capturados.

Para garantizar la estabilidad y evitar movimientos no deseados durante la captura, se ha utilizado un trípode para sostener y asegurar la cámara. Esto ayuda a obtener imágenes nítidas y consistentes en todas las tomas.

El objetivo de esta configuración es crear un entorno controlado y uniforme que permita capturar los objetos de manera precisa y sin interferencias visuales. Al mantener la consistencia en el entorno de captura, se facilita el posterior procesamiento y composición de las imágenes.

Ilustración 5.34: Set de captura.

Una vez preparado el set de captura, se pueden colocar los objetos seleccionados en la mesa blanca y realizar las tomas necesarias desde diferentes ángulos y alturas. Es necesario rotar los objetos durante el proceso de captura con el fin de obtener imágenes que cubran todas las partes del objeto [EveryPoint]. La rotación de los objetos permite capturar diferentes ángulos y perspectivas, lo que proporciona una representación completa y detallada de la forma, texturas y características del objeto en cuestión.

## 5.5. Proceso de generación de datos sintéticos en escena pseudoaleatoria con los modelos NeRF de los objetos capturados

Una vez se ha capturado el objeto, el siguiente paso a realizar es obtener las poses de las cámaras de las imágenes. Para esta acción, Nerfstudio tiene una orden en concreto para procesar los datos para que sean compatibles con la librería.

```
1    $ ns-process-data {video,images,polycam,record3d} --data {DATA_PATH} --
   output-dir {PROCESSED_DATA_DIR}
```

Entre los diversos parámetros que tiene, hay que especificar que tipo de datos tenemos, la carpeta donde se encuentran dicha información y la carpeta donde se almacenarán los datos procesados. En este caso se tiene imágenes. La opción *images* utiliza COLMAP para obtener las poses de las cámaras, necesarias para entrenar Nerfacto.

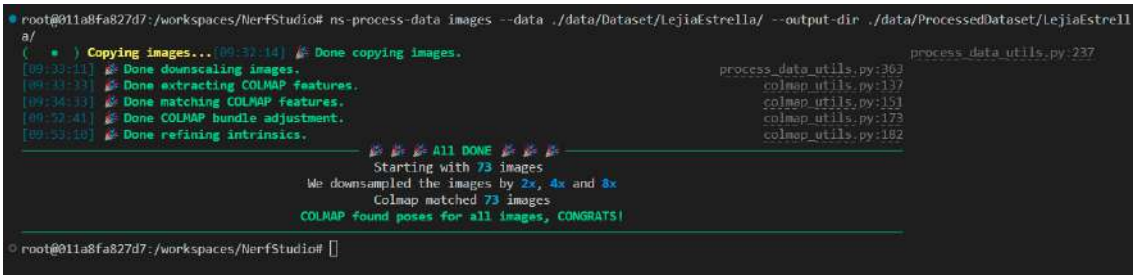


Ilustración 5.35: Salida del comando ns-process-data

Dicha orden realiza el siguiente proceso:

1. Copia las imágenes originales en el directorio donde se almacenarán los datos procesados.
2. Reduce la escala de las imágenes en un factor de 2, 4 y 8. Seguidamente, las almacena en tres carpetas distintas.
3. Busca características o puntos de interés en las imágenes. Estos puntos tienen que ser únicos dentro de la imagen, pero detectables en distintas imágenes.
4. Determina la correspondencia de los puntos en varias imágenes.
5. Optimiza y refina diversos parámetros de las poses.

Una vez finaliza el proceso, se indica en cuántas imágenes se han encontrado correspondencias. Mientras más poses se encuentra, mejor será el resultado del posterior entrenamiento.

A continuación, se elimina el fondo de las imágenes a las que COLMAP ha reducido la escala. La eliminación del fondo de las imágenes implica separar el objeto de interés del fondo . Esto se logra mediante técnicas de segmentación de imágenes. El objetivo es conservar únicamente el objeto en primer plano y eliminar cualquier información no deseada. Para esta tarea, se

ha utilizado la versión gratuita de la herramienta PhotoRoom, específicamente, la función de eliminar fondo.

Una vez se han obtenido las imágenes sin fondo, es posible repetir el proceso mencionado anteriormente para generar una escena pseudoaleatoria con física y realizar su posterior composición. Al aplicar este proceso a otros objetos y utilizar distintas semillas de inicialización de escena, se obtiene un conjunto de datos.

Sin embargo, se encontró un problema durante esta última etapa. Cuando se obtiene la malla de los objetos, su centro de masa u origen estaba desplazado y no estaba en el centro de la superficie, como se puede observar en la Ilustración 5.36. Este hecho provocaba que la física no fuese realista. Por otra parte, no se debe modificar la posición de dicho centro, ya que Nerfstudio lo utiliza como referencia para calcular las transformaciones realizada y si es desplazado, cuando se renderiza el vídeo a partir de las poses de cámara de Blender, se traslada también el NeRF.

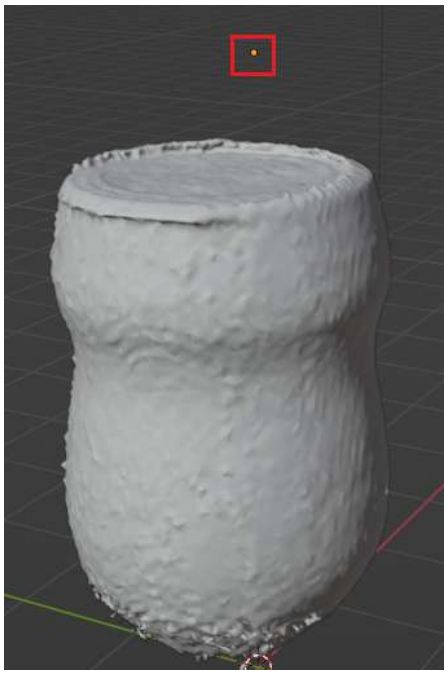

Ilustración 5.36: Malla de un bote de cacao en polvo. Resaltado en un rectángulo rojo se encuentra el centro de masa del objeto, el cual está desplazado

La solución tomada para este caso ha sido que cada objeto con el que se está simulando la física tenga dos mallas asociadas: una malla que se llamará padre y otra malla denominada hijo. El padre tiene el origen en el centro de la superficie, mientras que el hijo lo tiene en donde lo definió Nerfstudio. Además, el padre tiene la propiedad de cuerpo rígido, mientras que el hijo no. Esto permite simular la física utilizando el objeto padre, pero obtener las poses de la cámara a partir de la malla hijo.

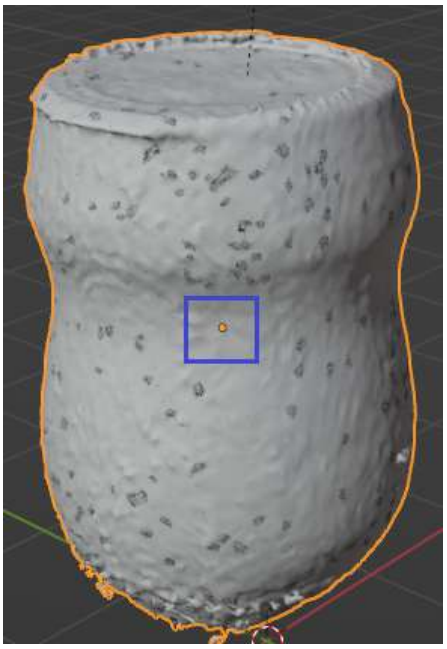

Ilustración 5.37: Malla de un bote de cacao en polvo con el origen en el centro de la superficie (malla padre). Resaltado en un rectángulo azul se encuentra el centro de masa del objeto, el cual está centrado

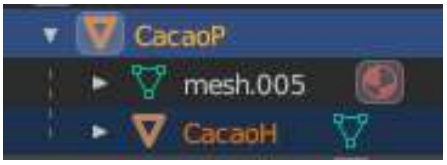


Ilustración 5.38: Asociación de la malla padre (CacaoP) y de la malla hijo (CacaoH)

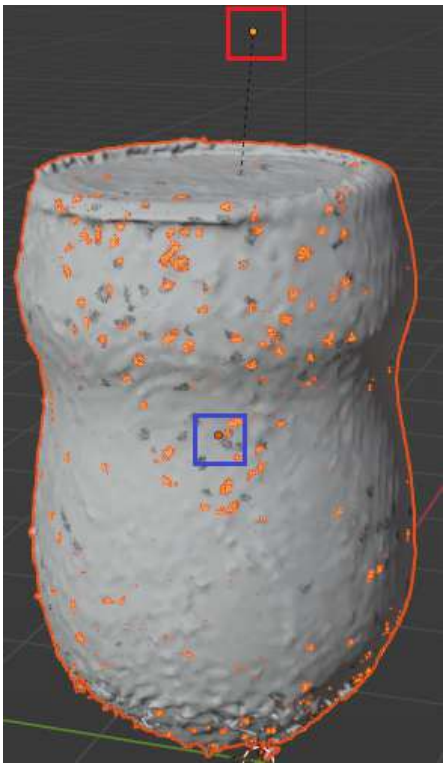

Ilustración 5.39: Resultado final de la asociación. En el rectángulo rojo se muestra el origen desplazado de la malla hijo y en el rectángulo azul se indica el origen centrado de la malla padre.

En resumen, esta solución brinda flexibilidad al separar las funciones de simulación de la física y el renderizado en dos mallas distintas asociadas a cada objeto. Esto facilita la gestión y manipulación de los objetos en la escena.

# Capítulo 6

# Generación de la base de datos de imágenes

En este capítulo se mostrarán los distintos objetos capturados y diversas escenas pseudoaleatorias realizas con dichos artículos.

Es importante asegurarse de capturar las fotografías de manera consistente y con una iluminación adecuada para obtener resultados óptimos. El objetivo de la captura exhaustiva es garantizar una cobertura completa de los objetos en términos de sus características y detalles. Al tomar imágenes desde distintas alturas y puntos de vista, se obtiene una representación integral de los artículos.

## 6.1. Objetos capturados

A continuación, se describirán los diversos objetos capturados, número de imágenes tomadas y métricas obtenidas tras el entrenamiento.

### 6.1.1. Botella de lejía

El primer artículo capturado fue una botella de lejía (Ilustración 6.1). Se realizaron un total de 485 imágenes de la misma desde diferentes alturas y puntos de vista. Una vez se han entrenado el NeRF, se obtuvo el resultado que se observa en la Ilustración 6.3.


Ilustración 6.1: Pila de imágenes de la botella de Lejía.


Ilustración 6.2: Comparación de la imagen de referencia con el NeRF obtenido. A la izquierda la imagen de referencia (*Ground truth*) y a la derecha el NeRF.


Ilustración 6.3: NeRF de la lejía obtenido tras el entrenamiento con las adaptaciones realizadas con Nerfacto.

### 6.1.2. Caja de cereales

El siguiente artículo que se capturó fue una caja de cereales. Para este artículo se tomaron 314 imágenes (Ilustración 6.4). El resultado obtenido del entrenamiento es el que se muestra en la Ilustración 6.6.


Ilustración 6.4: Pila de imágenes de la caja de cereales.

Ilustración 6.5: Comparación de la imagen de referencia con el NeRF obtenido. A la izquierda la imagen de referencia (*Ground truth*) y a la derecha el NeRF.


Ilustración 6.6: NeRF de la caja de cereales obtenido tras el entrenamiento con las adaptaciones realizadas con Nerfacto.

### 6.1.3. Lata de refresco

En este caso, fue necesario tomar 213 imágenes para poder tener una visión completa del objeto como se puede observar en la Ilustración 6.7. Se siguió el mismo procedimiento descrito anteriormente, hasta obtener el NeRF como se muestra en la Ilustración 6.9.

Ilustración 6.7: Pila de imágenes de la lata de refresco.


Ilustración 6.8: Comparación de la imagen de referencia con el NeRF obtenido. A la izquierda la imagen de referencia (*Ground truth*) y a la derecha el NeRF.


Ilustración 6.9: NeRF de la lata de refresco obtenido tras el entrenamiento con las adaptaciones realizadas con Nerfacto.

### 6.1.4. Rollo de papel

Para este objeto se tomaron 631 fotografías en ángulos y perspectivas distintas para intentar tener una amplia cobertura de información como se ve en la Ilustración 6.10. Seguidamente, se obtuvo el NeRF como se observa en la Ilustración 6.12.

Es esencial comentar que este artículo puede suponer un reto para las técnicas de fotogrametría tradicionales, debido a las transparencias que presenta el envoltorio que tiene, pero que para los modelos NeRF no resulta complejo modelar esta característica.


Ilustración 6.10: Pila de imágenes del rollo de papel.


Ilustración 6.11: Comparación de la imagen de referencia con el NeRF obtenido. A la izquierda la imagen de referencia (*Ground truth*) y a la derecha el NeRF.


Ilustración 6.12: NeRF del rollo de papel obtenido tras el entrenamiento con las adaptaciones realizadas con Nerfacto.

### 6.1.5. Cacao en polvo

A continuación, se capturó un bote de cacao en polvo como se presenta en la Ilustración 6.13. Para ello, fueron necesarias 157 imágenes. Una vez con esta información, se pudo obtener el NeRF que se muestra en la Ilustración 6.15.


Ilustración 6.13: Pila de imágenes del cacao en polvo.


Ilustración 6.14: Comparación de la imagen de referencia con el NeRF obtenido. A la izquierda la imagen de referencia (*Ground truth*) y a la derecha el NeRF.


Ilustración 6.15: NeRF del cacao en polvo obtenido tras el entrenamiento con las adaptaciones realizadas con Nerfacto.

### 6.1.6. Botella de champú

El último artículo que se capturó fue una botella de champú, como se ve en las imágenes 6.16. En esta ocasión, se realizaron 190 fotografías. Tras este paso, se procesaron las imágenes y se generó el NeRF que se observa en la Ilustración 6.18.


Ilustración 6.16: Pila de imágenes de la botella de champú.


Ilustración 6.17: Comparación de la imagen de referencia con el NeRF obtenido. A la izquierda la imagen de referencia (*Ground truth*) y a la derecha el NeRF.


Ilustración 6.18: NeRF de la botella de champú obtenido tras el entrenamiento con las adaptaciones realizadas con Nerfacto.

Una vez se tienen los NeRFs de los distintos objetos, se puede analizar los resultados obtenidos.

| Objeto | PSNR ↑ | SSIM ↑ | LPIPS ↓ | Tiempo |
|---|---|---|---|---|
| Lejía | **32.109** | **0.974** | **0.016** | 70m 26s |
| Cereales | 27.807 | 0.956 | 0.027 | 17m 46s |
| Refresco | 30.033 | 0.966 | 0.022 | 16m 6s |
| Rollo de papel | 26.357 | 0.8978 | 0.083 | 22m 54s |
| Cacao | 29.465 | 0.955 | 0.054 | 48m 20s |
| Champú | 28.076 | 0.967 | 0.045 | 86m 54s |

Cuadro 6.1: Métricas obtenidas de los NeRFs de los distintos artículos. Además, se ha destacado en negrita aquellos valores que sean los mejores para cada métrica.

Como se puede observar en el Cuadro 6.1, se ha observado que la botella de lejía ha obtenido

los mejores resultados en términos de precisión y calidad de los resultados generados. En segundo lugar se encuentra la lata de refresco, que también ha demostrado un rendimiento destacado. Sin embargo, es importante mencionar que la botella de lejía ha requerido un tiempo considerablemente mayor en el proceso de captura y generación de datos en comparación con la lata de refresco, que ha sido más rápido.

A pesar de las diferencias en los tiempos de captura, en general, los resultados obtenidos son altamente satisfactorios y se equiparan el estado del arte actual. Esto implica que se han logrado avances significativos en términos de precisión y calidad. Estos resultados demuestran el potencial y la efectividad del enfoque de este proyecto.

La equiparación con el estado del arte actual es un logro notable, ya que indica que los métodos y técnicas utilizados en este proyecto son altamente efectivos y están a la vanguardia de la investigación en este campo.

## 6.2. Escenas

Una vez se tienen los NeRFs de los objetos anteriores, se han obtenido sus mallas y se han generado diversas escenas pseudoaleatorias como se comentó anteriormente.

### 6.2.1. Escena 1

En esta primera escena, se han utilizado 2 botellas de lejía, 2 cajas de cereales, 1 rollo de papel, 1 lata de refresco, 1 botella de champú y 1 bote de cacao. En la Ilustración 6.19 se pueden observar algunos de los frames de la animación.


Ilustración 6.19: Secuencia de la primera escena

### 6.2.2. Escena 2

Esta escena contiene los mismos elementos que la anterior, pero se ha utilizado otra semilla para obtener otra escena distinta. Esta escena es la que se puede ver en la Ilustración 6.20.

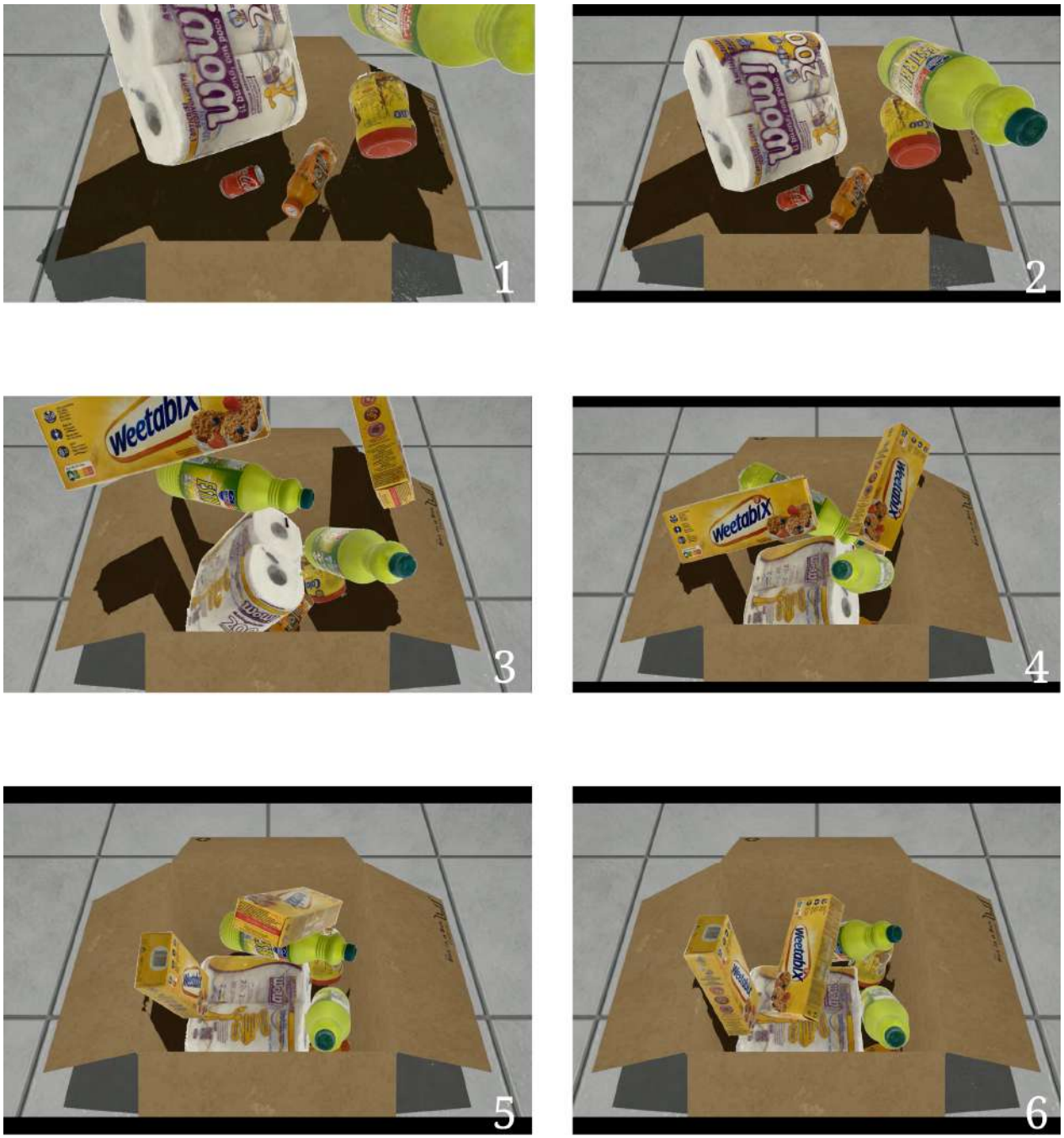


Ilustración 6.20: Secuencia de la segunda escena

### 6.2.3. Escena 3

Al igual que en los casos anteriores, se han dispuesto 2 botellas de lejía, 2 cajas de cereales, 1 rollo de papel, 1 lata de refresco, 1 botella de champú y 1 bote de cacao de manera pseudoaleatoria en la escena, como es posible observar en la Ilustración 6.21 .


Ilustración 6.21: Secuencia de la tercera escena

### 6.2.4. Escena 4

De nuevo, se ha modificado la semilla para obtener otra escena distinta con los mismos objetos que los comentados anteriormente.

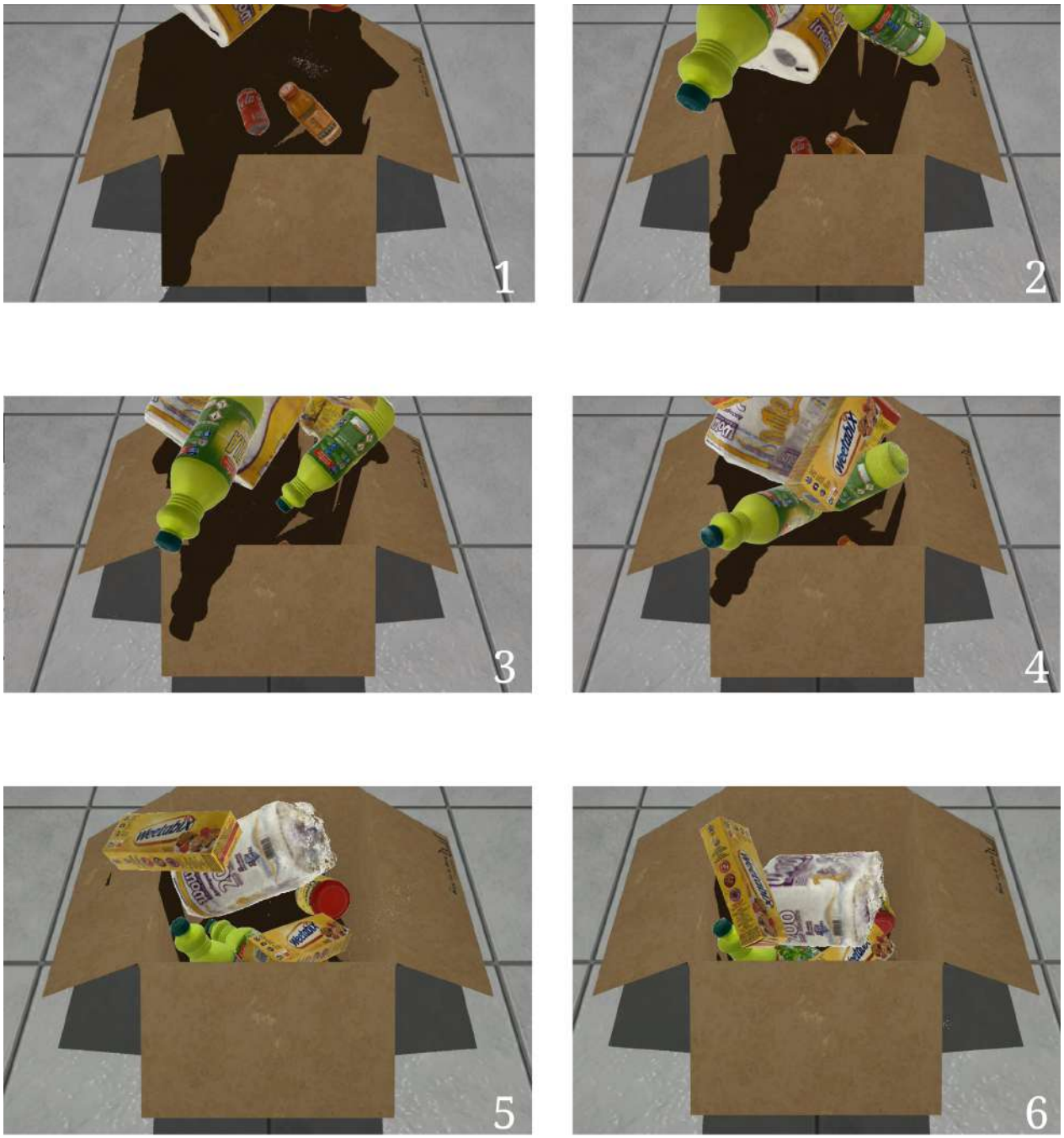


Ilustración 6.22: Secuencia de la cuarta escena

### 6.2.5. Escena 5

Una vez más, se ha modificado la semilla con el propósito de generar una escena diferente que contiene los mismos objetos mencionados previamente.

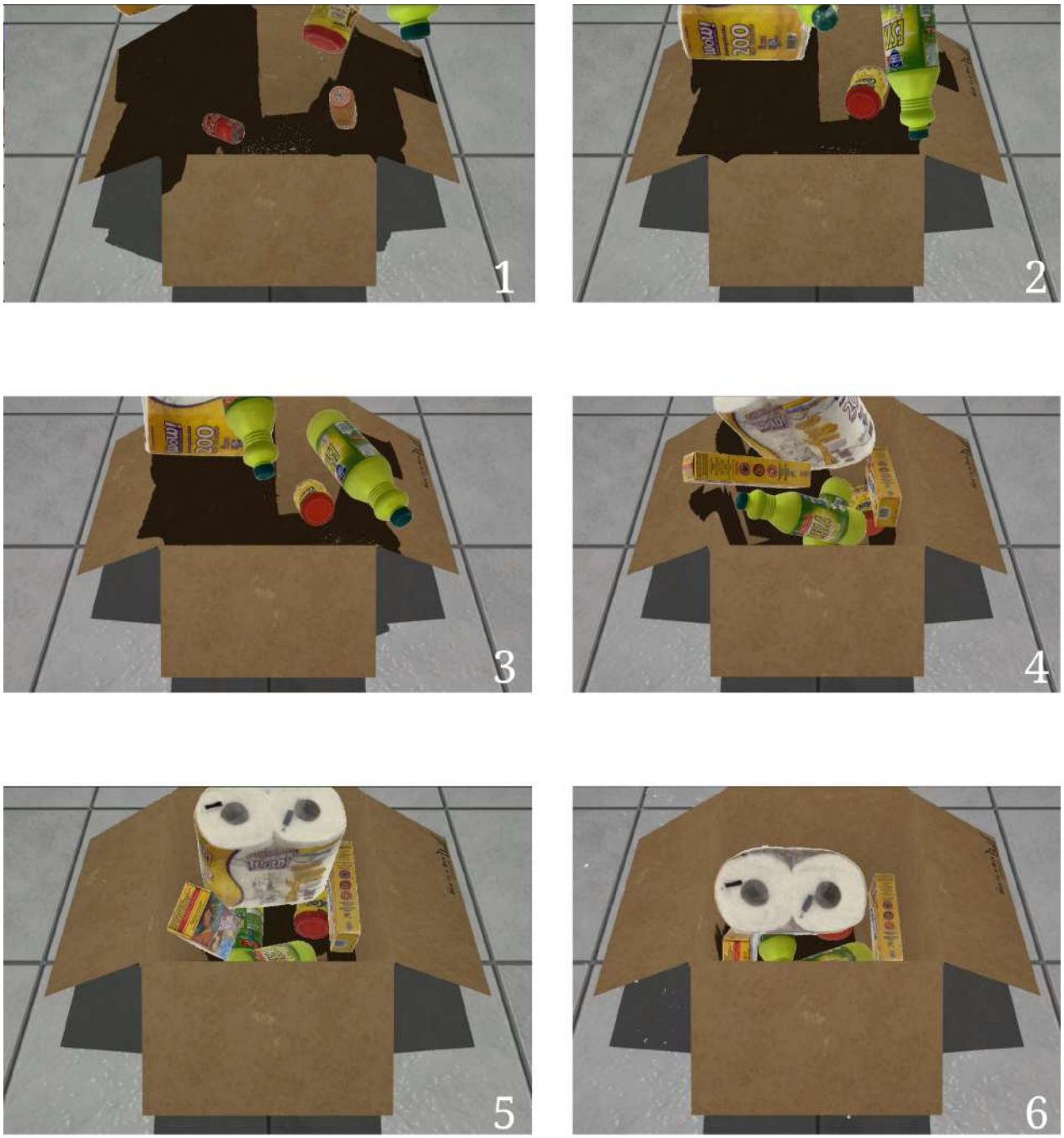


Ilustración 6.23: Secuencia de la quinta escena

### 6.2.6. Escena 6

En esta ocasión, se ha modificado la escena de manera que en vez de tener 2 botellas de lejía se tiene 1 y, en lugar de 1 botella de champú, se tienen 2.


Ilustración 6.24: Secuencia de la sexta escena

### 6.2.7. Escena 7

A partir de los objetos utilizados en la escena 6, se ha generado una nueva escena cambiando la semilla para la aplicación de las transformaciones pseudoaleatorias.


Ilustración 6.25: Secuencia de la séptima escena

### 6.2.8. Escena 8

Al igual que se especificó anteriormente, se ha modificado la semilla de inicialización para generar una nueva escena.


Ilustración 6.26: Secuencia de la octava escena

### 6.2.9. Escena 9

Siguiendo lo mencionado previamente, se ha procedido a alterar la semilla de inicio con el fin de generar una escena completamente nueva.


Ilustración 6.27: Secuencia de la novena escena

### 6.2.10. Escena 10

Finalmente, como ya se ha comentado en los apartados anteriores, se ha cambiado la semilla utilizada para la creación de una escena distinta a las anteriores.


Ilustración 6.28: Secuencia de la décima escena

# Capítulo 7

# Aportaciones

Este Trabajo de Final de Grado puede tener diversas contribuciones a nivel socio-económico, técnico y científico. En primer lugar, el uso de modelos como NeRF permite generar datos sintéticos realistas, capaces de simular la apariencia y comportamiento de los objetos. Esta característica permite comprobar el estado de las mercancías, ya que se puede simular distintas condiciones y escenarios, como puede ser de artículos dañados, desordenados o en distintas posiciones.

Además, con la técnica utilizada para generar diversas escenas pseudoaleatorias, se puede obtener un gran volumen de datos en menos tiempo y reduciendo el coste, ya que no es necesario tener una amplio número de objetos para poder generar el dataset ni requiere gran cantidad de recursos.

Por otra parte, el dataset generado en este trabajo puede ser utilizado en la investigación para el entrenamiento de algoritmos y modelos de inteligencia artificial, tales como la segmentación semántica, detección de objetos, análisis de datos, entre otras. Esto podría tener un impacto positivo en la automatización de tareas de detección y optimización de la mercancía de los comercios, pudiendo, además, crear y simular experiencias de compra.

Finalmente, este dataset podría ser utilizado para entrenar y evaluar sistemas robóticos de *pick and place* o *pick and grab* para tareas de identificación, selección y agarre de objetos en un entorno como el simulado en este proyecto, ya que se tienen objetos variados, dispuestos de diversas maneras, en distintas posiciones y alturas. Esta aplicación es de gran importancia, debido a que este tipo de robots pueden ocasionar daños a las personas y mercancías con las que trabajan si no están bien entrenados (si no son precisos o existen errores en el reconocimiento de objetos), dada la fuerza de agarre y alta velocidad a la que trabajan estos sistemas.

# Capítulo 8

# Conclusiones y trabajo futuro

En definitiva, este proyecto ha reforzado los conocimientos básicos que se tenía sobre la inteligencia artificial y ampliando aquellos relacionados con la visión por computador y en las redes neuronales basadas en campos de radiancia o NeRFs. Como se ha podido comprobar a lo largo de este proyecto, un elemento importante para que una red neuronal ejecute las tareas correctamente o genere los resultados deseados, son los datos de entrada que se le proporcionan para entrenar y evaluar. Es esta razón que este trabajo puede resultar muy útil en diversas aplicaciones, gracias a la base de datos de imágenes generado.

Además, este trabajo ha abordado el problema de la generación de volúmenes de objetos que tienen características que pueden ser complejas de modelar para la fotogrametría, como las transparencias, obteniendo resultados satisfactorios y haciendo posible la generación de un conjunto de datos que contenga este tipo de objetos.

Por otra parte, se ha puesto en práctica técnicas que forman parte del estado del arte y que están revolucionando el campo de la captura volumétrica, realizando un pequeño estudio de los distintos modelos existentes y las diferentes aplicaciones que tienen, escogiendo, así, la arquitectura que más se ajusta a este proyecto, como es Nerfacto.

Si bien es cierto que los resultados obtenidos son satisfactorios, sigue existiendo lugar para la mejora de los mismos. Como se ha podido comprobar, aunque se haya modificado el modelo utilizado, sigue existiendo cierto ruido en los NeRFs en forma de puntos blancos alrededor de ellos. Esto se debe a ciertas limitaciones del modelo o alguna deficiencia a la hora de la captura de imágenes del artículo.

Para poder solventar esta problemática, se están desarrollando nuevos métodos y artefactos que reducen o eliminan por completo los ruidos de los NeRFs. Este hecho podría mejorar la calidad de los resultados obtenidos y que sería una nueva tarea a aplicar en trabajos futuros.

Asimismo, otra posible propuesta para futuros trabajos puede ser la modificación de los NeRFs. En la actualidad, las herramientas basadas en modelos generativos están revolucionando la manera de trabajar y de crear arte, como ChatGPT o Stable Diffussion. Estas podrían ser utilizadas para cambiar el color de los NeRFs, deformación de su geometría, cambios en la iluminación de la escena, entre otras varias ampliaciones.

# Bibliografía


[1] Barron, J. T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., and Srinivasan, P. P. (2021). Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields.

[2] Barron, J. T., Mildenhall, B., Verbin, D., Srinivasan, P. P., and Hedman, P. (2022). Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[3] Blender (2023). blender.org - home of the blender project - free and open 3d creation software. Accedido: 04-05-2023.

[4] Conferences, S. (2022). Siggraph 2022 technical papers awards: Best papers and honorable mentions - acm siggraph blog.

[5] de Ingeniería Informática de la Universidad de Las Palmas de Gran Canaria, E. (2019). Memoria del plan de estudios del grado en ingeniería informática. Accedido: 24-05-2023.

[EveryPoint] EveryPoint. How to capture images for 3d reconstruction.

[7] Lindzon, J. (2022). Nvidia instant nerf: The 200 best inventions of 2022 — time. Accedido: 05-05-2023.

[8] Manzaneque, F. R. (2023). Revolutionizing virtual production: How neural radiance fields will supercharge production pipelines — neural radiance fields. Accedido: 05-05-2023.

[9] Martin-Brualla, R., Radwan, N., Sajjadi, M. S. M., Barron, J. T., Dosovitskiy, A., and Duckworth, D. (2021). Nerf in the wild: Neural radiance fields for unconstrained photo collections.

[10] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020). Nerf: Representing scenes as neural radiance fields for view synthesis. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12346 LNCS:405–421.

[11] Müller, T., Nvidia, S., Evans, A., Schied, C., and Keller, A. . (2022). Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41:102.

[12] Nilsson, J. and NVIDIA, T. A.-M. (2020). Understanding ssim.

[13] NumPy (2023). Numpy. Accedido: 04-05-2023.

[14] OpenCV (2023). Opencv. Accedido: 04-05-2023.

[15] PhotoRoom (2023). Photoroom - remove background and create product pictures. Accedido: 04-05-2023.

[16] Python (2023). Welcome to python.org. Accedido: 04-05-2023.

[17] PyTorch (2023). Pytorch. Accedido: 04-05-2023.

[18] Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F. A., Bengio, Y., and Courville, A. (2018). On the spectral bias of neural networks. *ICML 2019*.

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

[20] Schönberger, J. L., Price, T., Sattler, T., Frahm, J.-M., and Pollefeys, M. (2016a). A vote-and-verify strategy for fast spatial verification in image retrieval. In *Asian Conference on Computer Vision (ACCV)*.

[21] Schönberger, J. L., Zheng, E., Pollefeys, M., and Frahm, J.-M. (2016b). Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*.

[22] Tancik, M., Srinivasan, P. P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J. T., and Ng, R. (2020). Fourier features let networks learn high frequency functions in low dimensional domains. *CoRR*.

[23] Tancik, M., Weber, E., Ng, E., Li, R., Yi, B., Kerr, J., Wang, T., Kristoffersen, A., Austin, J., Salahi, K., Ahuja, A., McAllister, D., and Kanazawa, A. (2023). Nerfstudio: A modular framework for neural radiance field development.

[24] Tancik*, M., Weber*, E., Ng*, E., Li, R., Yi, B., Wang, T., Kristoffersen, A., Austin, J., Salahi, K., Ahuja, A., McAllister, D., and Kanazawa, A. (2022). Nerfstudio: A framework for neural radiance field development.

[25] Verbin, D., Hedman, P., Mildenhall, B., Zickler, T., Barron, J. T., and Srinivasan, P. P. (2021). Ref-nerf: Structured view-dependent appearance for neural radiance fields.

[26] Wang, Z., Wu, S., Xie, W., Chen, M., and Prisacariu, V. A. (2022). Nerf–: Neural radiance fields without known camera parameters.

[27] Weights&Biases (2023). Weights & biases – developer tools for ml. Accedido: 04-05-2023.

[28] Xie, Y., Takikawa, T., Saito, S., Litany, O., Yan, S., Khan, N., Tombari, F., Tompkin, J., Sitzmann, V., and Sridhar, S. (2022). Neural fields in visual computing and beyond. *Computer Graphics Forum*.

[29] Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. (2018). The unreasonable effectiveness of deep features as a perceptual metric.

[30] Zuza, M. (2018). Photogrammetry - 3d scanning with just your phone/camera - original prusa 3d printers. Accedido: 03-05-2023.