UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria
Corso di Laurea Magistrale in
Ingegneria Informatica e dell'Automazione

Generazione di modelli 3D attraverso Neural Surface Reconstruction

Generation of 3D models with Neural Surface Reconstruction

Relatore: Chia.mo
Prof. Primo Zingaretti

Tesi di Laurea di:
David Ceka

Anno Accademico 2022-2023

Università Politecnica delle Marche
Facoltà di Ingegneria
Corso di Laurea Magistrale in
Ingegneria Informatica e dell'Automazione
Via Brecce Bianche – 60131 Ancona (AN), Italy

Alla mia famiglia. Mamma, Babbo, Mervin, Jessie.

Sommario

Sin dalla nascita della computer graphics, è stato intento dell'uomo riuscire a creare una modellazione fotorealistica del mondo reale all'interno del calcolatore. La problematica della costruzione di modelli 3D a partire dal mondo reale ha sempre posato una grande sfida, data proprio dall'elevata difficoltà di questo task, in quanto la costruzione di un modello fedele morfologicamente e fotorealisticamente necessità di elevatissime risorse, ed allo stato dell'arte negli anni passati gli strumenti presenti non erano in grado di fornire modelli sempre validi. Tecniche come Structured Light Scanner, Fotogrammetria e Stereo Multi-View, sebbene avanzate per il loro tempo, spesso incontravano limitazioni nella gestione della complessità e nella cattura di dettagli fini e intricati delle superfici. L'introduzione di approcci neurali nella ricostruzione 3D ha rivoluzionato questa prospettiva, aprendo nuovi orizzonti nella generazione di modelli dettagliati e fotorealistici. Questi approcci, difatti, hanno portato ad uno sviluppo dei campi di radianza neurale e dei metodi di ricostruzione neurale delle superfici che negli ultimi anni hanno raggiunto punti di altissimo livello nell'ambito della rappresentazione di scene 3D prese dal mondo reale all'interno del calcolatore. Metodi NeRF, metodi SDF, e metodi basati su Gaussian Splatting si contendono i primati per queste ricostruzioni, ma allo stato dell'arte c'è la necessità di avere una comparazione completa di tutti e tre gli approcci per evidenziarne punti di forza, debolezza e valutarne la corretta destinazione di utilizzo. A tal fine, questo lavoro di tesi presenta uno studio approfondito di queste architetture attraverso l'utilizzo di diverse metodologie e dataset che racchiudano i più importanti scenari di applicazione. Si è riportato lo stato dell'arte di ognuna di queste tecnologie focalizzando su qualità, tempi di allenamento e risorse utilizzate, basandosi sui più recenti sviluppi in ambito di ricerca. E' obiettivo di questo studio creare una linea guida attraverso l'analisi dei modelli risultanti utilizzando metriche e proprietà di allenamento delle varie reti che compongono le strutture software che generano questi strumenti per facilitare una scelta consapevole di quale applicare ai propri dataset e alle proprie condizioni di utilizzo. Per raggiungere gli scopi preposti, sono stati adattati gli approcci sopracitati, valutando: metriche oggettive (PSNR), metriche che cercano di replicare la percezione umana (SSIM, LPIPS) e quantità di risorse impiegate a parità di resa grafica. Una volta estratto il modello 3D(Mesh) ne è stata eseguita una ripulitura per permetterne una più facile fruizione da software di editing, ma non solo, anche per il CAVE. Ultimo step di questi esperimenti difatti, è l'importazione sull'ambiente di realtà virtuale CAVE per la manipolazione della mesh. Infine, lo studio viene completato riportando risultati legati alle varie soluzioni

riportando che: NeRF eccelle nell'esplorazione veloce di scene 3D con una resa visiva elevata, ma mostra limitazioni nei dettagli critici della mesh; SDF si distingue per la ricostruzione di mesh ad alta fedeltà, ma a discapito di tempi di computazione più elevati; Gaussian Splatting rappresenta un compromesso versatile, offrendo una buona qualità della mesh con tempi di computazione moderati. In termini di tempi di allenamento, Nerfacto si contraddistingue per la velocità, richiedendo solo circa 30 minuti, ma produce mesh con un numero relativamente basso di vertici. Bakedangelo, con un tempo di allenamento più lungo di circa 24 ore, genera mesh dettagliate ad alto numero di vertici. Gaussian Splatting si posiziona come un equilibrio, con tempi di allenamento di circa 1 ora e un numero intermedio di vertici. Questi risultati forniscono un'ampia comprensione delle prestazioni di ciascun metodo, consentendo una scelta informata in base alle esigenze specifiche dell'applicazione, restringendo il campo consigliato di NeRF a scenari che richiedono esplorazione veloce con una buona resa visiva; SDF alle applicazioni che richiedono dettagli precisi anche se a costo di tempi di computazione più lunghi; Gaussian Splatting si configura come una scelta versatile per situazioni che richiedono un bilanciamento tra dettaglio e efficienza computazionale. Quest'analisi, dunque, fornisce un quadro completo e dettagliato delle performance di questi metodi di generazione mesh, contribuendo ad orientare la scelta del metodo più appropriato per diverse esigenze di applicazione.

Abstract

Since the birth of computer graphics, it has been man's intent to succeed in creating photorealistic modeling of the real world within the computer. The problem of constructing 3D models from the real world has always posed a great challenge, given precisely by the high difficulty of this task, as the construction of a morphologically and photorealistically faithful model needs very high resources, and at the state of the art in the past years the present tools were not able to provide consistently good models. Techniques such as Structured Light Scanner, Photogrammetry and Stereo Multi-View, although advanced for their time, often encountered limitations in handling complexity and capturing fine and intricate surface details. The introduction of neural approaches in 3D reconstruction has revolutionized this perspective, opening new horizons in the generation of detailed, photorealistic models. These approaches, in fact, have led to a development of neural radiance fields and neural surface reconstruction methods that in recent years have reached high points in the field of representing 3D scenes taken from the real world within the computer. NeRF methods, SDF methods, and Gaussian Splatting-based methods vie for primacy for these reconstructions, but at the state of the art there is a need to have a comprehensive comparison of all three approaches to highlight their strengths, weaknesses, and evaluate their proper intended use. To this end, this thesis work presents an in-depth study of these architectures through the use of different methodologies and datasets encapsulating the most important application scenarios. The state of the art of each of these technologies has been reported focusing on quality, training time and resources used, based on the latest developments in research. It is the goal of this study to create a guideline through the analysis of the resulting models using metrics and training properties of the various networks that make up the software structures that generate these tools to facilitate an informed choice of which one to apply to one's datasets and usage conditions. To achieve the intended purposes, the above approaches were adapted by evaluating: objective metrics (PSNR), metrics that attempt to replicate human perception (SSIM, LPIPS), and amount of resources used for the same graphical output. Once the 3D(Mesh) model was extracted, it was cleaned up to allow easier use by editing software, but not only that, also for CAVE. Last step of these experiments in fact, is the import on the virtual reality environment CAVE for manipulation of the mesh. Finally, the study is completed by reporting results related to the various solutions by reporting that: NeRF excels in fast 3D scene exploration with high visual rendering, but shows limitations in critical mesh details; SDF excels in high-fidelity mesh reconstruction, but at the expense of higher

computation times; Gaussian Splatting represents a versatile compromise, offering good mesh quality with moderate computation times. In terms of training time, Nerfacto stands out for speed, taking only about 30 minutes, but producing meshes with relatively few vertices. Bakedangelo, with a longer training time of about 24 hours, generates detailed meshes with a high number of vertices. Gaussian Splatting is positioned as an equilibrium, with training times of about 1 hour and an intermediate number of vertices. These results provide a broad understanding of the performance of each method, allowing an informed choice based on the specific needs of the application, narrowing the recommended range of NeRF to scenarios that require fast exploration with good visual performance; SDF to applications that require precise detail even if at the cost of longer computation times; Gaussian Splatting emerges as a versatile choice for situations that require a balance between detail and computational efficiency. This analysis, therefore, provides a comprehensive and detailed picture of the performance of these mesh generation methods, helping to guide the choice of the most appropriate method for different application needs.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduction

1.1 Contesto

La Computer Graphics, un ramo dell'informatica dedicato alla manipolazione di immagini e video attraverso calcolatori digitali, si è evoluta notevolmente negli ultimi anni. Uno degli obiettivi principali è la sintesi di immagini fotorealistiche, spinta dall'esigenza di rappresentare scenari digitali in modo sempre più fedele alla realtà. Tradizionalmente, algoritmi di rendering come rasterizzazione e ray tracing sono stati impiegati per ottenere immagini sintetiche a partire da rappresentazioni matematiche della geometria e dei materiali di una scena. Tuttavia, il design digitale di scene complesse richiede tempo ed esperienza umana. In risposta a questa sfida, sono emersi processi di image-based modeling basati sulla differenza tra la rappresentazione digitale di una scena e la sua controparte reale.

Vista l'evoluzione esponenziale della potenza di calcolo degli ultimi anni, sono nate numerose tecnologie che permettono di applicare tecniche di Machine Learning e Deep Learning alla tradizionale Computer Graphics, al fine di ottenere dei risultati quanto più fotorealistici possibile. In particolare, tecniche come i Neural Radiance Fields(NeRF) hanno attirato un grande numero di ricercatori del campo per il loro sviluppo, portando ad approcci sempre più nuovi e più evoluti. I NeRF cercano attraverso un procedimento di apprendimento automatico di ricostruire una scena 3D a partire da una scena del mondo reale, valutandone le differenze e progressivamente avvicinando il modello alla realtà.

La continua espansione dei NeRF ha portato anche ad esigenze di ricostruire superfici in modo molto dettagliato, non solo per la fruizione visiva (texturizzata) ma anche dal punto di vista della mesh risultante, per l'utilizzo in altri ambiti come videogames, o CAVE. Da qui nascono gli approcci Signed Distance Function (SDF), basati su una particolare funzione che assegna ad ogni punto dello spazio una distanza dalla superficie, permettendo quindi una ricostruzione più fedele delle superfici. Questo processo prende il nome di Neural Surface Rendering, e prioritizza la qualità rispetto al tempo di esecuzione.

Infine, nascono approcci rivali ai NeRF basati su concetti totalmente diversi, quali i 3D Gaussian Splatting, che ricostruiscono lo spazio attraverso l'affinamento continuo di gaussiane fino ad avere un oggetto molto dettagliato composto di tante gaussiane anisotrope.

1.2 Obiettivi e contributi principali

Vista la grande quantità di approcci sopraelencati, è intenzione di questo lavoro di tesi aiutare chiunque si affacci a questo mondo a guidare quale approccio sia il migliore nel proprio caso specifico. Vista l'assenza in letteratura di una comparazione completa di questi tre approcci insieme (NeRF, SDF, Gaussian Splatting), sono state effettuate sperimentazioni in diversi scenari di ciascun approccio, andando a confrontare proprietà e metriche, fornendo quindi un quadro completo di come si comportano questi approcci su vari dataset, indicando quali sono gli usi consigliati per ciascuno di essi.

Questi esperimenti verranno eseguiti sia su dataset già presenti in letteratura, sia su dataset esterni. Su ispirazione di un lavoro di tesi precedente riguardante la realizzazione di una pipeline per il Cultural Heritage[1], saranno utilizzati gli stessi dataset per avere un metro di paragone tra i risultati ottenuti dagli approcci NeRF classici con gli approcci SDF e gli approcci Gaussian Splatting. I dataset in questione sono ricavati da panorami e monumenti presenti nella regione Marche: tra questi si hanno sia video che immagini, opportunamente acquisiti da un drone. Come sottolineato nell'articolo sopraelencato, questo tipo di lavoro è importante nel mantenimento del patrimonio culturale, e gli approcci di Neural Surface Rendering si vedranno protagonisti in questo task.

Infine, si utilizzeranno gli elementi ricavati dai modelli sopraelencati per l'importazione su CAVE, un ambiente di realtà virtuale permette di interagire con scene tridimensionali attraverso l'utilizzo di tre telecamere dotate di tecnologia stereoscopica. Degli occhiali posizionati sulla testa dell'utente ed un telecomando detto "flystick"

permettono una facile esplorazione della scena, creando il connubio definitivo tra immersività e fotorealismo della scena.

1.3 Struttura della Tesi

La seguente introduzione sarà seguita da una presentazione dello stato dell'arte riguardante la generazione di campi di radianza neurale e l'applicazione di Neural Surface Rendering. Questi argomenti necessitano dell'introduzione di alcuni concetti alle fondamenta di essi: alcune nozioni riguardanti la computer grafica in generale, l'apprendimento automatico (Machine Learning) ed il Deep Learning (alla base delle reti neurali).

Si parlerà quindi delle tecnologie utilizzate per ciascun approccio (NeRF, SDF, Gaussian Splatting) in modo tale da avere un quadro completo di tutti quelli che si hanno a disposizione. Si discuteranno i metodi di convergenza e gli obiettivi di ciascuna di queste metodologie, evidenziando i loro punti di forza e le evoluzioni a partire dai primi approcci in letteratura.

Successivamente verranno illustrati gli esperimenti svolti nell'ambito di ciascuna delle reti scelte per lo svolgimento dei vari task, applicandole ad ogni dataset ed annotando i risultati in delle tabelle riassuntive, sottolineando anche eventuali miglioramenti nelle metriche utilizzate per la loro comparazione. Verranno empiricamente scelti i modelli migliori verificando sia a runtime sia ad allenamenti terminati l'andamento di questi ultimi attraverso delle funzioni di loss che definiscono il corretto svoltimento dell'operazione. Tra gli esperimenti si troveranno anche quelli con il CAVE, sottolineandone pregi, difetti e difficoltà riscontrate.

In conclusione, verrà evidenziato quello che si è potuto osservare durante la sperimentazione, e verrà fornita la baseline da cui partire per la scelta consapevole tra gli approcci elencati, ottimizzando quindi il lavoro di futuri utenti. Unitamente alle conclusioni, alcuni spunti per gli sviluppi futuri che si auspica possano essere trampolino di lancio per ulteriori ricerche in questo campo.

Capitolo 2

Stato dell'arte

Per parlare delle tecniche e degli strumenti che rappresentano l'interezza del lavoro svolto, è importante e necessario comporre un contesto preliminare generico di quelli che sono il Deep Learning ed il Machine Learning.

2.1 Machine Learning

Il Machine Learning (ML) è una branca dell'intelligenza artificiale (IA) che si occupa dello sviluppo di algoritmi e modelli che permettono ai computer di apprendere da dati senza essere esplicitamente programmati. L'obiettivo è far sì che le macchine possano migliorare le proprie prestazioni nel tempo attraverso l'esperienza.
Il processo di apprendimento automatico coinvolge l'identificazione di modelli nei dati e l'utilizzo di tali modelli per prendere decisioni o fare previsioni senza una programmazione specifica. Ciò è particolarmente utile in situazioni in cui i dati sono complessi o difficili da modellare manualmente. Esistono due tipologie principali di apprendimento e sono le seguenti:

- Supervised Learning (Apprendimento supervisionato)
- Unsupervised Learning(Apprendimento non supervisionato)

Il supervised learning consiste nell'addestrare un modello su un insieme di dati etichettati, dove sono note le risposte corrette, ovvero è nota le classe di ogni oggesso. In tal modo, confrontando volta per volta le predizioni che il modello farà per un oggetto con la sua classe effettiva, riusciremo a capire se la previsione è giusta o meno.
Nell'unsupervised learning invece non si conoscono le classi di appartenenza degli

oggetti ma si vanno a cercare i legami intrinseci tra loro. Sapere che tutti gli impiegati guadagnano X, e tutti i dirigenti guadagnano Y, se ho un nuovo dato che guadagna un valore simile ad X allora potrò dire che è un dipendente.

2.2 Deep Learning

Il Deep Learning (DL) è una sottocategoria del machine learning che si basa su reti neurali artificiali. Le reti neurali sono ispirate al funzionamento del cervello umano e sono composte da strati di neuroni artificiali, noti come unità di elaborazione.
Le reti neurali artificiali cercano di replicare il funzionamento neuronale umano, e come tale l'elemento fondamentale è il neurone: proprio come accade nei neuroni umani, questi inviano un impulso al successivo solamente se gli input ricevuti superano una certa soglia. Questo processo è chiamato "attivazione", e nei neuroni artificiali viene gestita attraverso delle "funzioni di attivazione". I neuroni artificiali, a imitazione di quelli biologici, accumulano i pesi degli input, applicano una funzione di attivazione, e se il risultato supera la soglia prestabilita, emettono un segnale di attivazione alla connessione successiva.
Si parla di Deep Learning quando nella struttura di una rete neurale sono presenti strati interni multipli interposti tra l'input e l'output. Questa configurazione permette di decomporre l'informazione che attraversa la rete in vari livelli di rappresentazione, ciascuno dei quali può essere associato a una serie di attributi, concetti e, in ultima analisi, feature. Tali concetti di alto livello emergono durante l'addestramento della rete in maniera autonoma, e sono spesso strettamente collegati al compito specifico (task) affidato alla rete. Introduciamo ora le reti neurali a percettrone multistrato, comunemente note come Multi-Layer Perceptron (MLP). Queste reti sono precorritrici di numerose architetture più avanzate adottate negli ultimi anni, e nella loro prima concezione, hanno introdotto concetti fondamentali che hanno permeato e evoluto con la ricerca in questo campo.

Rappresentano un pilastro nell'ambito della ricerca trattata, svolgendo un ruolo cruciale nella creazione di strumenti per la generazione di campi di radianza neurale. Come suggerisce il loro nome, le MLP sono architetture neurali artificiali che, come anticipato in precedenza, sono costituite da diversi strati: il primo strato comprende neuroni di input, i quali ricevono valori iniziali di attivazione dai dati a disposizione. Seguono uno o più strati di neuroni nascosti (hidden layer) e, infine, uno strato di neuroni di output.
Ogni neurone nell'architettura è logicamente composto da una funzione di attivazione, che definisce il suo stato in base alla somma ponderata degli ingressi di attivazione che riceve, e da una serie di connessioni pesate in uscita. La funzione di attivazione, spesso non lineare, è un elemento chiave. Nel contesto di questo lavoro, sarà frequentemente

menzionata e adottata l'Unità di Rettificazione Lineare o ReLU, caratterizzata da una funzione di attivazione basata esclusivamente sulla parte positiva dell'argomento. Ulteriori ed importanti funzioni di attivazione sono la Sigmoide Logistica, Tangente Iperbolica (Tanh), Leaky-ReLU, Softmax.

La Rectified Linear Unit (ReLU) introduce non linearità consentendo solo valori positivi, ed è ampiamente usata per affrontare il problema della scomparsa del gradiente. La funzione Sigmoid mappa i valori nell'intervallo [0, 1], adatta per problemi di classificazione binaria. La Tangente Iperbolica (Tanh) mappa i valori in [-1, 1], utile nei casi in cui la media dei dati è lontana da zero. Leaky ReLU mitiga la "morte" dei neuroni impostando una pendenza minore per i valori negativi. Infine, Softmax converte un vettore in una distribuzione di probabilità, essenziale per la classificazione multiclasse.

Negli ultimi anni, l'adozione del deep learning ha registrato un notevole aumento, un fenomeno attribuibile principalmente alla disponibilità crescente di risorse avanzate e ad un significativo incremento della potenza di calcolo. Questo progresso tecnologico ha permesso alle reti neurali artificiali di dimensioni più ampie e complesse di essere addestrate in modo più efficiente, aprendo nuove possibilità in vari settori. Tale tendenza ha alimentato l'innovazione e il successo del deep learning, posizionandolo come un pilastro fondamentale nell'ambito dell'intelligenza artificiale e delle applicazioni tecnologiche avanzate.

2.3 Computer Graphics e Computer Vision

Computer Graphics e Computer Vision sono due facce della stessa medaglia che è la multimedialità all'interno di un calcolatore. Sono entrambe parti fondamentali di questo studio e vedremo come, operando a tandem, convergono per plasmare l'esperienza visiva digitale.

La Computer Graphics si concentra sulla generazione e manipolazione di immagini, grafiche ed animazioni al fine di creare un ambiente visivo coinvolgente e potenzialmente fotorealistico. Tutto questo in uno spazio 2D che tipicamente è il monitor del computer ma non solo, fogli stampati, fotografie, sono solo alcuni esempi dei contesti in cui opera la computer graphics, ed avremmo possibilità di parlare di alcuni sviluppi anche in mixed reality negli sviluppi futuri.

La Computer Vision, d'altro canto, si occupa dell'interpretazione e dell'analisi delle informazioni visive, cercando in qualche modo di far comprendere al calcolatore il mondo visivo umano, andando a tentare di replicare il cosiddetto Human Visual

System. E qui si riesce a capire come queste due discipline si intreccino: le informazioni che la Computer Vision riesce ad interpretare vengono poi utilizzate dalla Computer Graphics per sintetizzarle e manipolarle in modo da generare delle scene di senso compiuto per l'utente umano.
Uno dei primi problemi affrontati in questi ambiti è stata l'estrazione di informazione su oggetti presenti all'interno dell'immagine: riuscire ad affermare che all'interno di un'immagine ci sia una mela, un gatto, una persona.

Parte integrante di questo lavoro di tesi è stata la necessità di ricostruire spazi tridimensionali a partire da immagini bidimensionali. E qui si capisce come le due discipline vadano di paripasso: per rappresentare queste scene 3D in uno spazio 2D è necessario andare a capire come sono formate queste immagini, da dove sono state scattate le foto, e questo lo si fa attraverso le tecniche di Feature Detection e Feature Matching. Sostanzialmente, si vogliono trovare quante più corrispondenze possibili tra due immagini diverse della stessa scena, individuando dei punti chiave che descrivono una regione dell'immagine. Di quei punti si vuole andare a verificarne le varie caratteristiche o Feature che descrivono il suo intorno in termini di illuminazione, traslazione, scala, ed inclinazione andando ad avere come risultato un vettore descrittore di ciascuna feature. Infine, questi descrittori vengono comparati tra le varie immagini per identificare feature simili, trovando corrispondenze tra una feature nell'immagine A ed una feature nell'immagine B.

Quest'operazione permette di ottenere con alta precisione una triangolazione degli oggetti della scena ricostruendo quindi la posizione di osservazione e scatto. Software fondamentale a questo scopo è COLMAP che, a partire da un certo numero di immagini, ricostruisce uno spazio tridimensionale (tipicamente una point-cloud) stimando le posizioni di scatto delle varie immagini, che sono alla base della generazione dei campi di radianza neurale.

Come anticipato però precedentemente questi spazi risultanti sono tridimensionali, quindi oltre a tutte le difficoltà che si hanno nell'elaborazione di immagini bidimensionali si aggiungono quelle di rappresentazione di scene 3D; una volta capita la geometria della scena è necessario ricostruirla attraverso degli elementi che tipicamente sono mesh.
Una mesh è una rappresentazione tridimensionale di una superficie, costituita da un insieme di vertici, spigoli e facce che definiscono la geometria della forma. I vertici sono punti nello spazio, gli spigoli collegano i vertici formando linee e le facce sono superfici che connettono i vertici definendo la struttura complessiva dell'oggetto. Le mesh sono però solamente la forma della scena, e per andare a dare un aspetto visivo più dettagliato e realistico, vengono spesso utilizzate le texture. Le texture sono immagini bidimensionali applicate sulla superficie della mesh, conferendo colore, dettagli e caratteristiche visive agli oggetti tridimensionali. Queste immagini possono

rappresentare, ad esempio, superfici come pelli, tessuti, muratura o qualsiasi dettaglio visivo desiderato. E per capire come generare le texture da "incollare" alla mesh bisogna capire per ogni punto dell'immagine quali sono le fonti luminose che hanno influenza su di esso: entrano in gioco i modelli di illuminazione.

I modelli di illuminazione più utilizzati nell'ambito del fotorealismo sono i cosiddetti definiti "globali", tra i quali si hanno due modelli principali: il ray-tracing e la radiosità.
Il Ray-Tracing segue il percorso ottico della luce, simulando più realisticamente gli effetti luminosi, le ombre, i riflessi e le rifrazioni. In breve, esso funziona emettendo raggi di luce virtuali da una sorgente di luce o da una telecamera virtuale tracciandone il percorso attraverso la scena. Lungo il loro tragitto i raggi possono interagire con gli oggetti nella scena, producendo effetti visivi come ombre, riflessi e rifrazioni. Sapendo quali sono tutte le fonti luminose che influiscono su un punto si riesce ad assegnargli un colore. Essendo un calcolo computazionalmente oneroso, negli ultimi anni si è optato per aggiungere a livello hardware un componente adibito al calcolo del ray tracing in modo tale da alleggerire il carico sulla GPU stessa.

L'altra tecnica fondamentale è la Radiosity che si basa sul calcolo della radianza, ovvero quanta energia luminosa viene emessa dalle superfici. Nella radiosity l'intera superficie dell'immagine viene divisa in patch di egual dimensione e ciascun patch viene considerato come sorgente luminosa autonoma. Più patch significa più qualità finale nella resa qualitativa dell'immagine, ma significa anche più costi computazionali. Difatti, per ogni coppia di patch viene calcolato il cosiddetto "fattore di forma", ovvero quanto la luminosità di un patch influisca sull'altro e viceversa. Di fondamentale importanza è anche il fatto che prima del calcolo di questi fattori di forma vanno applicati anche algoritmi per la rimozione di superfici nascoste (HSR), ed anche queste sono dispendiose di risorse.
Il vantaggio che ha la radiosity è che il risultato che si ottiene è indipendente dal punto di vista dell'osservatore, in quanto la luce nella scena non cambia se ci si sposta all'interno di essa. L'unico caso in cui bisogna ricalcolarla è quello in cui la scena subisce variazioni, sia nella geometria oppure nelle sorgenti luminose.

Quest'ultima tecnica è il fondamento dell'argomento trattato in questo lavoro: calcolare il campo di radianza di una scena significa proprio andare a calcolare la sua radianza che, a differenza degli approcci tipici, qui è calcolata attraverso tecniche di intelligenza artificiale. Questo approccio prende il nome di Neural Rendering.

2.4 Neural Radiance Fields

Un Neural Radiance Field(NeRF) è un modello di grafica computerizzata che viene utilizzato per rappresentare spazi e scene tridimensionali. La sua introduzione risale al 2020[2] ed è un approccio basato sulle reti neurali per sintetizzare immagini fotorealistiche.
La chiave del suo funzionamento è la rappresentazione volumetrica di un campo di radianza neurale che, rappresentando una funzione continua nello spazio tridimensionale associa ad ogni punto una serie di proprietà quali ad esempio colore e densità volumetrica. Invece quindi di modellare direttamente la geometria degli oggetti un NeRF modella la varianza nello spazio.

Per creare un NeRF è necessario un insieme di dati di addestramento che consiste in coppie di punti nello spazio 3D ed i corrispondenti colori dei punti osservati da diverse angolazioni. I dati di training in questione possono provenire da immagini catturate da diverse prospettive o da modelli 3D sintetici. L'architettura NeRF apprende durante la fase di addestramento i parametri di una rete neurale profonda MLP che stima il colore e la densità (opacità) per tutti i punti dello spazio. Essa prende in input le coordinate spaziali di un punto e restituisce tutte le proprietà caratteristiche della radianza nel campo. In questo modo si riescono a catturare le relazioni complesse tra i punti nello spazio riuscendo così a rappresentare in modo realistico una scena.

Una volta concluso l'addestramento il campo di radianza neurale risultante è completamente esplorabile andando a posizionare una camera virtuale lungo un percorso personalizzato che non stavolta non è vincolato alla posizione di scatto della foto o del video. Questo processo, come anticipato in precedenza, è ripetibile tutte le volte che si desidera essendo il campo di radianza invariante nel tempo se la scena non cambia.

NeRF raggiunge risultati notevoli nell'ambito della sintesi di viste fotorealistiche, ma un problema con NeRF classico e le sue varianti è la definizione dell'isosuperficie della densità di volume. Questo concetto è cruciale poiché determina quali parti della scena 3D vengono considerate "interne" o "esterne". Attualmente, la pratica comune spesso si basa su euristiche di sogliatura sulla densità volumetrica, ma questo approccio può generare superfici rumorose e inaccurate, poiché le euristiche di sogliatura non forniscono vincoli sufficienti sui set di livelli della densità. Di conseguenza, la superficie ottenuta può non rappresentare con precisione la struttura reale della scena, rendendo preferibile una modellazione più diretta delle superfici per problemi di ricostruzione fotogrammetrica.

2.5 Ricostruzione di mesh da immagini

La ricostruzione di mesh a partire da immagini costituisce una sfida fondamentale nell'ambito della computer vision e della grafica computerizzata. Nel corso degli anni, diversi approcci sono stati sviluppati per affrontare questa sfida, ciascuno con le proprie caratteristiche e limitazioni. Tra i metodi più diffusi si annoverano quelli basati su struttura, che sfruttano informazioni geometriche e topologiche per stimare la forma della superficie tridimensionale. Tuttavia, questi approcci spesso dipendono da condizioni di illuminazione e visibilità ben definite, risultando limitati in scenari complessi o in presenza di ombre e riflessi.

Un'altra categoria di approcci, più recenti e in continua evoluzione, si basa sull'utilizzo di reti neurali per la ricostruzione tridimensionale. Questi metodi sfruttano le capacità di apprendimento delle reti neurali per inferire la geometria della scena direttamente dalle immagini. Modelli come MeshRCNN [3] e Pixel2Mesh [4] utilizzano reti neurali convoluzionali per estrarre caratteristiche dalle immagini e generare mesh tridimensionali. Tuttavia, tali approcci possono essere influenzati da problemi di ambiguità nelle informazioni bidimensionali, spesso portando a ricostruzioni approssimate in presenza di dettagli complessi.

Recentemente, si è assistito a una crescente adozione di approcci basati su reti neurali generative per la ricostruzione di mesh. Tecniche come la Generative Adversarial Network (GAN) hanno dimostrato promettenti risultati nella generazione di mesh realistiche da immagini. Queste reti sono in grado di apprendere la distribuzione delle mesh tridimensionali reali e di generarne di nuove che rispettino le caratteristiche strutturali della scena osservata.

Nonostante i progressi significativi, la ricostruzione di mesh da immagini rimane un campo attivo di ricerca, con sfide aperte relative alla precisione, alla gestione dei dettagli fini e all'adattamento a scenari complessi e dinamici. Approcci ibridi che combinano informazioni geometriche tradizionali con le capacità di apprendimento delle reti neurali stanno emergendo come una prospettiva interessante per affrontare queste sfide e raggiungere risultati più accurati e robusti.

Anche se gli approcci NeRF non sono nati con l'intento di arrivare ad una generazione di mesh, la popolarità di questo tipo di rappresentazione di oggetti 3D ha portato allo sviluppo di diversi lavori per adattare la metodologia all'output desiderato: sono stati adottati sia approcci classici (e.g., marching cubes[5]) che metodologie create appositamente[6][7]. La ricostruzione di mesh da NeRF soffre, però, di alcune limitazioni, come ad esempio superfici ricostruite rumorose, o con elementi mancanti, specialmente in presenza di colori omogenei o variazioni cromatiche intense.

Per ovviare a queste problematiche e quindi puntare alla generazione di mesh più accurate sono nati vari approcci che verranno approfonditi nel capitolo 3 di approfondimento, e sono basati sulla Signed Distance Function(SDF).
Tra i primi approcci basati su SDF si ha NeuS[8], che punta alla ricostruzione neurale di superfici utilizzando architetture a MLP basati su coordinate per rappresentare la scena proprio attraverso funzioni di distanza firmata.

Un altro approccio più recente è quello di Neuralangelo, che si basa anch'esso sulla SDF per affrontare le limitazioni riscontrate negli approcci basati su NeRF. Neuralangelo, in particolare, adotta Instant NGP (Neural Graphics Primitives)[9] come rappresentazione SDF della scena.

In ultimo, verranno trattati approcci totalmente differenti dai campi di radianza neurale, basati invece su Gaussian Splatting, una tecnologia estremamente recente ma che ha visto un enorme sviluppo grazie anche agli ottimi risultati ottenuti sia nel campo di render di scene, che nell'estrazione di mesh.

2.6 NeRF

NeRF o Vanilla NeRF è la prima implementazione riuscita di campo di radianza neurale in letteratura. L'architettura rappresenta la scena tridimensionale come un vettore 5-dimensionale, i quali input sono: un punto 3D $x = (x, y, z)$ insieme con una direzione di visualizzazione 2D, dove $d = (\theta, \psi)$. L'output fornito da questi insiemi di coordinate danno un colore emesso $c = (r, g, b)$ ed una densità di volume σ. Ed in pratica, la direzione viene rappresentata da un vettore cartesiano 3D unitario. Questa rappresentazione 5-dimensionale della scena viene approssimata con una rete MLP $F_\Phi : (x, d) \rightarrow (c, \sigma)$ ed i suoi pesi Φ vengono ottimizzati per mappare ciascun input 5-dimensionale alla sua densità di volume ed al colore emesso nella direzione di visualizzazione.

NeRF incoraggia la rappresentazione ad essere consistente con la posizione multi-vista limitando la rete nel predire la densità di volume σ come una funzione della sola direzione x, permettendo invece al colore c di venire predetto come una funzione di entrambe posizione e direzione di visualizzazione.
Per ottenere questo risultato la rete MLP processa inizialmente l'input tridimensionale x con 8 strati fully-connected utilizzando ReLU come funzione di attivazione e 256 canali per strato, ottenendo come risultato il vettore di densità volumetrica σ ed un vettore di feature 256-dimensionale. Questo vettore viene infine concatenato con la direzione del raggio della telecamera e, successivamente, passato ad uno strato

fully-connected addizionale che fornisce come output il colore RGB dipendente dalla vista.

I ricercatori sottolineano però come un'architettura costruita in tal modo porta scarsi risultati alle alte frequenze, dove quindi colore e geometria variano repentinamente. Le reti neurali profonde hanno, di fatto, un bias che le porta ad apprendere meglio funzioni a bassa frequenza rispetto a quelle con frequenza più alta. Viene quindi introdotto un encoder definito encoder posizionale in modo analogo a come è stato fatto nell'architettura Transformer [10], seppur con uno scopo differente da essa; i Transformer impiegano questo encoder per fornire le posizioni discrete dei token in una sequenza di input ad un'architettura che non contiene nessuna nozione di ordine. Nella tecnologia NeRF, al contrario, viene utilizzato per mappare le coordinate di input continue in uno spazio di dimensione superiore al fine di consentire alla rete MLP di approssimare più facilmente le frequenze elevate.

Di fatto, ad ogni parametro in input F_Θ viene applicata la seguente funzione di encoding:

$$\gamma(p) = \left(\sin\left(2^0\pi p\right), \cos\left(2^0\pi p\right), \ldots, \sin\left(2^{(L-1)}\pi p\right), \cos\left(2^{(L-1)}\pi p\right)\right) \tag{2.1}$$

Questo encoder viene quindi applicato ai parametri sopracitati normalizzati sia dal punto x, che dalla visualizzazione d. Il parametro L, viene settato a 10 per la posizione, a 4 per la direzione di visualizzazione.

Al fine di ottimizzare i risultati necessari per eseguire il neural rendering di una funzione di volume, si decide di scartare l'approccio di campionare lo spazio in N punti. In alternativa, si opta per la definizione di due reti separate: una denominata "coarse" (grezza) e l'altra "fine" (dettagliata) per la rappresentazione del volume. Inizialmente, si valuta la funzione in modo più generale, e successivamente, in base ai risultati ottenuti, si considera la parte più dettagliata al fine di definire un campionamento discreto non uniforme. Ciò comporta un aumento del numero di campioni nelle aree in cui ci si attende un maggiore contenuto visibile. Questa strategia tiene conto del fatto che una parte significativa della scena può essere considerata spazialmente vuota, e quindi non è cruciale apprendere i parametri relativi a questa zona.

Viene quindi generato un rendering volumetrico in cui si riporta il colore di ogni raggio passante per la scena utilizzando i principi classici del volume rendering. Il singolo colore sarà dato dalla funzione

$$C(r) = \int_{t_n}^{t_f} T(t)\sigma(r(t))c(r(t), d)\, dt \tag{2.2}$$

con

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma(r(s))\, ds\right) \tag{2.3}$$

tn e tf sono i valori massimi e minimi della sezione di spazio infinitesimo. Questa funzione $T(t)$ definisce la probabilità che un raggio passante da tn a tf non colpisca altre particelle nel suo tragitto. È necessario quindi calcolare $C(r)$ per ogni raggio emesso da ogni singola vista per la posa richiesta. Per il calcolo viene utilizzata una quadratura deterministica con un campionamento stratificato, in modo che la rete MLP possa essere valutata in posizioni continue tra loro nel corso del training.

L'architettura è progettata in modo tale da consentire l'ottenimento di una rappresentazione volumetrica da una singola rete. Di conseguenza, la singola scena viene ottimizzata a partire dalle immagini e dalle pose corrispondenti. Ad ogni iterazione del processo di ottimizzazione, si effettua un campionamento casuale dei raggi, successivamente elaborati attraverso il campionamento gerarchico precedentemente definito. Per ciascun insieme di raggi campionati, viene impiegata la tecnica del volume rendering, precedentemente descritta, al fine di ottenere una rappresentazione di ciò che viene fornito in input all'architettura. La funzione di loss utilizzata per l'addestramento consiste nella somma degli errori quadrati tra i raggi renderizzati e quelli originali per ciascun canale di colore, sia per la rete sparsa che per quella fine. Infine, i parametri di addestramento, come il numero di raggi per batch, sono definiti in base alle capacità dell'hardware disponibile, con una raccomandazione di 64 campionamenti per raggio nel caso di una rete sparsa e 128 nel caso di una rete fine. Per completare il processo di addestramento, viene impiegato l'ottimizzatore Adam.

I dati di addestramento presenti nell'articolo di ricerca, ampiamente adottati in numerosi studi successivi, derivano da scene sintetiche create mediante l'uso dello strumento di progettazione grafica Blender. Da queste scene vengono estratte le matrici delle telecamere, che rappresentano la posizione dell'osservatore nello spazio. Oltre alle singole immagini, nel caso del Synthetic Dataset citato nel paper, comprendente 100 immagini per il training, 100 per la validazione e 100 per la valutazione, un file JSON contiene le coordinate di acquisizione delle viste e l'orientamento. Per le scene reali, viene menzionato l'utilizzo dello strumento COLMAP, di cui si parlerà successivamente. I risultati ottenuti vengono valutati mediante metriche come PSNR, SSIM e LPIPS, introdotte più avanti.

Gli autori del lavoro riportano, come già accennato, notevoli capacità di rappresentazione delle scene, nonostante i considerevoli inconvenienti legati al tempo di addestramento dello strumento. Per ottenere i risultati riportati, viene menzionata la necessità di hardware di alto livello, come una nVidia V100, per la notevole disponibilità di VRAM e almeno 12 ore di addestramento, preferibilmente nell'ordine

di un giorno. Queste limitazioni tecnologiche significative, come vedremo, saranno superate da ulteriori studi che apporteranno modifiche lievi ma sostanziali.
D'altra parte, rispetto ad altre tecniche di rendering di scene tridimensionali, questo approccio presenta diversi vantaggi significativi. Tra questi, la possibilità di racchiudere la rappresentazione della scena in memoria con minimo dispendio di memoria, determinato unicamente dai pesi dei neuroni della rete (5 MB per questa architettura), la capacità di sintetizzare immagini da nuove pose e i notevoli progressi fatti, al momento della pubblicazione della ricerca, nella capacità di rendering neurale dello strumento. Di seguito, in figura 2.1, una pipeline del processo NeRF.

2.6.1 Instant-NGP

La problematica principale riscontrata nella tecnica NeRF tradizionale è rappresentata dal notevole tempo di computazione richiesto per l'allenamento della rete, che può estendersi fino a diverse decine di ore. Nonostante numerosi studi abbiano cercato di migliorare sia la qualità dei risultati ottenuti che i tempi di esecuzione, un progresso significativo è stato raggiunto con l'introduzione di Instant-NGP.
Instant-NGP costituisce un passo avanti essenziale, prendendo spunto dal concetto introdotto nei NeRF tradizionali riguardante l'encoder posizionale. Questo componente svolge un ruolo chiave nell'elaborare caratteristiche complesse ad alta frequenza presenti nella scena, contribuendo a migliorare la qualità della rappresentazione tridimensionale. Tuttavia, l'uso di encoder di questo tipo introduce alcune limitazioni. In particolare, potrebbe comportare limitazioni in termini di prestazioni computazionali, e la complessità aggiunta potrebbe rendere più ardua l'implementazione di soluzioni hardware ottimizzate per il training delle reti neurali. Inoltre, la dipendenza da euristiche o modifiche strutturali può aumentare la complessità computazionale, presentando sfide ulteriori nella gestione del modello neurale.

I ricercatori introducono quindi un encoder a griglia hash multirisoluzione (Multiresolution Hashgrid Encoder). L'utilizzo di una tabella hash per apprendere le caratteristiche spaziali permette di affrontare alcune delle problematiche riscontrate nelle NeRF di Mildenhall, che utilizzavano una doppia architettura "coarse" e "fine". A basse risoluzioni, la tabella mantiene una mappatura 1:1 tra i punti della griglia e gli array di entry, mentre a risoluzioni più elevate, l'array funge da tabella hash gestendo le collisioni. Questo approccio consente di calcolare la media dei gradienti di addestramento, privilegiando i gradienti più rilevanti per la funzione di addestramento. In pratica, si dà priorità alle zone con dettagli più fini rispetto a quelle più sparse, accelerando la convergenza del modello. L'assunzione di base è che nelle zone sparse, caratterizzate da una minore densità di informazioni tridimensionali, ci sia meno da apprendere rispetto alle zone più dense. L'aumento dimensionale del positional

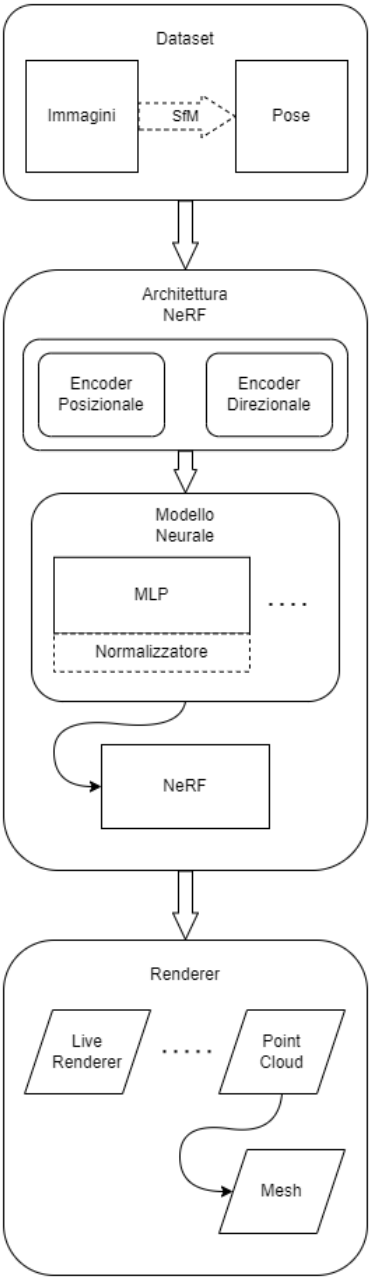

Figura 2.1: Pipeline del processo NeRF[1]. Il processo parte da un dataset di immagini che può già includere le pose di scatto; se queste non sono presenti vengono utilizzati strumenti di Structure from Motion per ricavarle. I dati vengono trasmessi all'architettura NeRF che li processa con uno strato di encoder. Il modello neurale, che può contenere un diverso numero di MLP, prende in ingresso l'output degli encoder e, quando presente, utilizza una funzione di normalizzazione per l'accumulo della densità. Attraverso il processo di training viene generato un campo NeRF. Questo può essere visualizzato attraverso tecniche di rendering, da un Live Renderer volumetrico all'accumulo dell'output in profondità in point cloud. Da questa poi è possibile andare ad applicare tecniche di estrazione di mesh.

encoder, presente nelle NeRF classiche, è mantenuto tramite tabelle hash legate a diverse risoluzioni dell'immagine. Questo consente di parametrizzare la risposta dell'immagine a diverse frequenze, offrendo maggiore flessibilità e adattabilità al modello complessivo. E' possibile quindi utilizzare quest'architettura per interrogare le tabelle hash di ogni risoluzione, contemporaneamente.

L'introduzione di questo encoder insieme ad altre migliorie alla rete MLP consentono quindi di ottenere miglioramenti nella qualità di resa grafica, ma soprattutto nelle tempistiche di allenamento: in particolare, nel paper originale si cita come da ore di calcolo si passi a 5 minuti sul blender dataset su una scheda che è la RTX 3090 NVIDIA. Ciò denota non solo un cambiamento nella qualità ma anche nell'ambito in cui questi allenamenti possono essere portati avanti; questa tipologia di schede grafiche è difatti caratteristica dell'ambito consumer.

2.6.2 Nerfacto

Lo sviluppo di varie tecnologie nell'ambito NeRF porta alla creazione di Nerfacto: un metodo sviluppato dal team di Nerfstudio[11], che è un toolkit software per l'allenamento di reti NeRF di cui si parlerà più in dettaglio nel capitolo 3. L'approccio di Nerfacto sfrutta le fondamenta di alcune tecnologie NeRF quali: NeRF–[12], Instant-NGP[9], NeRF-W[13] e Ref-NeRF[14] ed è possibile osservarne la pipeline nella figura 2.2.

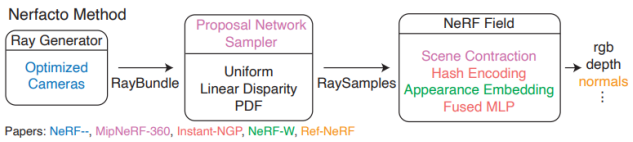

Figura 2.2: Diagramma di Nerfacto. Combina le funzionalità di vari paper (in basso a sinistra), tenendo conto del fatto che il diagramma varia continuamente in base allo sviluppo dello stato dell'arte ed alla loro aggiunta nella codebase di Nerfstudio

Il metodo Nerfacto ottimizza inizialmente le visuali della fotocamera mediante una trasformazione ottimizzata SE(3) che si riferisce alla "Special Euclidean group in Three Dimensions", gruppo formato da tutte le trasformazioni isometriche (che quindi conservano le distanze) che includono sia una traslazione che una rotazione tridimensionale. Queste visuali della fotocamera vengono successivamente utilizzate per generare i RayBundles: delle primitive rappresentanti di sezioni attraverso uno spazio 3D parametrizzate con un'origine, direzione, ed altre meta-informazioni come ad esempio indici di camera. Al fine di migliorare l'efficienza e l'efficacia del processo

di campionamento si utilizza un campionatore a tratti, che effettua campionamenti uniformi fino ad una distanza fissa dalla fotocamera, seguiti da campioni distribuiti in modo che la dimensione del passo aumenti con ciascun campione.

Questi campioni vengono poi dati in pasto ad una Proposal Network[15] che consolida le posizioni dei campioni nelle regioni della scena che contribuiscono maggiormente al rendering finale, migliorando notevolmente la qualità della ricostruzione. In aggiunta, viene utilizzata una rete MLP fusa con un hash-encoder come visto nella tecnica Instant-NGP[9](Figura2.3) per apportarne i vantaggi visti in precedenza.

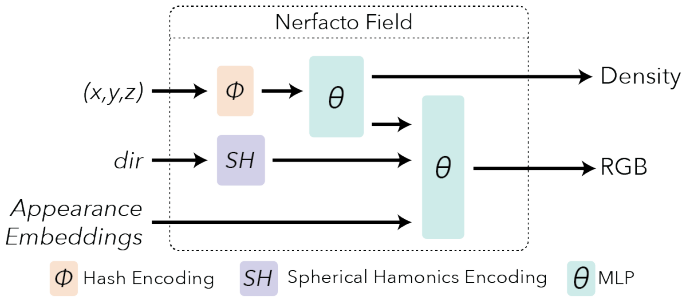

Figura 2.3: Architettura di generazione ed interrogazione di un campo di radianza neurale in Nerfacto.

Il campo generato nella figura 2.3 sfrutta l'encoder posizionale definito in Instant-NGP insieme con un encoder direzionale basato su armoniche sferiche. La risposta viene fornita con due MLP, una per la scena in densità, l'altra in colore formato RGB.

2.7 SDF

L'evoluzione nella ricostruzione di superfici 3D è passata attraverso diverse fasi, da approcci tradizionali come NeRF a tecniche più recenti basate sulla SDF, incarnate da metodi come NeuS. NeRF ha dimostrato la sua potenza nella sintesi di viste fotorealistiche, mappando le scene attraverso MLP per la generazione di colori e densità volumetrica. Tuttavia, la definizione dell'isosuperficie, cioè la superficie dove la densità è costante, porta spesso a ricostruzioni rumorose e imprecise, specialmente in scenari con colori omogenei o variazioni cromatiche intense. I nuovi metodi basati su SDF puntano quindi ad assegnare ad ogni punto dello spazio la distanza dalla superficie, in modo tale da aumentare la resa e la qualità delle superfici ricostruite. In figura 2.4, un esempio di spazio SDF.

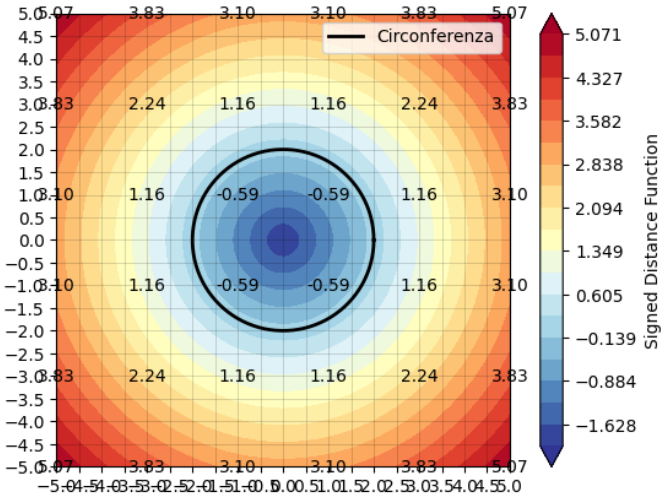

Figura 2.4: **Spazio SDF.**La sfera rappresenta la superficie, ed ogni punto nello spazio ha distanza > 0 se è fuori dalla superficie, < 0 se interno.

2.7.1 NeuS

Metodi antecedenti a NeuS per la ricostruzione neurale delle superfici come ad esempio DVR[16] ed IDR [17], necessitano di maschere di primo piano come supervisione, le quali però soffrono di problematiche nel facile intrappolamento in minimi locali. I metodi più recenti come NeRF e le sue varianti utilizzano il volume rendering per produrre una rappresentazione neurale della scena da cui però è difficile estrarre superfici di qualità per via dei pochi vincoli di qualità introdotti per esse.

Per superare queste sfide, NeuS si basa sull'utilizzo di Signed Distance Function (SDF) per rappresentare le superfici come il livello zero di una funzione di distanza firmata. Utilizza un nuovo metodo di rendering volumetrico per addestrare una rappresentazione neurale SDF, superando le limitazioni degli approcci tradizionali e mostrando una maggiore accuratezza nella ricostruzione, specialmente in scenari complessi con auto-occlusioni e strutture sottili. Rispetto ai metodi tradizionali, NeuS affronta e risolve problemi di bias (errori) geometrici intrinseci al rendering volumetrico convenzionale. In particolare, propone una formulazione priva di bias nel primo ordine di approssimazione, garantendo una ricostruzione della superficie più accurata anche senza la supervisione della maschera di primo piano, superando quindi la necessità di maschere rappresentanti il primo piano per una convergenza valida ed eliminando la difficoltà nell'ottenere superfici di alta qualità da rappresentazioni implicite apprese.

La superficie della scena S viene quindi rappresentata dal livello zero della SDF:

$$S = \left\{ x \in \mathbb{R}^3 \mid f(x) = 0 \right\} \tag{2.4}$$

dopodiché viene introdotta una funzione densità di probabilità chiamata S-density che viene utilizzata per allenare la rete SDF con solamente immagini 2D in input come supervisione. Una volta raggiunta la convergenza nella minimizzazione di una funzione di loss, il livello zero della network-encoded SDF rappresenta accuratamente la superficie ricostruita S, mentre la sua S-densità indotta assume perlopiù valori alti vicino alla superficie.

Per l'allenamento del modello, viene minimizzata la differenza tra i colori renderizzati ed i colori delle ground truth, senza nessuna supervisione 3D. Oltre ai colori, è possibile utilizzare delle maschere per la supervisione se disponibili.
Nello specifico, vengono ottimizzate sia le reti neurali che la deviazione standard inversa s campionando in maniera casuale un batch di pixel ed i loro rispettivi raggi nello spazio P, dipendente da C_k e M_k, dove C_k è il colore del pixel ed $M_k \in \{0, 1\}$ è la maschera opzionale, per una sola immagine in ogni iterazione.
La funzione di loss è definita come:

$$\mathcal{L} = \mathcal{L}_{\text{color}} + \lambda \mathcal{L}_{\text{reg}} + \beta \mathcal{L}_{\text{mask}} \tag{2.5}$$

La funzione loss del colore $\mathcal{L}_{\text{color}}$ viene definita come:

$$\mathcal{L}_{\text{color}} = \frac{1}{m} \sum_k \mathcal{R}(\hat{\mathcal{C}}_k, \mathcal{C}_k) \tag{2.6}$$

Dove $\mathcal{R}$ viene impiegata come L1 loss come in IDR[17].
Viene inoltre aggiunto un termine Eikonale[18] sui punti campionati per regolarizzare la SDF tramite:

$$\mathcal{L}_{\text{reg}} = \frac{1}{nm} \sum_{k,i} \left(\left\| \nabla f(\hat{p}_{k,i}) \right\|_2 - 1 \right)^2 \tag{2.7}$$

Infine, la loss relativa alla maschera opzionale $\mathcal{L}_{\text{mask}}$ è definita come:

$$\mathcal{L}_{\text{mask}} = \text{BCE}(M_k, \hat{O}_k) \tag{2.8}$$

dove $\hat{O}_k$ è la somma dei pesi lungo il raggio della camera e BCE è la loss cross entropia binaria.

La sperimentazione di questa rete è stata fatta su varie scene nel DTU dataset [19], ed al tempo della pubblicazione del rispettivo paper, viene riportato come le

performance di questo approccio superassero quelle dello stato dell'arte. Tuttavia, da NeuS sono passati diversi anni e sono nate nuove tecnologie, in particolare la prossima che verrà affrontata.

2.7.2 Neuralangelo e Bakedangelo

Dopo aver esplorato l'approccio innovativo proposto da NeuS per la ricostruzione neurale di superfici 3D, si volge ora ad un'evoluzione significativa rappresentata da Neuralangelo. Mentre NeuS ha apportato miglioramenti cruciali nella rappresentazione delle superfici attraverso la funzione di distanza firmata (SDF) e il rendering volumetrico, Neuralangelo si distingue per l'introduzione di Instant NGP (Neural Graphics Primitives), una rappresentazione ibrida che promette di superare alcune delle limitazioni riscontrate nei metodi precedenti. Neuralangelo adotta Instant NGP come una forma avanzata di rappresentazione SDF, integrandola con tecniche di rendering neurale per la ricostruzione di superfici ad alta fedeltà. Come visto in precedenza, l'introduzione di una griglia 3D multi-risoluzione basata su hash e una struttura di rete neurale leggera permettono di rappresentare dettagli fini e gestire strutture complesse.

I ricercatori di Neuralangelo sottolineano come però sia presente una problematica legata alla computazione del gradiente numerico nell'ambito della codifica hash: poiché il gradiente analitico risulta limitato a regioni locali, viene proposto l'uso di gradienti numerici per garantire una rappresentazione pià coerente delle superfici attraverso ottimizzazioni non locali (Figura 2.5)

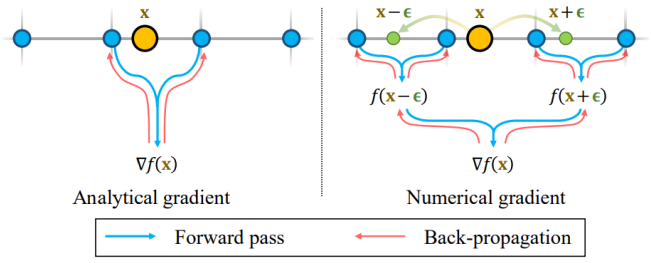

Figura 2.5: Utilizzare **gradienti numerici** per le derivate di ordine superiore distribuisce gli aggiornamenti della back-propagation oltre le griglie hash locali, realizzando quindi una versione rilassata del **gradiente analitico**

Un po' come nell'ambito di Nerfacto, Neuralangelo adotta una strategia "coarse-to-fine" sia per rappresentare la loss in modo tale da evitare di finire in minimi locali, ma anche per ricostruire le superfici utilizzando una tecnica a livello di dettaglio progressivo, rendendolo quindi capace di catturare dettagli a diverse scale durante il

processo di ottimizzazione. Il level of detail impiegato in Neuralangelo è influenzato da due aspetti principali: la dimensione del passo ϵ e la risoluzione della griglia hash V.

La dimensione del passo ϵ controlla la dimensione della "finestra" entro cui vengono calcolati i gradienti numerici sopracitati. Un passo (ϵ) maggiore introduce uno smoothing su una scala più ampia, producendo superfici coerenti a livello globale ma con dettagli meno definiti. Al contrario, un passo più piccolo si concentra su dettagli più fini a discapito della coerenza delle superfici su larga scala. Va da se che essendo il gradiente numerico una versione rilassata del gradiente analitico, se la dimensione del passo è più piccola rispetto alla grandezza della griglia di hash, il gradiente numerico coincide con il gradiente analitico.

La griglia hash, invece, è organizzata in diverse risoluzioni: durante l'ottimizzazione si attivano progressivamente risoluzioni più dettagliate della griglia permettendo al modello di concentrarsi, inizialmente, sui dettagli di scala più ampia (coarse) arrivando mano mano poi a raffinarsi per rappresentare i dettagli geometrici più fini (fine). Tutto questo senza perdere la coerenza strutturale.

Al fine di rendere più intuitivo il funzionamento del LoD, viene riportato un esempio di come la griglia hash a multi risoluzione generi vari livelli di dettaglio in figura 2.6

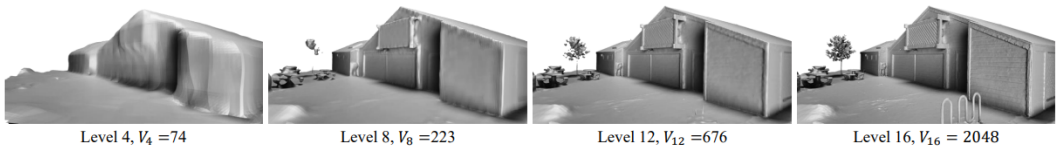

Figura 2.6: **Risultati a differenti risoluzioni hash.** Mentre alcune strutture come ad esempio l'albero, il tavolo e la pensilina per la bici vengono mancate alle risoluzioni "coarse", le risoluzioni "fine" riescono progressivamente a recuperare queste superfici mancanti.

Il processo di training di Neuralangelo inizia con la previsione della densità volumetrica utilizzando una rete neurale che genera previsioni di densità nei punti 3D della scena,previsioni che vengono poi convertite in rappresentazioni SDF. Inoltre, similmente a come avviene per NeuS, Neuralangelo applica una regolarizzazione eikonale per garantire una rappresentazione coerente e precisa della superficie.Tuttavia è importante sottolineare la proprietà della SDF è la sua differenziabilità con il gradiente a norma unitaria. Il gradiente della SDF soddisfa l'equazione eikonale $\|\nabla f(x)\|_2$ è uguale ad 1 quasi ovunque e viene quindi imposta la eikonal loss sulle predizioni:

$$\mathcal{L}_{\text{eik}} = \frac{1}{N}\sum_{i=1}^{N}(\|\nabla f(x_i)\|_2 - 1)^2 \tag{2.9}$$

dove N è il numero di punti campionati.

L'ottimizzazione del modello prosegue incoraggiando la regolarità della superficie ricostruita andando ad imporre una regolarizzazione della curvatura media della SDF che viene calcolata utilizzando il Laplaciano discreto:

$$\mathcal{L}_{\text{curv}} = \frac{1}{N}\sum_{i=1}^{N}|\nabla^2 f(x_i)|_2 \tag{2.10}$$

Infine, la loss totale è definita come la somma pesata di tutte le loss:

$$\mathcal{L} = \mathcal{L}_{\text{RGB}} + w_{\text{eik}}\mathcal{L}_{\text{eik}} + w_{\text{eik}}\mathcal{L}_{\text{curv}} \tag{2.11}$$

dove la $\mathcal{L}_{\text{RGB}}$ è la stessa loss RGB vista in precedenza.

I test riportati nel paper Neuralangelo sono stati effettuati su molteplici dataset, tra i quali il DTU ed il Tanks and Temples dataset[20] e comparati con i risultati di Neus e NeuralWarp. Esempio in figura 2.7

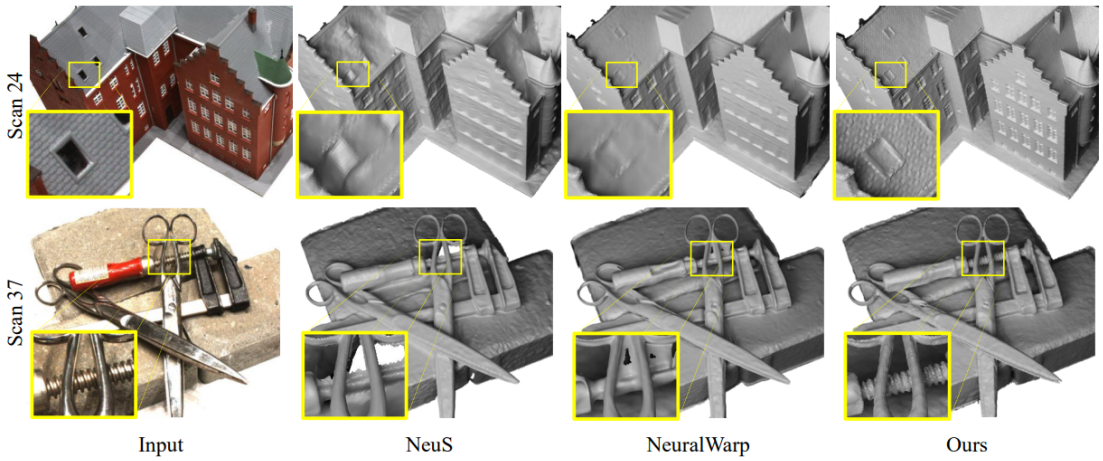

Figura 2.7: **Comparazione qualitativa sul dataset DTU.**. E' possibile osservare come Neuralangelo produca superfici più accurate rispetto ai metodi precedenti.

2.8 Gaussian Splatting

Un'ultima parte degli esperimenti è stata ispirata da un nuovo approccio basato sulle gaussiane 3D[21]. Il termine "splatting" si riferisce all'idea di "schiacciare" o "proiettare" i punti di dati nello spazio 3D su un piano di visualizzazione bidimensionale. L'approccio "Gaussian" indica che durante questo processo viene utilizzata una funzione Gaussiana per diffondere e pesare l'influenza dei punti su un'area più ampia, rendendo la visualizzazione più continua e facilitando la percezione delle distribuzioni di densità dei dati.

1. **Proiezione dei punti:** I punti di dati tridimensionali vengono proiettati su un piano di visualizzazione bidimensionale. Questo può essere fatto utilizzando una proiezione ortografica o prospettica, a seconda delle esigenze dell'applicazione.

2. **Creazione degli "splat":** Ogni punto proiettato viene trasformato in uno o più "splat", che sono essenzialmente delle forme bidimensionali che rappresentano il punto originale. La forma può essere una gaussiana bidimensionale, da cui il termine *Gaussian Splatting*.

3. **Pesatura gaussiana:** Per ciascuno "splat", viene applicata una funzione Gaussiana per determinare quanto influenzi i pixel circostanti. Questa funzione può essere utilizzata per modellare la distribuzione spaziale della densità dei dati, con un peso maggiore al centro del "splat" e un decadimento graduale verso i bordi.

4. **Accumulo di valori:** I valori dei pixel sono accumulati considerando le sovrapposizioni di diversi "splat". Questo processo consente di ottenere una rappresentazione visiva più fluida e continua dei dati, con una transizione graduale tra i punti.

Nello specifico di questo approccio, si fornisce un esempio della pipeline Gaussian Splatting in figura 2.8

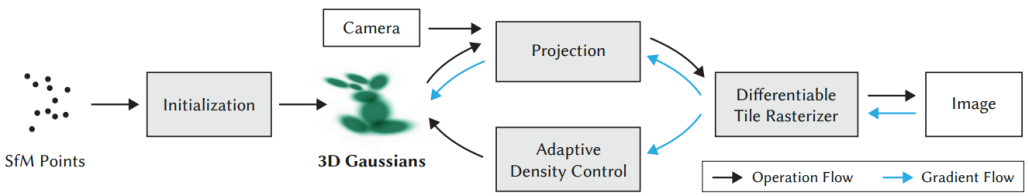

Figura 2.8: **Pipeline di Gaussian Splatting.**

L'approccio di Gaussian Splatting si basa sulla rappresentazione della scena attraverso le Gaussiane 3D. Queste Gaussiane offrono una descrizione dettagliata della geometria tridimensionale della scena, includendo informazioni sulla varianza tramite la matrice di covarianza, la posizione spaziale, l'opacità (α), e i coefficienti sferici armonici. Questa rappresentazione dettagliata consente di modellare in modo flessibile una vasta gamma di oggetti e forme presenti nella scena, fornendo una base per la successiva ottimizzazione. Quest'ultima è fondamentale per adattare dinamicamente la rappresentazione alla complessità specifica della scena. L'utilizzo di tecniche di discesa del gradiente stocastico consente di regolare in modo continuo le proprietà delle Gaussiane, ottimizzando la posizione, l'opacità, la covarianza e i coefficienti sferici armonici. Questo processo di ottimizzazione è combinato con passaggi di controllo di densità adattiva, assicurando che la rappresentazione sia compatta, dettagliata e adeguata alla scena reale. Il controllo adattivo della densità delle Gaussiane è una componente cruciale per gestire la complessità della scena in modo efficiente. Durante questa fase, le Gaussiane vengono aggiunte o rimosse sulla base delle esigenze specifiche della scena. Si presta particolare attenzione alle aree "under-reconstruction" e "over-reconstruction". L'utilizzo di un criterio di saturazione assicura che il numero di Gaussiane coinvolte nel blending sia gestito in modo ottimale.

La rasterizzazione veloce è implementata attraverso un rasterizzatore basato su piastrelle specificamente progettato per le Gaussiane. Questo approccio consente una resa rapida e un ordinamento efficiente delle primitive, facilitando il processo di rendering. Proprio per il processo di rasterizzazione,è fondamentale che sia differenziabile per consentire una retropropagazione efficiente durante il processo di apprendimento. L'α-blending approssimato delle Gaussiane è una componente chiave per ottenere una resa visiva di alta qualità. Questo processo di fusione approssimata durante la rasterizzazione è esteso anche a situazioni di "splatting" anisotropo, contribuendo alla resa dettagliata della scena.

La fase di retropropagazione è progettata per gestire in modo efficiente il calcolo dei gradienti durante l'apprendimento. Questo è cruciale per adattarsi dinamicamente a scenari con profondità variabile senza limiti rigidi sul numero di primitive coinvolte nel processo di blending. L'approccio si basa sulla differenziabilità dell'intero processo, garantendo che l'apprendimento possa avvenire in modo ottimale. In ultimo, il controllo della densità e l'ottimizzazione possono essere iterati in modo continuo, garantendo che la rappresentazione della scena rimanga adattativa e di alta qualità. Questo approccio complessivo offre un equilibrio tra efficienza computazionale e capacità di apprendimento differenziabile, rendendolo adatto a scenari reali complessi. Visto il rilascio del codice proprio nei giorni della stesura di questa tesi, sono stati portati avanti degli esperimenti anche con questa metodologia che porta un approccio innovativo nella ricostruzione di scene 3D.

Capitolo 3

Materiali e Metodi

3.1 Software Utilizzato

3.1.1 Nerfstudio

L'approccio utilizzato è stato quello di partire dallo stato dell'arte per quanto riguarda la generazione di campi di radianza neurali per il rendering di scene 3D. Un tool che si è dimostrato di vitale importanza è stato Nerfstudio[11].

Nerfstudio è un tool software che fornisce delle API per il controllo del processo end-to-end di creazione, allenamento, e testing di campi di radianza neurale. Grazie al suo frontend grafico (Figura 3.1) permette di interagire con il processo di allenamento dando la possibilità di impostare vari parametri: dal rendering alla eventuale mesh.

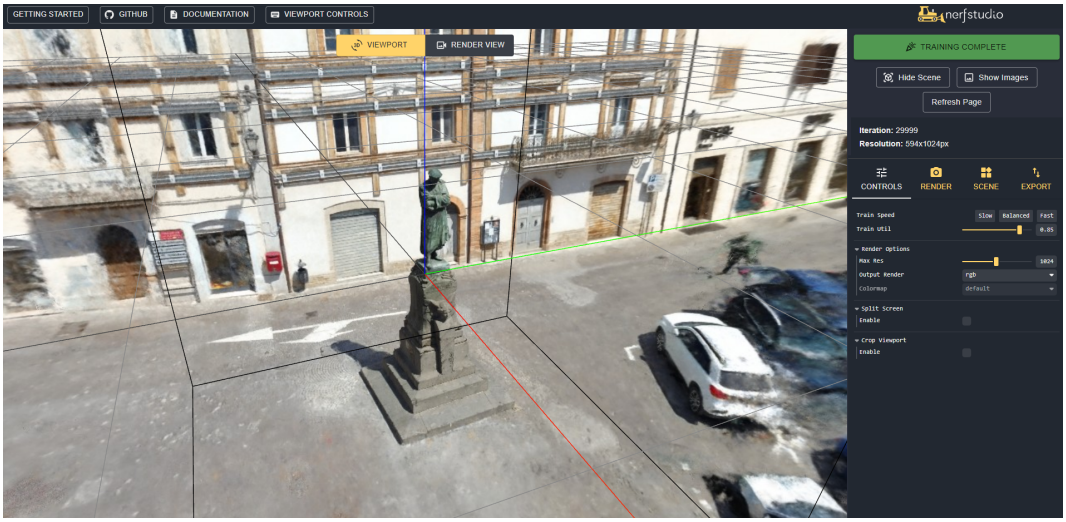

Figura 3.1: **Visualizzatore di Nerfstudio.**

Questo tool è comprensivo di diverse reti, tra cui:

- Nerfacto[1]: Metodo che racchiude varie tecniche al suo interno.
- Instant-NGP[9]: Instant Neural Graphics Primitives with a Multiresolution Hash Encoding
- NeRF[2]: Vanilla Neural Radiance Fields
- Mip-NeRF[15]: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields
- TensoRF[22]: Tensorial Radiance Fields
- Splatfacto[2]: Implementazione Gaussian Splatting

ed è in continuo aggiornamento per l'aggiunta di sempre più nuovi approcci con l'avanzamento della ricerca: nello specifico, il primo modello utilizzato è stato Nerfacto. Questo tool è allo stato dell'arte per la generazione di campi di radianza neurali, trovando un compromesso tra qualità di rendering e tempi di allenamento della rete.

Nerfstudio possiede un elemento fondamentale e prende il nome di Viewer: è un componente GUI in grado di interrogare a runtime l'architettura neurale per generare l'immagine a partire dal campo di radianza. Questa architettura è addestrabile attraverso un trainer che, sulla base del modello fornito, genera un campo di radianza piuttosto che un altro. Seppur quindi l'avvio dell'allenamento sia via terminale di comando, è possibile verificarne il corretto andamento attraverso il viewer che mano mano, affinerà il modello fino al numero di iterazioni richieste. Questo vantaggio non viene però a costo zero, in quanto il rendering in tempo reale necessità comunque di computazione, ma i costi sono trascurabili se il viewer viene utilizzato solo occasionalmente.

L'output di questo processo è un file contenente i pesi della rete neurale addestrata, e sempre da questo file è possibile generare un output point cloud, al quale è possibile applicare la tecnica di Poisson [23] per generare una mesh della scena. La scelta di Nerfstudio rispetto al codice originale degli ideatori delle varie reti è stata falla sulla base della semplicità di utilizzo di Nerfstudio, che si è ormai affermato nel panorama NeRF permettendo attraverso poche righe di comando di avviare un'allenamento che altrimenti richiederebbe un'estesa conoscenza del codice originale ma anche di

[1] Maggiori informazioni disponibili al sito https://docs.nerf.studio/nerfology/methods/nerfacto.html
[2] Maggiori informazioni disponibili al sito https://docs.nerf.studio/nerfology/methods/splat.html

linguaggi di programmazione a basso livello come ad esempio CUDA.

3.1.2 SDFStudio

Nerfstudio però al momento è comprensivo solamente di tecniche NeRF per la generazione di campi di radianza neurali. Si è vista necessaria quindi l'esigenza di un tool simile per l'allenamento di modelli basati su SDF. Nasce quindi SDFStudio[24], un framework unificato e modulare per la neural implicit surface reconstruction, e nasce sulle fondamenta di Nerfstudio. SDFStudio è comprensivo di vari modelli per la surface reconstruction, tra cui:

- Neuralangelo[25]
- Bakedangelo[24]
- VolSDF[26]
- NeuS[8]

Il funzionamento è analogo a quello di Nerfstudio ma, come vedremo nel capitolo dedicato agli esperimenti, ci sono differenze nell'esportazione delle mesh (che è proprio l'output di queste reti).

Entrambi Nerfstudio ed SDFStudio necessitano di lavorare su un dataset che può essere composto di foto o video, ma da sole non bastano: il funzionamento di questi modelli si basa sull'informazione di posizione di scatto dell'immagine e c'è quindi bisogno di ricavarle; per fare ciò, si utilizza il software COLMAP[27], di cui verrà fatta una trattazione più approfondita nella prossima sezione.

3.1.3 Gaussian Splatting

Per quanto riguarda l'approccio Gaussian Splatting, l'implementazione classica permette l'allenamento del modello solo al fine dell'esplorazione della scena. Difatti, la natura disorganizzata delle milioni di piccole Gaussiane 3D risultanti dall'ottimizzazione presenta una significativa problematica nella generazione di una mesh strutturata e organizzata. Per superare questa sfida, viene introdotto un termine di regolarizzazione chiave, noto come sugar, che mira a incoraggiare l'allineamento dei Gaussiani 3D con la superficie della scena. Questo termine di regolarizzazione si rivela

fondamentale nel guidare l'ottimizzazione delle Gaussiane verso una disposizione più coerente e orientata alla superficie, facilitando così l'estrazione successiva della mesh.

SuGaR[28], è quindi uno degli unici approcci riusciti di estrapolazione di mesh a partire da Gaussiane, e ne verrà riportata l'implementazione, corredata con tutti i test per valutarne le metriche.

3.2 CAVE Autonomous Virtual Environment

Il CAVE (Cave Automatic Virtual Environment) è un ambiente virtuale avanzato progettato per fornire un'esperienza immersiva e interattiva agli utenti. Esso presenta diverse caratteristiche chiave:

- **Pareti proiettive:** Utilizza schermi proiettivi ad alta risoluzione per creare uno spazio virtuale tridimensionale.
- **Pavimento proiettivo:** In alcuni casi, il pavimento può essere proiettato per consentire interazioni più immersive.
- **Sistemi di tracciamento:** Usa telecamere o sensori di movimento per monitorare la posizione e l'orientamento degli utenti e degli oggetti.
- **Dispositivi di input:** Gli utenti interagiscono con il CAVE utilizzando controller, guanti o altri dispositivi di tracciamento del movimento.
- **Visualizzazione stereo:** Indossando visori a realtà virtuale, gli utenti ottengono una visualizzazione stereo per una maggiore profondità.
- **Software di simulazione:** Il CAVE funziona con software specializzato per gestire la simulazione virtuale.

I CAVE sono ampiamente utilizzati in settori come la progettazione industriale, la simulazione di ambienti complessi, la formazione medica e altro ancora. Sono ospitati solitamente all'interno di cluster molto performanti, ed è presente anche un software chiamato TechViz che effettua una traduzione dagli input in Blender ad un formato riproducibile dal sistema CAVE. Sarà proprio attraverso questo software che si potrà usufruire della visualizzazione in 3D degli output mesh delle reti discusse in precedenza.

3.3 Dataset

Per prendere dimestichezza con i vari tool sopra evidenziati, sono stati effettuati dei primi esperimenti attraverso una serie di dataset già conosciuti in letteratura e riportati negli stessi paper dei modelli scelti. Il primo dataset utilizzato è molto famoso nell'ambito delle architetture NeRF, in quanto è stato introdotto con la prima implementazione di Mildenhaal ed è il Synthetic Dataset (conosciuto anche come Blender Dataset)[2]. E' composto da immagini di 8 oggetti che riportano geometrie complesse e materiali non-Lambertiani[3]. Per ogni oggetto sono disponibili 100 viste per il training, 100 per la valutazione e 100 per il testing, tutte ad una dimensione di 800x800 pixel.

Figura 3.2: Esempio di alcune immagini facenti parte della scena Lego del dataset Synthetic.

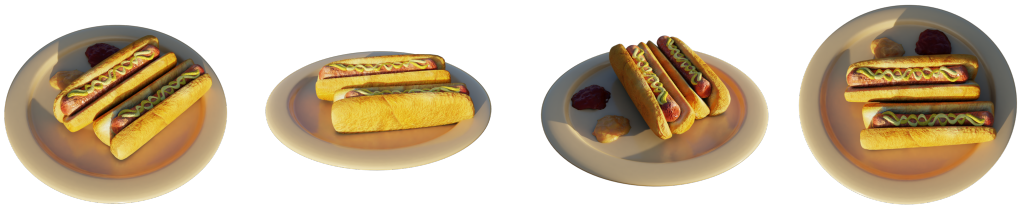

Figura 3.3: Esempio di alcune immagini facenti parte della scena Hotdog del dataset Synthetic.

Di questi 8 oggetti, dati i vincoli temporali per la realizzazione di ciascun esperimento, sono stati scelti 2 oggetti che sono: chair, lego.

Questa tipologia di dataset però è composta di immagini acquisite in condizioni ottimali e controllate: non sono quindi molto indicativi di prestazioni in ambienti più improvvisati o comunque personalizzati. Il Synthetic Dataset, difatti, riporta già le posizioni di scatto delle immagini nei vari transforms.json relativi ai vari split

[3]I materiali non lambertiani sono quelli che non seguono il modello semplificato di riflessione di Lambert. A differenza dei materiali lambertiani, la luminosità di un punto su una superficie non dipende solo dall'angolo di incidenza della luce, ma può variare in modo più complesso in base all'angolo di osservazione, alla posizione della luce e ad altri fattori.

training, evaluation,test. Si rivelano quindi necessari degli approcci per la ricerca delle coordinate di acquisizione, e tra questi spicca COLMAP[27]. COLMAP esegue il task di Structure From Motion, che si occupa di ricostruire angolazione e posizione delle camere virtuali a partire da un processo di triangolazione inversa delle feature estraibili dalle immagini. Se vengono trovate corrispondenze tra le immagini e le pose, queste poi si possono fornire alle architetture NeRF ed SDF in modo tale da ricostruirne la scena e le mesh con le metodologie sopracitate.

Nell'ambito dei dataset Cultural Heritage utilizzati in questo lavoro di tesi, il software COLMAP è stato fondamentale per ricostruire le informazioni di posizione che invece il Synthetic Dataset già forniva dotazione. In particolare, è stata utilizzata un'implementazione interna a Nerfstudio di COLMAP per ricostruire le posizioni delle immagini dei tre siti forniti:

- Il santuario di Macereto, un complesso religioso situato a Visso, nel versante occidentale dei Monti Sibillini
- Il castello di Magalotti nel comune di Fiastra
- Piazza A.Gentili a San Ginesio

Per i primi due dataset, si parla di video 2160p@30fps, di cui sono stati estratti solo i frame chiave, si parla di 725 frame per Magalotti e di 300 frame per Macereto. Nel caso di San Ginesio il dataset è composto dalla totalità delle foto, 224. Questi dataset sono poi stati sottoposti a due tipologie di train-validation split, sulla base del tipo di modello richiesto: Nerfstudio o SDFStudio.
Difatti, Nerfstudio ed SDFStudio utilizzano un approccio differente per la ripartizione del dataset in train e validation: mentre Nerfstudio effettua uno split 90-10, SDFStudio utilizza un approccio differente in quanto tutte le immagini di train vengono utilizzate anche come validation, ma con dei vincoli per utilizzarne solo una percentuale e non tutte. (controllare il significato di sta cosa)

Figura 3.4: Esempio di alcuni frame estratti dal video della scena ambientata al santuario di Macereto.

Figura 3.5: Esempio di alcuni frame estratti dal video della scena ambientata al castello Magalotti.

Figura 3.6: Esempio di alcune immagini facenti parte della collezione fotografica della scena ambientata in piazza A. Gentili a San Ginesio.

3.4 Metriche

Al fine di valutare i risultati ottenuti per ciascun esperimento è necessario riportare varie metriche sulla base delle quali si possa esprimere un giudizio della resa totale. La più semplice e conosciuta nell'ambito del confronto tra immagini è il Peak Signal-to-Noise Ratio(PSNR), una metrica diffusa per valutare la qualità della riproduzione di un'immagine o di un segnale audio dopo essere stato compresso o elaborato. Si misura il rapporto tra il picco del segnale e il rumore, fornendo un'indicazione della fedeltà della riproduzione rispetto all'originale. Questa tecnica utilizza il calcolo dell'errore quadratico medio per ogni punto della differenza tra le due immagini in input, riportando il valore in scala logaritmica.

$$\text{MSE} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} [I(i,j) - K(i,j)]^2 \tag{3.1}$$

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{\text{Picco del segnale}^2}{\text{Errore Quadratico Medio}} \right) \tag{3.2}$$

dove:

- m è il numero di righe dell'immagine.
- n è il numero di colonne dell'immagine.
- I(i,j) rappresenta il valore del pixel nell'immagine originale alla posizione(i,j). K(i,j) rappresenta il valore del pixel nell'immagine ricostruita o compressa alla posizione (i,j).

La formula dell'MSE calcola la differenza quadratica tra ogni coppia di pixel e ne fa la media, fornendo una misura complessiva della discrepanza tra l'immagine originale e quella ricostruita. Un MSE più basso indica una minore discrepanza e, teoricamente, una migliore fedeltà dell'immagine ricostruita rispetto all'originale. Inoltre, il "picco del segnale" nella formula del PSNR si riferisce al valore massimo che un pixel dell'immagine può rappresentare. Questo valore dipende dalla profondità di bit dell'immagine e quindi nel caso di un'immagine RGB il picco del segnale sarà 255^3 (3 canali).

Se PSNR effettua una comparazione a livello di pixel, quindi più "tecnica", Structural Similarity Index(SSIM)[29] è una metrica utilizzata per valutare la qualità di un'immagine, tenendo conto di aspetti come la luminanza, la contrasto e la struttura. A differenza di PSNR, SSIM cerca di modellare la percezione visiva umana in modo più accurato. La formula di base per SSIM è complessa, ma può essere suddivisa in tre componenti principali: luminanza (l), contrasto c, e struttura s

$$\text{SSIM}(x, y) = [l(x, y)]^{\alpha} \cdot [c(x, y)]^{\beta} \cdot [s(x, y)]^{\gamma} \tag{3.3}$$

dove:

- x e y sono le due immagini che si stanno comparando.
- l(x,y) rappresenta la similitudine nella luminanza.
- c(x,y) rappresenta la similitudine nel contrasto.
- s(x,y) rappresenta la similitudine nella struttura.
- I parametri α, β, γ sono costanti che regolano l'importanza relativa di ciascuna componente.

L'ultima metrica che si vuole introdurre è basata su modelli di apprendimento automatico che cercano di modellare la percezione visiva umana (HVS). LPIPS, acronimo di "Learned Perceptual Image Patch Similarity," è una metrica di similarità per valutare la qualità percettuale di immagini. considerando le caratteristiche locali e percettuali. Invece di concentrarsi solo su luminanza, contrasto e struttura, LPIPS utilizza reti neurali profonde per estrarre caratteristiche di alto livello dall'immagine, cercando di catturare aspetti più complessi della percezione umana.
A differenza delle metriche di prima, qui non si ha una formula standard, ma il processo di calcolo di LPIPS coinvolge l'uso di modelli neurali preaddestrati. LPIPS è particolarmente utile in applicazioni in cui la qualità percettuale dell'immagine è critica, ad esempio nell'ambito della generazione di immagini, del deep learning

per la visione artificiale e in scenari in cui le metriche tradizionali potrebbero non riflettere accuratamente la percezione umana.

In ultimo, per il tracciamento delle metriche è stato utilizzato wandb.ai[4], un tool comprensivo anche di versione gratuita per tracciare i dati dell'allenamento di un modello, molto comodo per la fruizione multipiattaforma dei dati.

I risultati ottenuti per Nerfacto e Bakedangelo sono stati ricavati sulla base di esperimenti ripetuti in cui venivano osservate le metriche di output per scegliere il modello migliore.

3.4.1 Training Nerfacto

Train Loss Dict/rgb_loss: Misura la differenza tra i colori predetti per i punti 3D e i colori reali corrispondenti nella scena. Questa loss contribuisce a garantire che i colori della superficie ricostruita siano il più possibile fedeli alla realtà, guidando l'apprendimento dei parametri del modello verso una rappresentazione visivamente coerente.

Train Loss Dict/distortion_loss: Valuta la distorsione della forma superficiale. Questa loss è progettata per penalizzare le discrepanze geometriche tra la superficie prevista e la superficie reale. In questo modo, contribuisce a garantire che la forma della superficie ricostruita si avvicini il più possibile alla forma effettiva degli oggetti nella scena.

Train Loss Dict/interlevel_loss: Questa loss è specifica per il livello finale del modello. Può essere progettata per guidare la convergenza del modello in modo da ottenere risultati accurati e dettagliati nel livello di maggior risoluzione. Ad esempio, potrebbe essere utilizzata per controllare la qualità della ricostruzione nei dettagli fini della superficie.

Train Loss Dict/camera_opt_regularizer: Questa potrebbe essere una loss di regolarizzazione applicata alla parte della loss associata all'ottimizzazione della telecamera durante il training. La regolarizzazione può avere lo scopo di controllare la variazione o l'adattamento eccessivo dei parametri della telecamera, contribuendo così a mantenere stabile e coerente l'apprendimento della telecamera nel contesto della ricostruzione 3D.

[4]Disponibile al sito https://wandb.ai/

3.4.2 Training Bakedangelo

All'interno del contesto dell'allenamento dei modelli di Bakedangelo, la supervisione della rete avviene attraverso diverse componenti di perdita, ciascuna progettata per indirizzare specifici aspetti del processo di apprendimento. Il **Train Loss Dict** rappresenta un insieme di diverse perdite, tra cui **rgb_loss**, **eikonal_loss**, e **curvature_loss**, che sono fondamentali per il conseguimento di una rappresentazione 3D accurata e dettagliata.

Le componenti **rgb_loss** e **interlevel_loss** sono analoghe alla controparte Nerfacto. L'aggiunta fondamentale che hanno implementato i ricercatori di Neuralangelo(implementate poi da Bakedangelo) sono la *eikonalloss* e *lacurvatureloss*. L'**eikonal_loss**, si focalizza sull'addestramento della funzione di distanza firmata (SDF) predetta. Questa perdita enfatizza l'importanza di una SDF valida, in conformità con l'equazione eikonale, che assicura una rappresentazione coerente e accurata delle superfici tridimensionali. L'eikonal_loss contribuisce a stabilire la precisione geometrica della rappresentazione. Infine, la **curvature_loss** gioca un ruolo chiave nella promozione della fedeltà alle caratteristiche curve delle superfici. Questa componente di perdita è progettata per guidare l'allenamento verso la cattura accurata delle curvature, contribuendo a preservare le sfumature dettagliate delle forme geometriche.

Nel contesto degli esperimenti eseguiti, sono state tenute d'occhio ciascuna di queste loss insieme con la evaluation loss (che valuta eventuali overfitting del modello), ed in figura 3.7 si possono osservare gli output delle loss di training definitive per i modelli svolti.

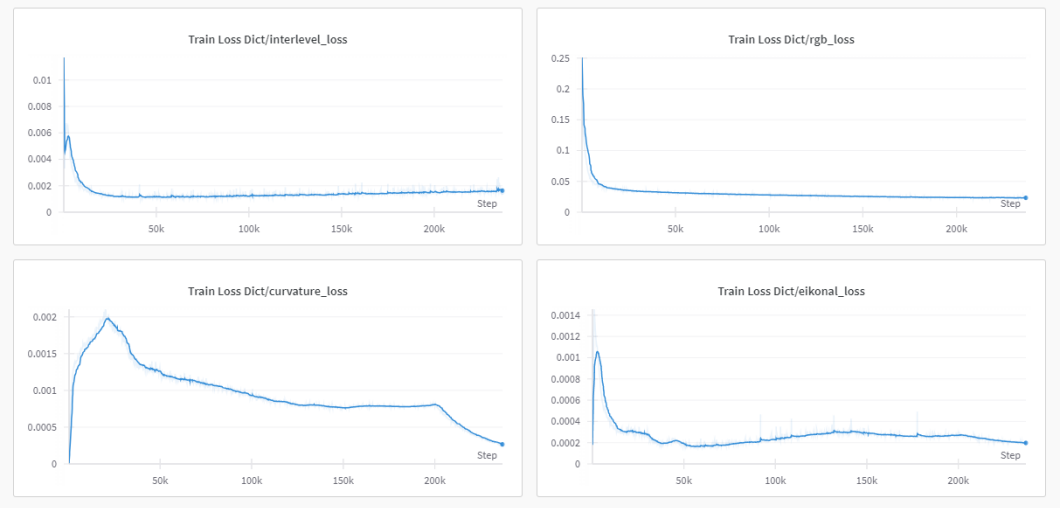

Figura 3.7: **Grafico delle varie loss prese in considerazione.**

In particolare, si è cercato un compromesso tra l'inizio della risalita della *interlevel_loss* e l'improvvisa discesa della *curvature_loss*. In particolare, si è lasciato allenare il modello per un tempo molto più lungo della sua normale durata, per calibrare quanto questa loss potesse scendere prima di raggiungere valori trascurabili. In Figura

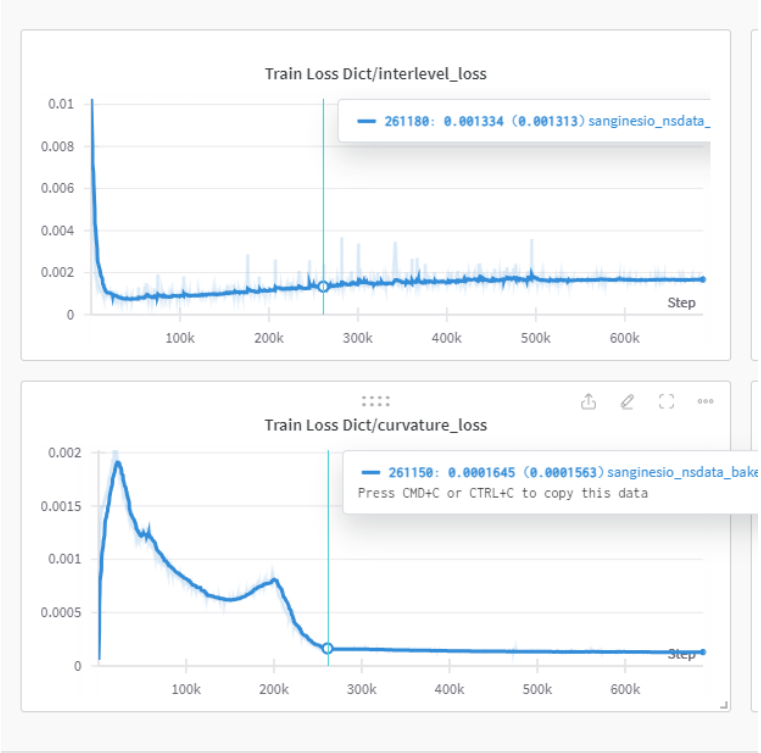

Figura 3.8: **Grafico interlevel-loss vs curvature-loss**

3.8 si può notare come la *curvature*$_l$*oss* smetta di decrescere intorno all'iterazione 260.000. Empiricamente quindi, per tutti gli esperimenti, sulla base delle metriche PSNR,SSIM ed LPIPS in uscita si sono scelti i valori di iterazioni ottimali su cui applicare un early stopping dell'allenamento.

Per quanto riguarda gli allenamenti con i modelli Gaussian Splatting, non sono stati utilizzati strumenti di tracciamento in quanto non configurati. I training sono stati portati avanti seguendo le specifiche di iterazioni consigliate nelle relative repositories di progetto.

Capitolo 4

Esperimenti e Risultati

4.1 Descrizione degli Esperimenti

Per condurre i nostri esperimenti, si sono sfruttati una varietà di dispositivi, ognuno dei quali ha contribuito in modo significativo alla realizzazione dei risultati. Alcuni esperimenti, tra cui quelli relativi a Nerfacto, sono stati eseguiti su una macchina consumer dotata di una scheda grafica GTX1080 (8GB VRAM). Pur dimostrando la capacità di raggiungere la convergenza, è importante sottolineare che la velocità di tale convergenza risultava inferiore rispetto alle prestazioni offerte dalla scheda RTX A6000 (50GB VRAM). Quest'ultima è stata messa a disposizione dal dipartimento VRAI e ha giocato un ruolo cruciale nell'esecuzione di esperimenti più complessi, in particolare quelli relativi a Bakedangelo e Gaussian Splatting.

La macchina su cui operava la RTX A6000 è basata su un sistema Linux(Ubuntu), su cui è stato creato un container Docker appositamente configurato per l'esecuzione degli esperimenti. Questo approccio ha fornito un ambiente isolato e gestibile, garantendo la riproducibilità e la facilità di gestione degli esperimenti. L'accesso alla macchina avveniva tramite connessione SSH, consentendo al team di ricerca di interagire con il container Docker in modo remoto.

La potente architettura della scheda RTX A6000 ha offerto significativi vantaggi in termini di potenza di calcolo, consentendo convergenze più veloci degli esperimenti e l'analisi di modelli più complessi. L'utilizzo del container Docker ha semplificato il processo di distribuzione e gestione degli esperimenti, contribuendo alla replicabilità dei risultati. La combinazione di hardware avanzato e ambiente di esecuzione ottimizzato ha quindi costituito una soluzione completa e affidabile per affrontare sfide computazionali più impegnative.

4.1.1 Nerfstudio

Questi esperimenti sono relativi alla metodologia Nerfacto, andando quindi a verificare come si comportano i metodi a funzione di densità volumetrica. Come anticipato, questi approcci vanno a modellare ciascun punto dello spazio con una funzione di densità volumetrica che rappresenta quanta "materia" c'è in un determinato punto: maggiore sarà il valore, più si sta parlando di un punto denso/agglomerato. (todo aggiungere informazioni su configurazione macchine e schede video). Il primo step per l'allenamento del modello è stato quindi andare a ad effettuare un preprocessing.

Per effettuare il preprocessing ed il porting del dataset in un formato compatibile con Nerfstudio, è presente il comando $ns - process - data$ che, una volta fornito il dataset effettua il task di structure from motion, restituendo su un percorso di destinazione il modello risultante, insieme con un numero di copie del dataset di cui è stato fatto un downsize: queste serviranno a Nerfstudio nel caso in cui le immagini siano troppo grandi, scegliendo quindi autonomamente le immagini di dimensioni adeguate.

L'allenamento è stato lanciato attraverso il comando $ns - train$ che si occupa di avviare il processo.

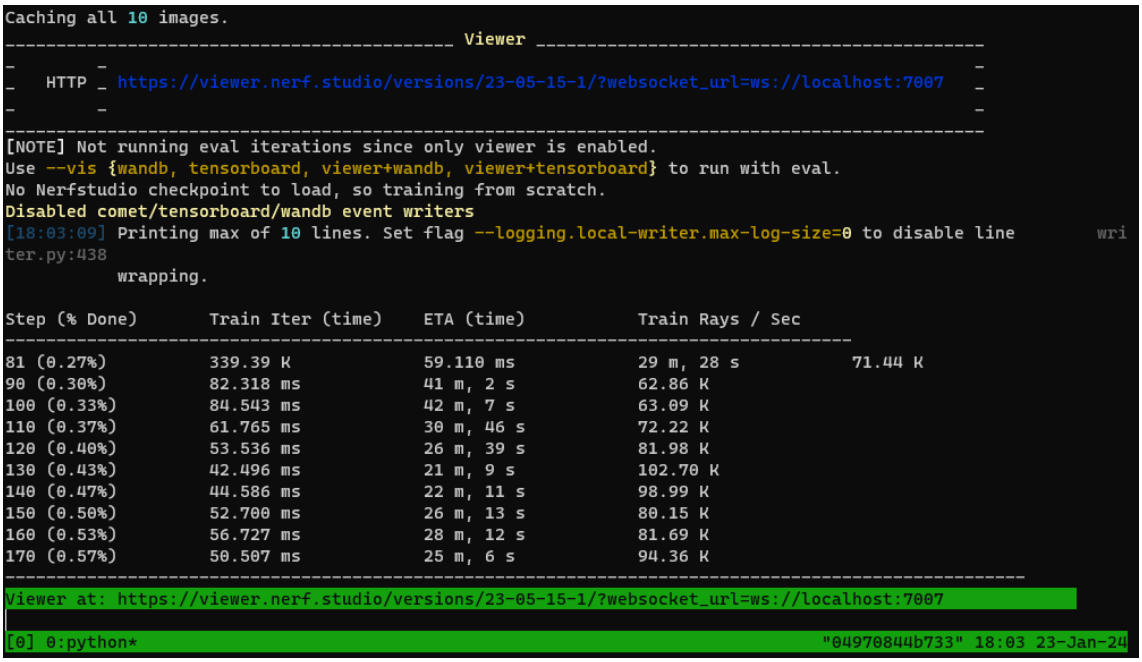

Figura 4.1: **Finestra di allenamento di Nerfstudio.**

Come si può osservare in figura 4.1, il framework Nerfstudio si presenta in maniera molto intuitiva con una schermata contenente il progresso di allenamento, ed il link

al viewer che permette la visualizzazione in tempo reale del progresso di training della scena.

Una volta terminato il processo di allenamento impostato a 30.000 iterazioni, la scena è completamente esplorabile dal viewer, ma sopratutto è possibile iniziare il processo di estrazione mesh: Attraverso il comando $ns - export$ è possibile estrarre la mesh scegliendo il metodo che più si ritiene opportuno. Nel nostro caso avendo una point-cloud è stato scelto Poisson.

4.1.2 SDFStudio

Per gli esperimenti con SDFStudio, è presente l'analogo comando ns-train con la stessa interfaccia di Nerfstudio. Per l'esportazione di mesh c'è una piccola variante: essendo i metodi basati su SDF predefiniti per fornire un output in mesh, non è presente il comando $ns-export$ ma uno ad-hoc: $ns-extract-mesh$. La metodologia di estrazione è quella basata su marching cubes.

4.1.3 Gaussian Splatting

Per gli esperimenti legati al Cultural Heritage, infine, sono stati svolti degli esperimenti aggiuntivi relativi all'approccio Gaussian Splatting di cui si è parlato nel capitolo relativo allo stato dell'arte. Questo approccio si interpone tra Nerfacto e Bakedangelo come compromesso tra i tempi di computazione e qualità della scena risultante. Peculiarità del gaussian splatting è la generazione di mesh di qualità migliore rispetto ai metodi NeRF tradizionali, e con texture di molto alto livello di dettaglio. Solo per questo approccio quindi verranno riportate le scene complete anche di texture.

4.2 Nerf Synthetic

4.2.1 Lego Object

Nerfacto

Come anticipato, gli esperimenti iniziali sono stati effettuati sul dataset nerf Synthetic. Di fatto, il nerf synthetic dataset fornisce le posizioni di scatto già

annotate, ma al fine di rendere gli esperimenti omogenei dal punto di vista del processo di structure from motion, è stato necessario effettuarne il preprocessing con COLMAP.

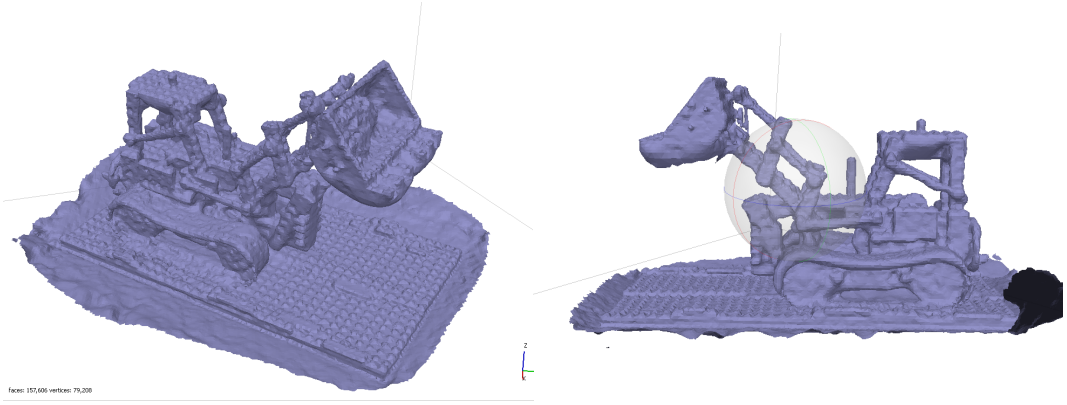

Figura 4.2: **Mesh Lego(Nerfacto).**

Nelle figure 4.2 è possibile osservare l'output mesh corrispondente al modello lego. E' possibile notare come la geometria di fondo non sia sbagliata, ma risulta grossolana ed in alcuni punti ci sono agglomerati che nelle immagini originali non sono presenti. Il numero di vertici risultante è di circa 80000 e l'allenamento è durato circa 30 minuti.

Bakedangelo

Per l'allenamento in Bakedangelo i parametri cambiano, sono state portate a 230000 le iterazioni ed i tempi di allenamento crescono fino a diventare 26 ore.

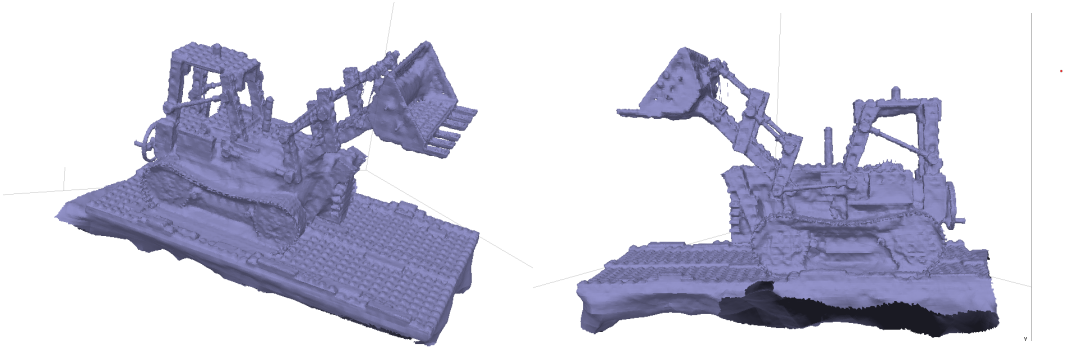

Figura 4.3: **Mesh Lego con Bakedangelo.**

In figura 4.3 è possibile notare come il miglioramento sia netto per quanto riguarda i tasselli lego, molto più definiti rispetto alla controparte Nerfacto. Anche i cingoli presentano più dettagli ed il numero di artefatti agglomerati è minore. Anche il

numero di vertici aumenta a circa 450000, giustificando anche l'aumento di qualità e dettaglio.

Comparazione

Nella seguente tabella vengono riportate le configurazioni e le proprietà del training in questione.

Lego	Caratteristiche		
	Iterazioni	Tempo di Training	Numero di Vertici
Nerfacto	30000	30 minuti	80000
Bakedangelo	230000	25 ore	450000

Esperimento	PSNR	SSIM	LPIPS
Nerfacto	15.02	0.49	0.39
Bakedangelo	15.02	**0.66**	**0.32**

Tabella 4.1: Confronto dei risultati tra gli approcci "nerfacto" e "bakedangelo" nell'esperimento "lego" in termini di PSNR, SSIM e LPIPS. I valori sono troncati alla seconda cifra decimale.

E' possibile notare la minima discrepanza per quanto riguarda il PSNR, ma per quanto riguarda le altre due metriche si hanno netti miglioramenti: +34.96% SSIM e -17.96% per LPIPS.

4.2.2 Chair

Nerfacto

La seconda scelta fatta nel Nerf Synthetic dataset è stata quella riguardante l'oggetto "chair". Analoghi esperimenti e preprocessing sono stati eseguiti anche per questo oggetto, e si riportano le mesh risultanti per il modello nerfacto in figura 4.4.

Analogamente all'oggetto Lego, la forma è buona ma vengono a mancare i dettagli ad alta frequenza. Il numero di vertici si attesta intorno ai 30000.

Figura 4.4: **Mesh dell'oggetto chair estratta da nerfacto**

Bakedangelo

La controparte Bakedangelo presenta a primo impatto una forma molto più dettagliata ma non solo: braccioli, schienale, gambe, riportano i dettagli con molta più cura, delineando il tessuto dallo scheletro della sedia. Il modello è stato allenato per 25 ore a 260000 iterazioni, per un incremento nel numero di vertici a 165000, quasi 6 volte tanto.

Figura 4.5: **Mesh dell'oggetto chair estratta da bakedangelo**

Comparazione

Nella seguente tabella vengono riportate le configurazioni e le proprietà del training in questione.

Chair	Caratteristiche		
	Iterazioni	Tempo di Training	Numero di Vertici
Nerfacto	30000	30 minuti	30000
Bakedangelo	260000	25 ore	165000

Esperimento	PSNR	SSIM	LPIPS
Nerfacto	11.25	0.39	0.45
Bakedangelo	**13.65**	**0.79**	**0.25**

Tabella 4.2: Confronto dei risultati tra gli approcci "nerfacto" e "bakedangelo" nell'esperimento "chair" in termini di PSNR, SSIM e LPIPS. I valori sono troncati alla seconda cifra decimale.

Anche per l'oggetto chair i valori subiscono un notevole aumento: PSNR +23.33%, +102.56% SSIM e -44.44% per LPIPS.

4.3 Cultural Heritage

Come discusso antecedentemente, gli elementi fin'ora osservati sono appartenenti al dataset "Synthetic", realizzati ad hoc per l'allenamento con queste reti. Al fine di dimostrare la validità dei metodi in questo studio, si è mostrata necessaria l'esigenza di testarli su dataset personalizzati: si è optato per un insieme di dataset relativi al Cultural Heritage. Questa scelta è stata fatta in relazione ad un altro studio[1] proposto da un tesista dell'UNIVPM relativo alle reti NeRF tradizionali in collaborazione con il VRAI. Stilare un confronto tra le metodologie di quello studio e quelle SDF proposte qui è interessante vista l'esigenza di raccogliere mesh di alta qualità oltre ad un rendering fine a se stesso della scena.

4.3.1 Macereto

La prima operazione è stata quella di preprocessing, e si è puntato ad un target di immagini pari a 300. COLMAP ha quindi effettuato un processo di structure from motion andando a prelevare 300 degli oltre 8000 frame del video, facendone anche un downsample a fullHD(1080p).

Nerfacto

Di seguito, si riportano i risultati di training estraendo le mesh dal modello Nerfacto:

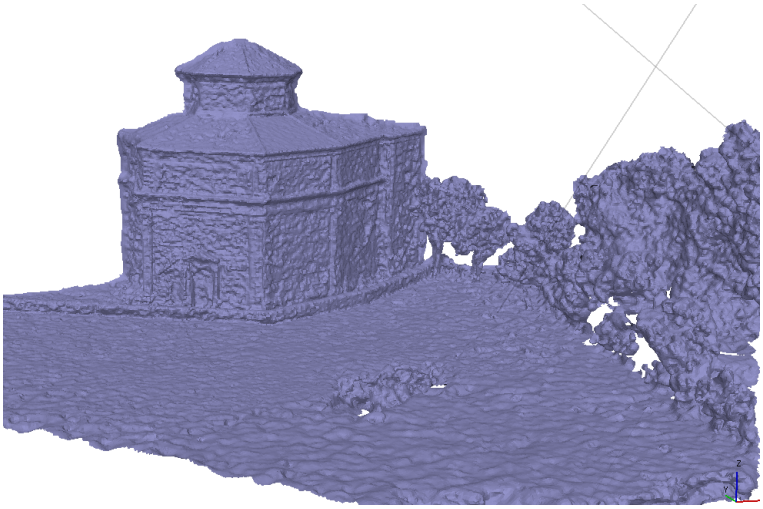

Figura 4.6: **Mesh del dataset Macereto estratta da nerfacto**

Facendo riferimento all'immagine 4.6, è possibile vedere come la mesh risultante è grossolana, seppur riporti la scena nella sua quasi-interezza. Le mura ad esempio sono piene di agglomerati e molto frastagliate. Anche il tetto (Figura 4.7) è privo di dettagli, che si potranno osservare meglio nella modellazione con Bakedangelo. Il modello è stato allenato per 35 minuti, a 30000 iterazioni con un numero di vertici pari a 330000.

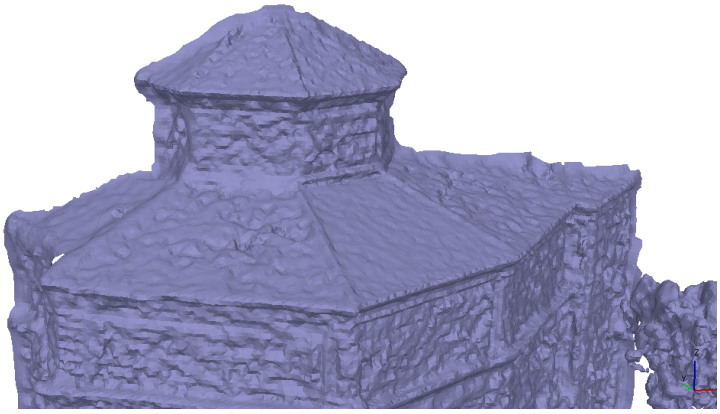

Figura 4.7: **Tetto ravvicinato della mesh Macereto.**

Bakedangelo

Il training con Bakedangelo è durato circa 26 ore per un totale di 240000 iterazioni. Il numero di vertici risultanti è di 2.600.000, a parità di grandezza della scena.

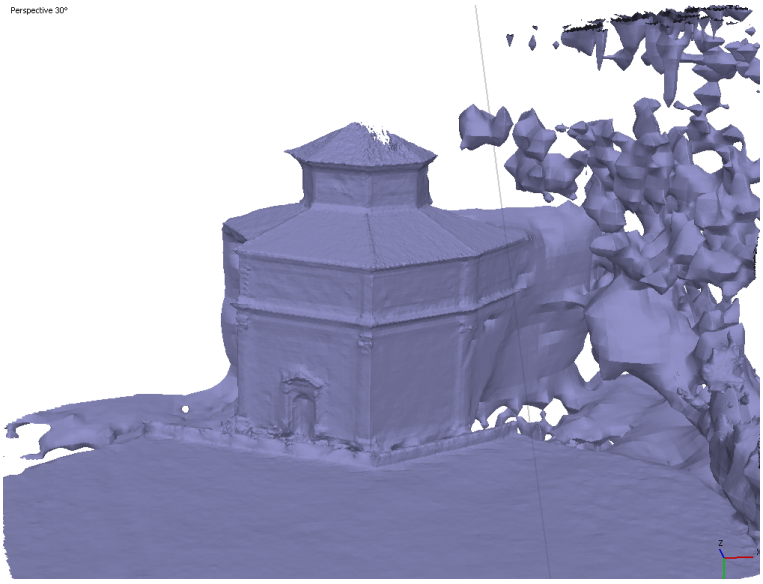

Figura 4.8: **Mesh Macereto (Bakedangelo).**

Dalla figura 4.8, si può riscontrare immediatamente come le superfici siano molto più ben definite e prive di agglomerati. La vegetazione, tuttavia, viene modellata male negli esperimenti eseguiti, ma per l'uso che ne è stato scelto non è molto rilevante, essendo il soggetto della mesh il santuario in se. In contrapposizione alla figura 4.7, la modellazione del tetto eseguita con Bakedangelo 4.9 è molto più dettagliata rappresentando tegole e mattoni, andando a migliorare la realisticità della mesh.

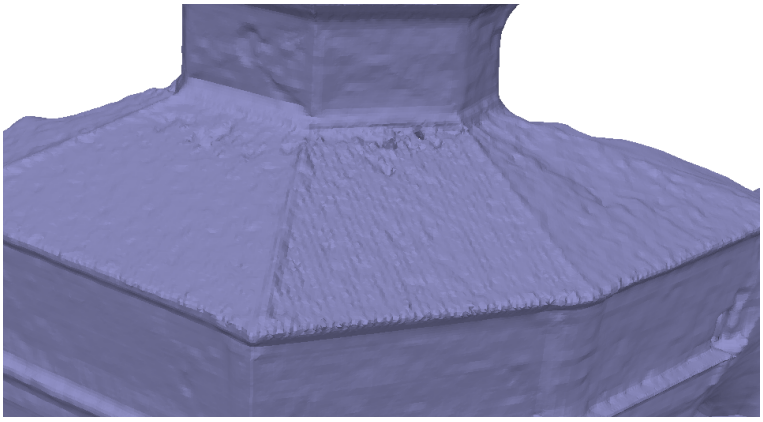

Figura 4.9: **Tetto ravvicinato della mesh Macereto (Bakedangelo).**

Gaussian Splatting

Figura 4.10: **Mesh del dataset Macereto estratta da Gaussian Splatting.**

Figura 4.11: **Immagini ravvicinate della mesh texturizzate di Macereto (Gaussian Splatting).**

L'approccio gaussian splatting genera delle mesh simili a quelle in Nerfacto, ma con la differenza che riesce a modellare meglio la scena, anche con dataset di qualità non ottimale. Dalle figure 4.10 e 4.11 si può notare come sia evidente anche il chiostro del santuario, che non viene ricostruito da nessuno degli approcci precedenti.

Comparazione

Anche per il dataset Macereto i valori subiscono un notevole miglioramento: PSNR +28.12%, +35.38% SSIM e -57.14% per LPIPS. Il numero di vertici aumenta di circa 8 volte, aumentando la complessità della mesh ma anche la qualità. L'approccio Gaussian splatting si comporta meglio nei confronti di Nerfacto sulle metriche PSNR e SSIM, rispettivamente +29.53% e +29.23%. Per LPIPS invece le prestazioni sono lievemente peggiori, +23.81%.

Macereto	Caratteristiche		
	Iterazioni	Tempo di Training	Numero di Vertici
Nerfacto	30000	35 minuti	330.000
Bakedangelo	260000	25 ore	2.600.000
Gaussian Splatting	7000	1 ora	400.000

Esperimento	PSNR	SSIM	LPIPS
Nerfacto	21.38	0.65	0.21
Bakedangelo	27.42	**0.88**	**0.09**
Gaussian Splatting	**27.70**	0.84	0.26

Tabella 4.3: Confronto dei risultati tra gli approcci "nerfacto" , "bakedangelo" e "gaussian splatting" nel dataset "macereto" in termini di PSNR, SSIM e LPIPS. I valori sono troncati alla seconda cifra decimale.

4.3.2 San Ginesio

Il prossimo dataset utilizzato è stato quello di San Ginesio. L'oggetto della scena è la statua presente al centro della piazza. Il dataset è composto di 224 immagini di cui è è stato effettuato uno split 90% training e 10% validation.

Nerfacto

Non si notano particolari discrepanze dagli altri modelli, il livello di dettaglio è grossolano ma la forma è buona. La base della statua riporta la forma angelica e le principali geometrie , e la rappresentazione della persona riporta alcune agglomerazioni con livello di dettaglio basso. (Figura 4.13)
Il modello è stato allenato per 25 minuti a 30000 iterazioni, per un totale di 340000 vertici.

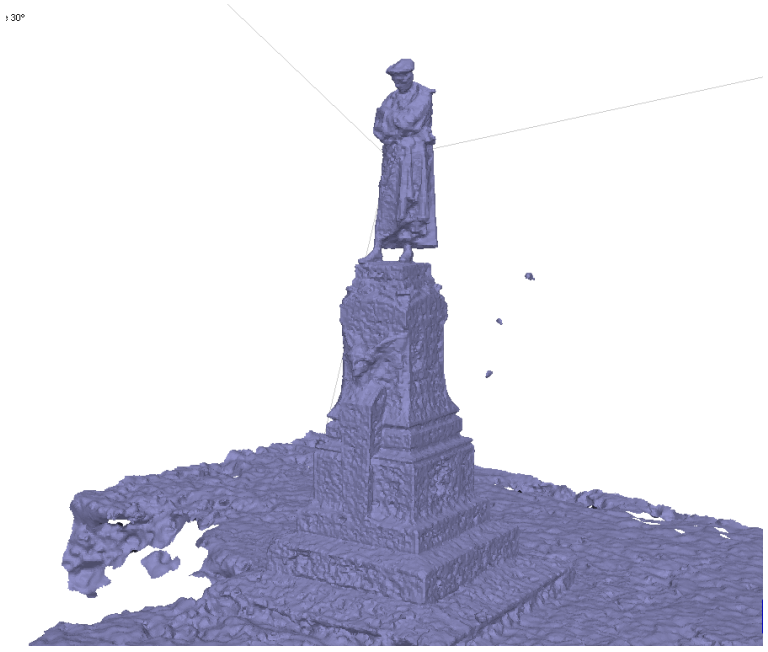

Figura 4.12: **Mesh del dataset Macereto estratta da nerfacto.**

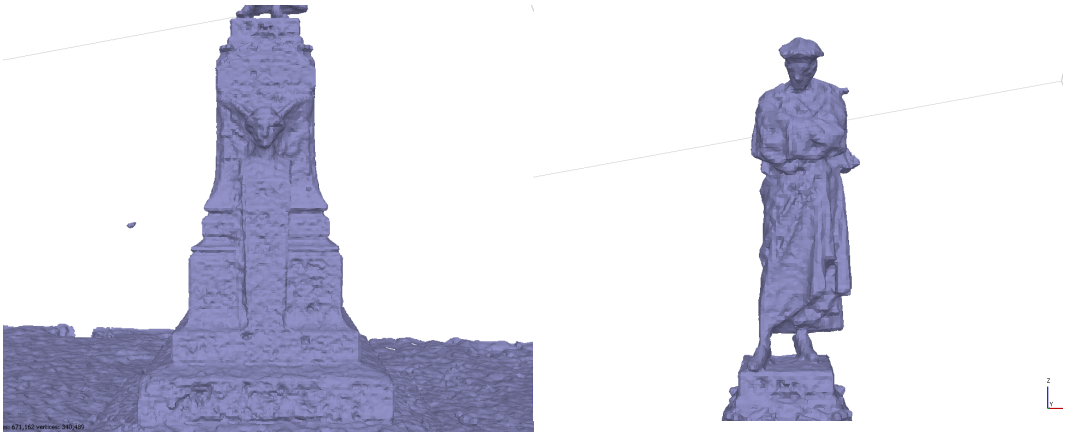

Figura 4.13: **Immagini ravvicinate della mesh di San Ginesio (Nerfacto).**

Bakedangelo

Il modello è stato allenato per 25 ore a 230000 iterazioni, per un totale di 880000 vertici. La statua nella sua interezza è molto più definita: sia il piedistallo che la persona riportano dettagli non presenti nella mesh ricavata da Nerfacto. La superficie ricostruita è molto più omogenea e riportante la maggior parte dei dettagli. Anche la tunica della persona è stata ricostruita in maniera più definita facendola sembrare di tessuto piuttosto che una sua modellazione.

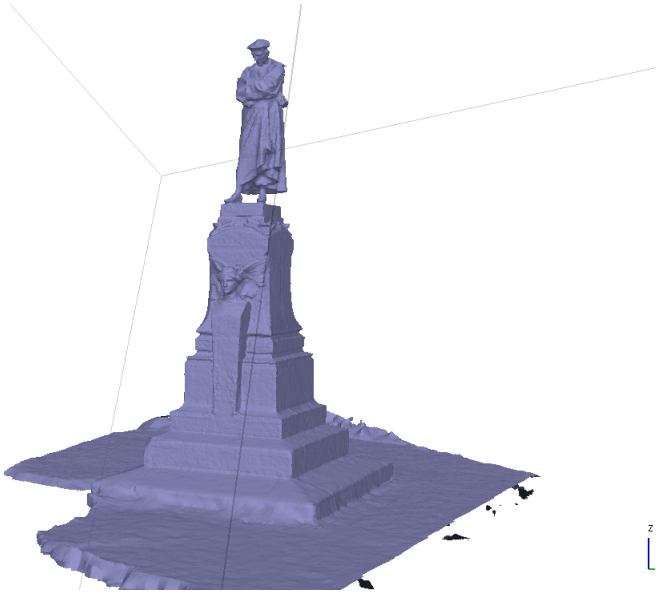

Figura 4.14: **Mesh del dataset San Ginesio estratta da bakedangelo.**

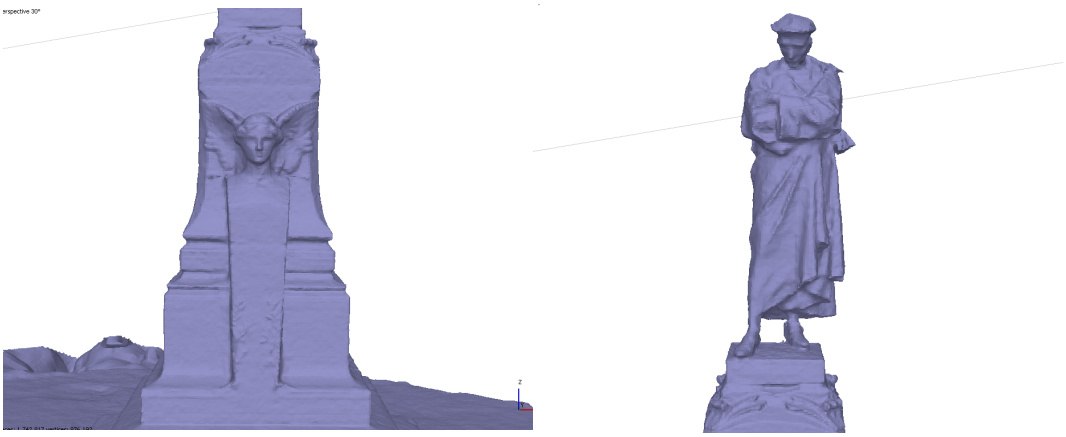

Figura 4.15: **Immagini ravvicinate della mesh di San Ginesio (Bakedangelo).**

Gaussian Splatting

Anche qui, la ricostruzione di gaussian splatting riesce molto bene a ricavare la quasi interezza della geometria della scena, fornendo anche una texture molto dettagliata, anche per quanto riguarda il background. La mesh tuttavia risente di alcuni difetti nella parte bassa della statua (Figura 4.17) in cui sono presenti fori che non fanno parte della statua originaria. Il modello è stato allenato per 1 ora a 7000 iterazioni, con circa 360000 vertici.

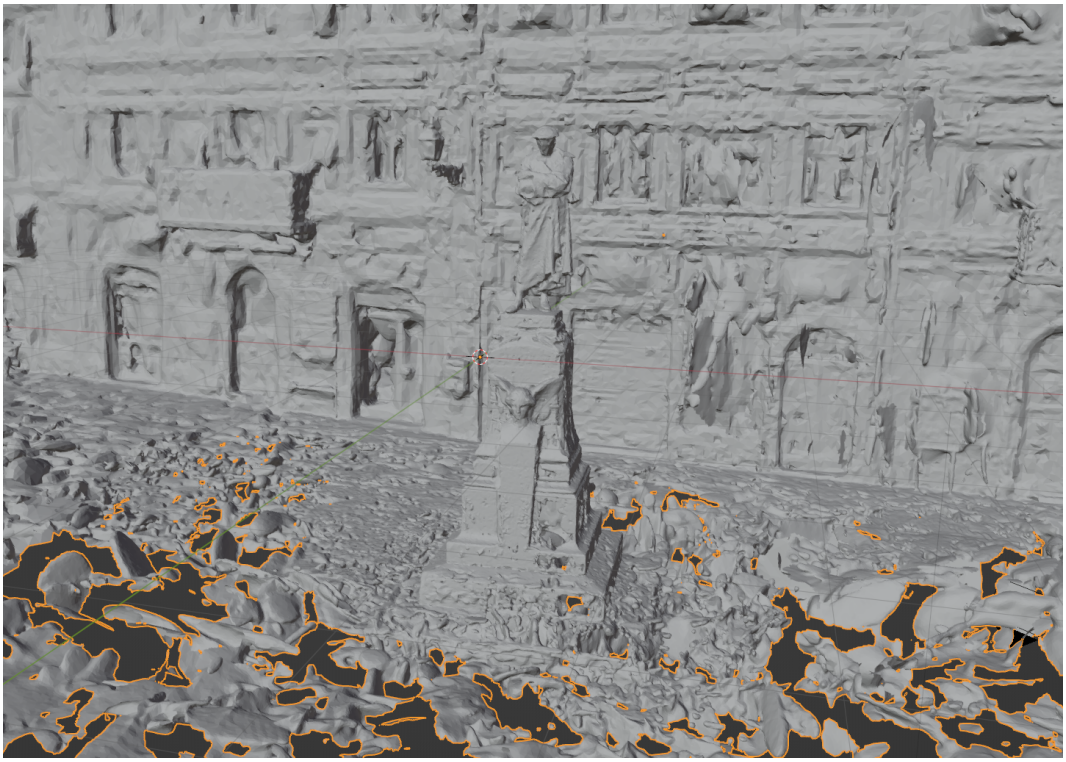

Figura 4.16: **Mesh del dataset San Ginesio estratta da Gaussian Splatting.**

Figura 4.17: **Immagini ravvicinate della mesh texturizzate di San Ginesio (Gaussian Splatting).**

Comparazione

Per il dataset San Ginesio i valori delle metriche riportano: PSNR +10.33%, +30.51% SSIM e -61.90% per LPIPS, tutti in favore di Bakedangelo. Il numero di vertici aumenta di poco più del doppio, ma facendone un utilizzo di netto migliore

SanGinesio	Caratteristiche		
	Iterazioni	Tempo di Training	Numero di Vertici
Nerfacto	30000	25 minuti	340000
Bakedangelo	230000	25 ore	880000
Gaussian Splatting	7000	1 ora	360000

Esperimento	PSNR	SSIM	LPIPS
Nerfacto	17.61	0.59	0.42
Bakedangelo	19.43	**0.77**	**0.16**
Gaussian Splatting	**20.16**	0.76	0.31

Tabella 4.4: Confronto dei risultati tra gli approcci "nerfacto" , "bakedangelo" e "gaussian splatting" nel dataset "San Ginesio" in termini di PSNR, SSIM e LPIPS. I valori sono troncati alla seconda cifra decimale.

rispetto alla controparte Nerfacto. Gaussian Splatting supera Nerfacto in maniera significativa su PSNR (+14.68%) e SSIM (+29.66%), sottolineando una maggiore fedeltà e somiglianza strutturale delle immagini generate. Tuttavia, in termini di LPIPS, Gaussian Splatting mostra un leggero degrado delle prestazioni rispetto a Nerfacto, con un aumento del +26.19%. Questo indica che, nonostante le migliorate prestazioni generali, il metodo potrebbe comportarsi leggermente peggio in termini di percezione dell'errore rispetto a Nerfacto in questo contesto specifico.

4.3.3 Magalotti

In ultimo, gli esperimenti sul dataset Magalotti. Questi esperimenti sono quelli che hanno riportato il peggior risultato per via delle limitazioni del dataset in se: le angolazioni di cattura del video sono molto poche e ripetitive, portando quindi ad una non coerenza del modello con la scena originale.

Nerfacto

Come anticipato, la rappresentazione non è delle migliori: c'è confusione, molti agglomerati per modellare gli alberi, il castello è solo parzialmente visibile. E' facilmente intuibile quali siano l'angolazione e l'orientamento di scatto puramente frontali. Eppure, contrariamente ai modelli precedenti, il modello che visivamente performa meglio la scena in se è proprio Nerfacto. Questo modello è stato allenato per 30 minuti, a 30000 iterazioni, con 285000 vertici risultanti.

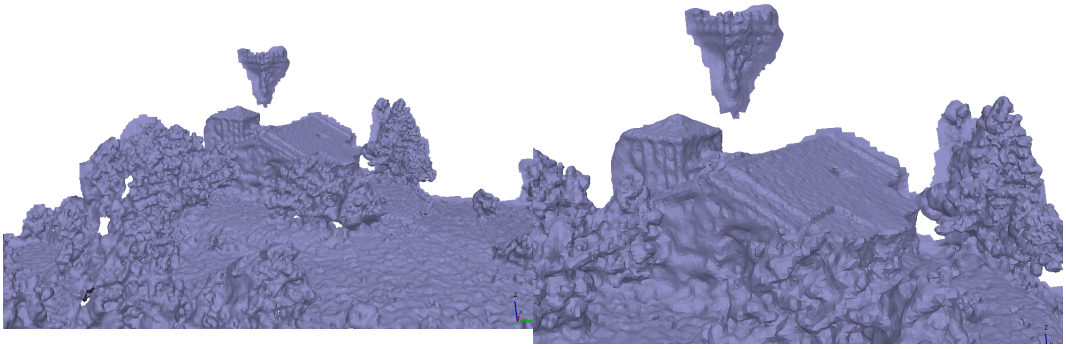

Figura 4.18: **Mesh del dataset Macereto estratta da Nerfacto.**

Bakedangelo

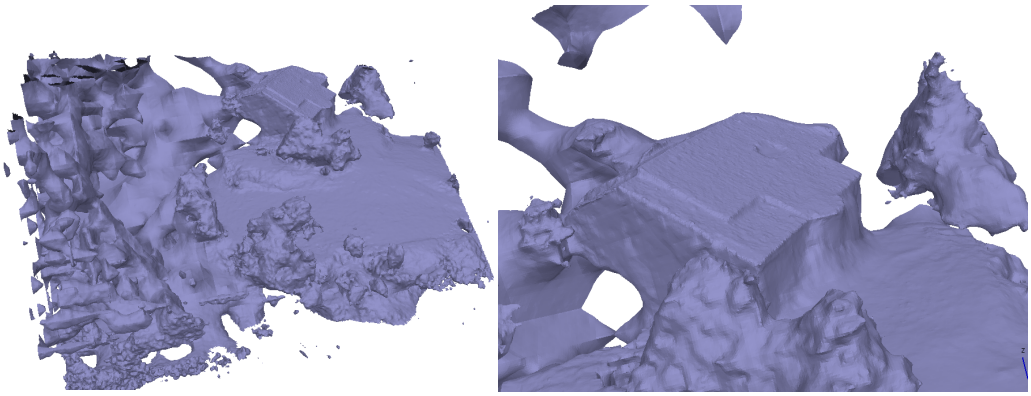

Figura 4.19: **Mesh del dataset Macereto estratta da bakedangelo.**

La problematica principale con questo modello, è che necessita di molte angolazioni per la ricostruzione fedele della scena. Si riesce quindi a sottolineare come con dataset non ottimali dal punto di vista degli scatti, bakedangelo performi male visivamente, seppur le metriche lo diano come vincitore. Si suppone che questa vittoria sia data dal fatto che le superfici che il modello è riuscito a ricostruire siano comunque migliori delle ricostruzioni eseguite da nerfacto. Il modello è stato allenato per 26 ore a 240000 iterazioni, per un totale di circa 2000000 di vertici.

Gaussian Splatting

In Figura 4.20 è possibile osservare la mesh texturizzata del dataset Magalotti. A primo impatto si riesce a notare che anche qui la scena è catturata nella sua interezza, andando a generare una texture che la ricopre totalmente. Andando però ad avvicinarsi alla scena (Figura 4.21) i colori diventano più disomogenei ed approssimati, riportandosi al difetto del dataset di cui si è parlato in precedenza: la

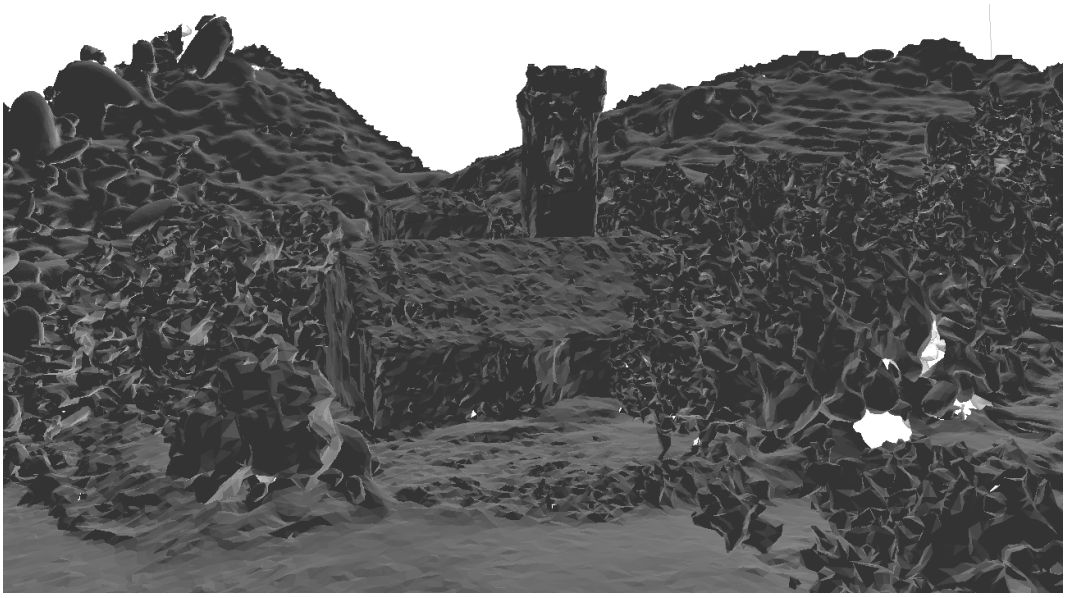

Figura 4.20: **Mesh texturizzate del dataset Magalotti estratta da Gaussian Splatting.**

Figura 4.21: **Immagini ravvicinate della mesh texturizzate del dataset Magalotti (Gaussian Splatting).**

mancanza di varietà nelle angolazioni di scatto. Più avanti nella comparazione si discuterà delle metriche che daranno riscontro a quanto appena discusso.

Comparazione

Le metriche riportano: PSNR +6.16%, +9.98% SSIM e -53.85% per LPIPS, tutti in favore di Bakedangelo.Gaussian Splatting, pur mostrando un risultato rispettabile con un PSNR di 24.17, evidenzia un calo significativo nelle metriche di similarità. La SSIM scende a 0.71, suggerendo una minore coerenza strutturale rispetto agli altri metodi, mentre il valore di LPIPS aumenta a 0.33, indicando un incremento nella percezione dell'errore percettivo. In confronto con Nerfacto, Gaussian Splatting presenta una

Magalotti	Caratteristiche		
	Iterazioni	Tempo di Training	Numero di Vertici
Nerfacto	30000	30 minuti	258000
Bakedangelo	240000	26 ore	2000000
Gaussian Splatting	7000	1 ora	400000

Esperimento	PSNR	SSIM	LPIPS
Nerfacto	26.72	0.81	0.13
Bakedangelo	**28.37**	**0.89**	**0.06**
Gaussian Splatting	24.17	0.71	0.33

Tabella 4.5: Confronto dei risultati tra gli approcci "nerfacto" , "bakedangelo" e "gaussian splatting" nel dataset "Magalotti" in termini di PSNR, SSIM e LPIPS. I valori sono troncati alla seconda cifra decimale.

performance inferiore su PSNR (-9.52%) , SSIM (-12.35%) e LPIPS (+153.85%). Nonostante alcune metriche oggettive indichino un risultato apparentemente inferiore per Gaussian Splatting rispetto a Nerfacto, l'osservazione visuale rivela un quadro più complesso e favorevole per Gaussian Splatting. In particolare, la percezione visiva suggerisce un miglioramento sostanziale nella completezza della scena con Gaussian Splatting rispetto a Nerfacto. La sua capacità di adeguarsi alla complessità della mesh, contribuisce a un risultato complessivo più omogeneo. Riscontriamo in questo particolare caso che Gaussian Splatting ha delle performance generalmente migliori per quanto riguarda la ricostruzione della totalità della scena, rispetto al singolo focus che potrebbe essere ad esempio il castello di Magalotti.

4.4 CAVE

Un ultima parte della sezione esperimenti riguarda il testing su ambiente CAVE. Tuttavia si è rivelata necessaria una fase preliminare di ripulitura della mesh da artefatti indesiderati, in maniera tale da poter esplorare la scena nel massimo della comodità. Questa ripulitura è stata effettuata attraverso il software Metashape che permette un'intuitiva modifica della mesh, selezionando le porzioni che si vogliono tagliare via.

Le mesh sono state importate in blender e successivamente, attraverso il software TechViz si è scelta la finestra da virtualizzare (in questo caso Blender), e ci si è preparati per immergersi nell'ambiente virtuale.

La sperimentazione ha permesso di evidenziare come l'immersività e la possibilità di esplorare la scena in completa libertà riesca a dare valore aggiunto alla mesh

estratta in precedenza. Tra i vari sperimentatori, si sottolinea come l'esplorazione della scena abbia permesso di notare dettagli che a primo impatto non si sono notati nella scena reale con la statua davanti, confermando quindi l'utilità di questo approccio nell'ambito studiato. In figura 4.22, alcune immagini di sperimentazione su CAVE.

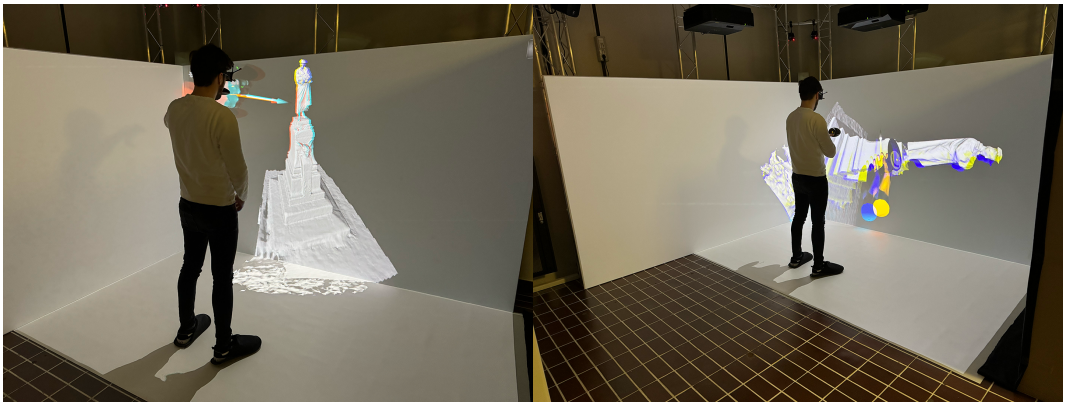

Figura 4.22: **Sperimentazione in CAVE.**

4.5 Analisi dei risultati

I risultati degli esperimenti, come riassunto nelle tabelle 4.6 e 4.7, forniscono una panoramica dettagliata delle prestazioni dei metodi di generazione di mesh Nerfacto, Bakedangelo e Gaussian Splatting. Analizzando attentamente i dati, emergono considerazioni significative che possono orientare la scelta del metodo più adatto a specifiche esigenze di applicazione.

In termini di destinazioni di utilizzo, il metodo NeRF si distingue per l'esplorazione veloce di scene 3D, ma mostra limitazioni nei dettagli delle mesh critici. La sua elevata velocità di esecuzione lo rende adatto a contesti in cui la resa visiva della scena è prioritaria rispetto ai dettagli intricati della mesh. D'altra parte, l'approccio SDF eccelle nella ricostruzione di mesh ad alta fedeltà per scene complesse, ideale per applicazioni che richiedono dettagli precisi, come lo studio della degradazione di monumenti o l'utilizzo in videogiochi. Tuttavia, va notato che tale precisione comporta tempi di computazione elevati.

Gaussian Splatting, invece, si posiziona come un compromesso versatile tra velocità di computazione e qualità della mesh. La sua moderata velocità di esecuzione lo rende adatto a una vasta gamma di utilizzi, mentre la buona qualità della mesh

lo rende un'opzione interessante per applicazioni che richiedono un equilibrio tra dettaglio e efficienza computazionale.

Metodo	**Destinazioni di Utilizzo**	**Caratteristiche Principali**
NeRF	Esplorazione veloce di scene 3D, dettagli mesh non critici.	Velocità elevata, buona resa visiva delle scene, dettaglio mesh limitato.
SDF	Ricostruzione di mesh ad alta fedeltà per scene complesse, studio della degradazione di monumenti, utilizzo in videogames.	Mesh di alto livello, dettagli precisi, tempo di computazione elevato.
Gaussian Splatting	Compromesso tra velocità di computazione e qualità della mesh, versatile in molteplici contesti.	Velocità di computazione moderata, buona qualità della mesh, adatto a una vasta gamma di utilizzi.

Tabella 4.6: Confronto tra i metodi Nerfacto, Bakedangelo e Gaussian Splatting in base alle destinazioni di utilizzo e alle caratteristiche principali.

Metodo	**Dettagli**	**Tempi di Allenamento**	**Num. Vertici**
Nerfacto	Molto Veloce	Breve (~30 minuti)	Basso
Bakedangelo	Alti dettagli	Lungo(~24 ore)	Alto
Gaussian Splatting	Ottime Texture	Breve(~1 ora)	Basso

Tabella 4.7: Riassunto delle caratteristiche principali dei tre metodi di generazione di mesh.

Guardando più da vicino alle caratteristiche principali e ai tempi di allenamento dei singoli metodi, emerge una chiara differenziazione. Nerfacto si distingue per la velocità di generazione della mesh, richiedendo tempi di allenamento brevi (circa 30 minuti) e producendo mesh con un numero relativamente basso di vertici. Bakedangelo, d'altra parte, eccelle nella generazione di dettagli, ma richiede un tempo di allenamento significativamente più lungo (circa 24 ore) e produce mesh con un alto numero di vertici. Gaussian Splatting, con tempi di allenamento brevi (circa 1 ora) e un basso numero di vertici, rappresenta un equilibrio ragionevole tra dettaglio e efficienza.

Capitolo 5

Conclusioni e Sviluppi Futuri

5.1 Conclusioni

In questo lavoro di ricerca, ci si è dedicati all'esplorazione di approcci innovativi per la ricostruzione tridimensionale di scene da immagini bidimensionali. Si sono valutati tre modelli di generazione di mesh: Nerfacto, Bakedangelo e Gaussian Splatting, confrontandone i risultati attraverso metriche oggettive come PSNR, SSIM e LPIPS, insieme a valutazioni visive. L'obiettivo era quello di realizzare un tracciato per chiunque si debba avventurate nel campo della ricostruzione di scene 3D, sapendo quali tecnologie sono presenti allo stato dell'arte in modo tale da decidere in autonomia la più adatta alle proprie esigenze.

Il modello Nerfacto si comporta molto bene nella resa di scene 3D permettendone l'esplorazione attraverso il posizionamento di camere virtuali all'interno della scena. E' molto veloce nell'elaborazione ma lascia un po' a desiderare nel comparto mesh. Questa soluzione è indicata per scene in cui il dettaglio non è necessario a livello di mesh ma è d'interesse l'esplorazione della scena.

Il modello Bakedangelo porta grandi innovazioni nel campo della ricostruzione di mesh ad alta fedeltà, a scapito però di un alto tempo di computazione. Molto indicato per scene at alto dettaglio in cui è importante avere una mesh di alto livello, ad esempio per un task di studio della degradazione di un monumento nel tempo. Non solo, le mesh estratte da questo modello si prestano anche all'utilizzo nell'ambito videogames: se si volesse ricostruire la città di New York con i suoi monumenti, basterebbe procurarsi un dataset contenente tali immagini per generare una mesh di alto livello che permetta un intervento minimo da parte del designer nella sua implementazione.

Gaussian Splatting è un nuovo approccio che si interpone tra i due precedentemente descritti, e si sta affermando come tecnologia rivale NeRF nell'ambito della velocità di computazione senza sacrificare troppo l'ambito mesh. Si prevede un grande sviluppo in questo ambito, specie considerato il fatto che l'estrazione di mesh da gaussian splatting è contemporanea alla data di stesura di questo lavoro, ed i risultati sono molto promettenti.

Infine, gli esperimenti svolti su CAVE sono stati fondamentali per capire quanto le mesh estratte fossero di alta fedeltà: riuscire a muoversi attorno alla mesh, avvicinarsi, ruotarla come se si fosse in assenza di gravità promette grandi utilizzi sia nell'ambito Cultural Heritage per l'analisi di monumenti, ma in tantissimi altri: automotive, aerospaziale, cinematografico, moda. Dare la possibilità ad un cliente di poter esplorare come meglio crede ed a fedeltà massima l'articolo che vuole comprare, e' grande incentivo per l'acquisto, permettendogli di togliersi molti dubbi.

5.2 Sviluppi Futuri

Interessante può essere l'approfondimento di altre reti NeRF, SDF e Gaussian Splatting, poichè ciascuna tecnologia presenta ha molteplici implementazioni, ciascuna con vari pregi e difetti. Inoltre, andare a studiare nello specifico se ci siano elementi che possano migliorare approcci già esistenti (e.g. Neuralangelo con Instant-NGP e Bakedangelo con Neuralangelo e BakedSDF) potrebbe portare a risultati ancora migliori di quanto non si abbiano già. Nuovi approcci promettenti come NeuS2[30] si prefissano di ridurre il tempo per la creazione di mesh ad alta definizione a pochi minuti, senza sacrificare livello di dettaglio. Tuttavia queste tecnologie sono ancora in fase di studio ed è necessario mantenersi aggiornati giorno per giorno, essendo questo campo di ricerca molto attivo.

Si è parlato di come il modello Nerfacto sia una combinazione di molteplici approcci: sebbene NeRF e Gaussian Splatting non siano tecnologie affini, sarebbe d'interesse cercare se ci fossero punti di incontro e margini di miglioramento tra di esse. In egual modo, uno studio sulla possibile implementazione di approcci SDF all'interno di metodologie NeRF potrebbe portare allo sviluppo di approcci che uniscono i vantaggi delle prime con quelli delle seconde, realizzando nel fotorealismo più totale una proiezione vera e propria degli spazi 3D del mondo reale all'interno del calcolatore.

Riguardo ai dataset utilizzati nel corso di questo studio, infine, sarebbe interessante verificare le prestazioni del modello al migliorare della qualità e quantità di pose ed angolazioni di scatto; si nota difatti come ad esempio per il dataset Magalotti, i modelli 3D estratti non siano di buona qualità proprio per l'assenza di esse. L'aggiunta di queste proposte auspica un netto miglioramento per le mancanze elencate in ciascuno dei modelli, portando sempre più dettagli e fedeltà nella ricostruzione, valorizzando questi approcci.

Ringraziamenti

I primi ringraziamenti vanno sicuramente al Professor Primo Zingaretti, che continuativamente ha prestato spunti di riflessione per il miglioramento continuo di questo lavoro di tesi, rendendolo qualcosa di cui posso andare fiero.

Insieme con lui ringrazio il Dott. Emanuele Balloni per la vicinanza, l'enorme pazienza e la tempestiva comunicazione, sempre a portata di messaggio per delucidazioni e confronti.

Un grazie anche a tutto il VRAI, dipartimento che ha reso possibile questo lavoro di tesi mettendo a disposizione tutta la strumentazione necessaria al suo svolgimento.

Ringrazio inoltre Melissa e David per essere stati compagni di questo lungo percorso sin dalla triennale e maggiormente nel percorso magistrale, rendendo i gruppi di progetto sempre una scelta vincente, potendo contare su di loro sia nello studio, che in amicizia.

Ringrazio soprattutto la mia famiglia: mamma, babbo, Mervin, Jessie, per il supporto continuo durante tutto il mio percorso universitario che si chiude qui, ma che non avrei mai potuto affrontare senza di loro.

Ringrazio tutti i miei amici per aver creduto in me anche nei momenti più difficili, che sia con un abbraccio, con un saluto, o un semplice messaggio.

E infine, grazie al mio fedele braccialetto portafortuna, che mi ha accompagnato ad ogni esame uscendone sempre vincente e fiero di tutto quello che ho raggiunto in questo mio percorso.

Fabriano, Dicembre 2023

David Ceka

Bibliografia

[1] David Caprari. Neural radiance fields:architettura e pipeline per il cultural heritage. 2023.

[2] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[3] Georgia Gkioxari, Peng Wang, and Jitendra Malik. Mesh r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019.

[4] Nanyi Wang, Yinda Zhang, Zerong Li, Yanwei Fu, and Wei Liu. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[5] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 21(4):163–169, 1987.

[6] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[7] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3D shapes. 2021.

[8] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *NeurIPS*, 2021.

[9] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant

neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[11] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, SIGGRAPH '23, 2023.

[12] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. NeRF−−: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064*, 2021.

[13] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR*, 2021.

[14] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-NeRF: Structured view-dependent appearance for neural radiance fields. *CVPR*, 2022.

[15] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022.

[16] Michael Niemeyer, Lars M. Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. *CoRR*, abs/1912.07372, 2019.

[17] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *Advances in Neural Information Processing Systems*, 33, 2020.

[18] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. *CoRR*, abs/2002.10099, 2020.

[19] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aa-

naes. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[20] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4), 2017.

[21] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023.

[22] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision (ECCV)*, 2022.

[23] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson Surface Reconstruction. In Alla Sheffer and Konrad Polthier, editors, *Symposium on Geometry Processing*. The Eurographics Association, 2006.

[24] Zehao Yu, Anpei Chen, Bozidar Antic, Songyou Peng, Apratim Bhattacharyya, Michael Niemeyer, Siyu Tang, Torsten Sattler, and Andreas Geiger. Sdfstudio: A unified framework for surface reconstruction, 2022.

[25] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[26] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[27] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[28] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering, 2023.

[29] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[30] Yiming Wang, Qin Han, Marc Habermann, Kostas Daniilidis, Christian Theobalt, and Lingjie Liu. Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.