Alma Mater Studiorum
Università di Bologna
Campus di Cesena

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA
Corso di Laurea in Ingegneria e Scienze Informatiche

Utilizzo di reti avversarie generative per il miglioramento di render di immagini 3D

Elaborato in:
Metodi numerici

Relatore:
Prof.
Alessandra Lumini

Presentata da:
Giovanni Prete

I Sessione
Anno Accademico 2023-2024

Alla mia mamma
che mi interrogava sui verbi irregolari in inglese,
questa laurea è anche sua

Indice

Introduzione **iii**

1 Il problema del rendering **1**

1.1 La funzione del rendering 1

1.2 La pipeline del rendering 2

1.2.1 Vertex Shading 2

1.2.2 Rasterization 2

1.2.3 Fragment Shading 3

1.3 Il Ray Tracing e l'illuminazione globale 3

1.4 Problemi del ray tracing 4

1.5 Perché ridurre i tempi di rendering 4

1.6 Obiettivo di questo lavoro 5

2 L'uso dell'intelligenza artificiale nel Rendering **7**

2.1 Panoramica sull'intelligenza artificiale 7

2.1.1 Le reti neurali 8

2.1.2 Deep learning e reti convoluzionali 8

2.1.3 Generative Adversarial Networks 10

2.2 Stato dell'arte 10

2.2.1 Riduzione del rumore con autoencoder 10

2.2.2 Riduzione del rumore con Kernel-Predicting Convolutional Networks 12

3 Dataset **13**
3.1 Infinigen e la generazione procedurale 13
3.1.1 La generazione procedurale 14
3.1.2 Blender e Cycles 14
3.2 Creazione del dataset 15
3.2.1 La divisione in tiles 15

4 Metodo proposto **19**
4.1 Dati di input 19
4.2 Elaborazione dell'input 20
4.2.1 pix2pix 20
4.2.2 Output della rete e ricostruzione delle immagini 22

5 Prove sperimentali **27**
5.1 Indicatori di Performance 27
5.1.1 Peak Signal-to-Noise Ratio (PSNR) 28
5.1.2 Structural Similarity Index Measure (SSIM) 28
5.2 Protocollo di Testing 29
5.2.1 Preparazione dei Dati 29
5.2.2 Procedura di Esecuzione dei Test 29
5.3 Risultati sperimentali 29
5.4 Analisi dei Risultati 32
5.4.1 Usabilità 32

Conclusioni **33**

Bibliografia **35**

Introduzione

Negli ultimi decenni, l'intelligenza artificiale ha rivoluzionato numerosi campi del sapere umano, dalla medicina alla finanza, dall'industria manifatturiera alla gestione delle risorse naturali. Tra le molteplici applicazioni dell'intelligenza artificiale, il campo del rendering delle immagini digitali ha visto un impatto particolarmente significativo. Il rendering, processo mediante il quale un'immagine 3D viene convertita in una rappresentazione bidimensionale, è fondamentale per molteplici settori, tra cui l'industria cinematografica, la progettazione architettonica, i videogiochi e la realtà virtuale.

La complessità intrinseca del rendering risiede nella necessità di calcolare in modo preciso l'interazione della luce con gli oggetti tridimensionali, un compito che richiede potenza computazionale elevata e tempi di calcolo prolungati. È proprio in questo contesto che l'intelligenza artificiale si pone come una soluzione innovativa, capace di ridurre drasticamente i tempi di rendering senza compromettere la qualità delle immagini.

In questa tesi, esploreremo l'applicazione delle reti neurali, in particolare delle Generative Adversarial Networks (GANs), nel miglioramento del processo di rendering. Le reti neurali hanno dimostrato un potenziale straordinario nel riconoscimento di pattern complessi e nella generazione di dati sintetici, rendendole strumenti ideali per affrontare le sfide del rendering.

Il lavoro è suddiviso in diversi capitoli: inizieremo con una panoramica sul problema del rendering e la sua importanza, proseguendo con un'analisi dettagliata delle pipeline di rendering tradizionali. Successivamente, esplore-

remo l'uso dell'intelligenza artificiale nel rendering, focalizzandoci sulle reti neurali e sulle tecniche di deep learning. Verranno presentati vari dataset utilizzati per l'addestramento e la validazione dei modelli, descrivendo in dettaglio il processo di creazione e suddivisione dei dati. Nel capitolo dedicato al metodo proposto, esamineremo i dati di input, l'elaborazione tramite la rete neurale pix2pix, e i risultati ottenuti. Infine, discuteremo i risultati sperimentali e le possibili applicazioni pratiche, concludendo con un'analisi dell'usabilità delle tecniche sviluppate.

Elenco delle figure

3.1 Immagine renderizzata con 100 spp e denoising. 15
3.2 Immagine renderizzata con 1 spp. 16

4.1 Architettura di una U-net. 21
4.2 Differenze tra immagine di input, ground truth e output del modello. 23
4.3 Immagine ricomposta senza elaborazione. 23
4.4 Immagine ricomposta facendo la media fra i pixel sovrapposti. 24
4.5 Immagine ricomposta con Poisson blending. 25
4.6 Immagine ricomposta facendo la media ponderata. 26
4.7 Immagine di input e immagine ricomposta con media ponderata. 26

5.1 Differenze strutturali (c) tra ground truth (a) e output del modello (b) con SSIM pari a 0.8156. 31

Capitolo 1

Il problema del rendering

1.1 La funzione del rendering

Nell'industria cinematografica e videoludica, l'animazione e gli effetti speciali sono diventati una parte centrale della creatività visiva, in grado di trasportare il pubblico in luoghi che sfidano l'immaginazione e la percezione umana. Tuttavia, dietro a ogni sequenza mozzafiato, si celano tecnologie complesse e un processo intricato che richiede l'applicazione di avanzati algoritmi di rendering. Quando parliamo di rendering in questo contesto, ci riferiamo a tutte quelle tecniche che si occupano di trasformare scene virtuali elaborate, ricche di complessi modelli 3D, effetti e illuminazione, in immagini coinvolgenti e realistiche. Sebbene questo processo al giorno d'oggi sia molto efficace e in grado di produrre immagini praticamente indistinguibili dalla realtà, è un procedimento estremamente lungo. Questo è particolarmente vero se consideriamo la produzione di film destinati al grande schermo e quindi ad alta risoluzione. Possono volerci anche più di 24 ore per la realizzazione di un singolo frame di scene con elementi complessi come corpi d'acqua.

1.2 La pipeline del rendering

Una volta stabilita una scena tridimensionale e una camera virtuale, possiamo definire la pipeline di rendering come la proiezione di tale scena in uno schermo bidimensionale attraverso una serie di trasformazioni geometriche e di sistemi di riferimento. In particolare, possiamo identificare tre momenti chiave all'interno di questo processo: Vertex Shading, Rasterization e Fragment Shading.

1.2.1 Vertex Shading

Il primo passo nella pipeline di rendering consiste nella trasposizione di ogni vertice che compone un modello dal sistema di coordinate proprio dell'oggetto a quello del mondo in cui si trova, ovvero il World Coordinate System. Successivamente, l'origine del sistema di riferimento viene spostata sulla camera virtuale, che non è altro che il punto di vista dell'osservatore. A questo punto viene definito il volume di vista, ovvero la regione della scena che viene effettivamente vista dalla camera e che va quindi renderizzata; tutto ciò che si trova al di fuori di questo volume verrà scartato attraverso il clipping, che consiste nel creare un nuovo set di primitive geometriche escludendo tutte quelle che si trovano fuori dal volume di vista. Infine, le coordinate vengono mappate da un sistema tridimensionale a quello bidimensionale dei pixel che compongono il viewport.

1.2.2 Rasterization

Con il termine rasterization intendiamo il processo che trasforma una primitiva geometrica collocata in uno spazio continuo nella sua versione discreta, più adatta a uno schermo con un numero limitato di pixel. In particolare, si tratta di algoritmi per la generazione di frammenti occupati da una primitiva geometrica; i pixel di questi frammenti condivideranno lo stesso colore nell'immagine finale.

1.2.3 Fragment Shading

Infine, ogni frammento di pixel rasterizzato ottiene un valore RGB che dipende in primis dal materiale del modello, dall'illuminazione e dalle ombre che si intersecano con l'oggetto. Una volta identificato il colore del materiale assegnato al poligono, questo viene modificato in luminosità a fronte della posizione e della rotazione del poligono stesso rispetto alle fonti di luci. L'angolo tra il vettore normale del poligono e il vettore che modella la fonte di luce modifica il colore finale della primitiva.

1.3 Il Ray Tracing e l'illuminazione globale

Il modello tradizionale di rendering con la pipeline vista sopra è detto locale in quanto le primitive vengono trattate indipendentemente dalle altre e il modo in cui un poligono viene visualizzato nell'immagine finale dipende esclusivamente dal esso e dalla luce che lo colpisce direttamente. Questo porta una serie di vantaggi a livello di performance, infatti il processo di per sé non è eccessivamente complesso poiché l'illuminazione è relativamente facile da gestire, il costo di elaborazione è costante per ogni primitiva e il rendering si può parallelizzare a livello di primitiva, rendendo ancora più agile il processo. Tuttavia, questo metodo produce immagini poco realistiche, soprattutto per quanto riguarda l'illuminazione e i riflessi che, nel rendering locale, non influiscono sull'illuminazione degli oggetti circostanti. Per migliorare la qualità e il realismo dei render, vengono introdotti degli algoritmi in grado di modellare aspetti quali la riflessione diffusiva e speculare, insieme a una corretta gestione delle ombre; fra questi, uno dei più utilizzati è il ray tracing.
Il ray tracing è un algoritmo view-dependent che sfrutta l'eliminazione delle superfici nascoste in combinazione con lo shading per gestire ombre, rifrazione e riflessione. L'idea alla base di questo metodo è che di tutti i raggi luminosi che vengono generati, solo quelli che raggiungono il punto di vista dell'osservatore, direttamente o per effetto delle interazioni con altre superfi-

ci, contribuiscono all'immagine finale. Poiché calcolare la traiettoria di ogni raggio emesso è impossibile, si simula il percorso inverso dei raggi a partire dall'osservatore fino ad arrivare alla sorgente di luce e, per far sì che questo algoritmo produca risultati soddisfacenti, per ogni pixel vengono calcolati più raggi, detti samples. Maggiore è il numero di campioni (samples) per pixel, minore sarà il rumore presente nell'immagine e maggiore sarà il suo realismo. A causa della complessità delle interazioni luminose, ogni singolo campione può dare un risultato molto variabile, specialmente in aree con ombre, riflessioni complesse o illuminazione indiretta. Con un numero limitato di campioni per pixel, il risultato finale tende a essere rumoroso, perché ogni campione contribuisce in modo significativamente diverso alla colorazione finale del pixel.

1.4 Problemi del ray tracing

Sebbene il ray tracing possa garantire un ottimo grado di realismo, è un processo assai complesso e dispendioso a livello di calcolo. Infatti, per ottenere immagini accettabili a livello qualitativo, è necessario un numero considerevolmente alto di samples per pixel, così da ridurre al minimo il rumore complessivo dell'immagine. Considerando che anche le immagini più semplici con una semplice illuminazione conducono a tempi di rendering importanti, per le grandi produzioni cinematografiche, soprattutto quelle che coinvolgono elaborati effetti speciali tipici dei film d'azione, i tempi di produzione aumentano esponenzialmente. Per i colossal movie di Hollywood, può volerci anche un anno intero per la renderizzazione completa di un film.

1.5 Perché ridurre i tempi di rendering

Sebbene i moderni software per la creazione di video con effetti speciali e modelli 3D offrano delle anteprime accurate, gli effetti della luce su superfici riflettenti, con trasparenze o con particolari textures non possono essere

riprodotti con fedeltà in una preview. Questo vuol dire che anche per sequenze brevissime non si ha la concezione di come sarà effettivamente il risultato finché questo non viene renderizzato e, laddove il risultato non dovesse essere soddisfacente, si dovrebbe ripetere il processo a fronte delle modifiche correttive. Per questo motivo, ridurre le tempistiche della pipeline di rendering aiuterebbe le grandi produzioni ad avere una visione più chiara del prodotto ancora prima di partire con un render accurato e pronto per la distribuzione.

1.6 Obiettivo di questo lavoro

L'obiettivo principale di questa tesi è esplorare e implementare l'uso delle reti avversarie generative (GANs) per migliorare il processo di rendering delle immagini 3D. Il rendering, fondamentale per settori come il cinema, l'architettura e i videogiochi, è un processo che richiede un'alta potenza computazionale per calcolare l'interazione della luce con gli oggetti tridimensionali. Questo lavoro si propone di ridurre significativamente i tempi di rendering mantenendo alta la qualità delle immagini finali. Attraverso l'utilizzo delle GANs, si mira a ottimizzare l'efficienza del rendering, trasformando il modo in cui vengono create le immagini digitali e fornendo una valida alternativa ai metodi tradizionali, con potenziali applicazioni pratiche nelle fasi di sviluppo e pre-visualizzazione.

Capitolo 2

L'uso dell'intelligenza artificiale nel Rendering

Nel campo del rendering, la qualità visiva e i tempi di calcolo rappresentano sfide significative. L'utilizzo di reti neurali in questo campo può produrre risultati molto soddisfacenti in tempi significativamente minori rispetto a quelli del rendering tradizionale. In questo capitolo vengono esplorate alcune delle problematiche più rilevanti incontrate nel rendering avanzato e le soluzioni proposte dalla letteratura. Successivamente, viene introdotto il concetto di reti neurali e deep learning, descrivendo il loro funzionamento e le applicazioni specifiche nel miglioramento del rendering delle immagini 3D.

2.1 Panoramica sull'intelligenza artificiale

Negli ultimi decenni, l'intelligenza artificiale (IA) ha subito un'incredibile evoluzione che ha permesso il suo inserimento in numerosi settori, come la medicina, i processi industriali e, più di recente, la creazione di contenuti. Di seguito, si esplorano i fondamenti delle reti neurali e delle reti convoluzionali, evidenziando il loro impatto significativo sul miglioramento della qualità dei render.

2.1.1 Le reti neurali

Le Artificial Neural Network (ANN) [1] sono una famiglia di algoritmi che fanno parte del più ampio settore del machine learning, il cui scopo è quello di prendere decisioni in maniera simile a come funziona la mente umana, imitandone il funzionamento. Si tratta di una struttura a più livelli, dove ogni livello è formato da nodi chiamati neuroni artificiali. I nodi sono connessi tra di loro con pesi e soglie di attivazioni specifiche: se un nodo riceve in input un valore sufficientemente alto esso viene attivato e trasmette l'input al nodo successivo, fino ad ottenere uno o più valori di output finale.

Backpropagation Quello che rende le reti neurali così efficaci è la famiglia di algoritmi di backpropagation. Questi algoritmi si occupano di modificare i pesi che collegano i vari nodi secondo un learning rate specifico, che corrisponde alla velocità con cui vengono modificati i pesi della rete. Questo permette alle ANN di apprendere dai propri errori, migliorando la loro accuratezza.

2.1.2 Deep learning e reti convoluzionali

Il numero di livelli di una rete neurale è molto variabile e dipende soprattutto dal tipo di lavoro che si vuole affidare al modello. Se l'architettura comprende più di tre livelli si parla di rete neurale profonda e quindi di deep learning [2]. Questo tipo di ANN viene addestrato su grandi dataset poiché il numero di pesi da modificare aumenta esponenzialmente con il numero di livelli e, inoltre, avere un numero limitato di dati può portare all'overfitting, ovvero quel fenomeno che fa sì che il modello fornisca previsioni accurate sui dati di training ma non su dati nuovi. A differenza delle reti neurali più piccole, le reti profonde sono in grado di classificare fenomeni e riconoscere pattern nascosti con maggiore accuratezza.

Tipologie di addestramento La versatilità di questi modelli li rende capaci di apprendere in diversi modi, in particolare i tre metodi di appren-

dimento più utilizzati sono quello supervisionato, quello non supervisionato e l'allenamento con rinforzo. Con l'allenamento supervisionato si intende un addestramento basato su un "labeled dataset" che quindi presuppone un lavoro umano volto alla categorizzazione dei dati in input, mentre in quello non supervisionato è il modello che deve trovare in autonomia i pattern nei dati e effettuare clusterizzazione. Nel reinforcement learning invece, il modello viene messo in un ambiente dove deve eseguire una determinata azione e impara attraverso un sistema di feedback dove riceve un premio o una penalità in base alle sue performance.

Convolutional Neural Networks Tra le architetture che rientrano nella classe del deep learning troviamo le Convolutional Neural Networks (CNN), delle reti che utilizzano dati per la classificazione di immagini e il riconoscimento di pattern [3]. A differenza delle ANN classiche, queste sono formate da livelli di diverso tipo: convolutional layers, pooling layers e fully-connected layers. I convolutional layers, che danno il nome alla rete, sono la parte principale e si occupano della "feature extraction", ovvero l'individuazione di tratti definiti a priori che riducono l'input alle sue caratteristiche principali. Per fare ciò si utilizzano filtri o kernel (matrici 3x3) che vengono fatti scorrere sull'input e che restituiscono una "feature map", che conterrà tutte le proprietà più rilevanti dei dati di input.
I livelli di pooling servono a ridurre le dimensioni dell'input in modo da ridurne la complessità e aumentarne l'efficienza, a costo di perdere informazioni. Il loro funzionamento è simile a quello dei livelli convoluzionali e i metodi più utilizzati sono "max pooling" e "average pooling". Sebbene possano essere presenti più livelli di pooling e convoluzionali all'interno della stessa rete, lo stesso non si può dire per i quelli fully connected, che si trovano solamente alla fine della catena. Ogni nodo di questi livelli è connesso ai nodi dei livelli precedenti e sono i quelli che producono effettivamente l'output della rete, ovvero eseguono la classificazione.

2.1.3 Generative Adversarial Networks

Nell'ultimo decennio è emerso un nuovo e rivoluzionario tipo di intelligenza artificiale. Se prima lo scopo principale delle AI era quello di classificare e raggruppare dati o fare predizioni ora si pone molta attenzione sulla possibilità di generare nuovi dati, sotto forma di immagini, audio o testi. Una delle architetture che ha dato il via a quest'evoluzione è sicuramente quella delle Generative Adversarial Networks (GAN)[4]. Con GAN ci riferiamo a una AI basata su due reti neurali concorrenti, una con il ruolo di generatore e l'altra che ha la funzione di discriminatore, la prima riceve in input del rumore casuale e genera un output che verrà poi passato alla seconda rete insieme a dei dati appartenenti a un dataset reale. Lo scopo del discriminatore è quello di identificare quali dati sono reali e quali sono generati, mentre il generatore deve "ingannare" il discriminatore facendo passare il proprio output come reale.
La forza di questa architettura sta proprio nella concorrenza delle due reti, poiché più una delle due reti è ben allenata, più l'altra sarà costretta a migliorare.

2.2 Stato dell'arte

2.2.1 Riduzione del rumore con autoencoder

Poiché il ray tracing con metodi di Monte Carlo produce molto rumore, specialmente con un numero di sample per pixel basso, sono stati studiati molti metodi per la ricostruzione di render che sfruttano l'intelligenza artificiale e le reti neurali. Uno studio rilevante in questo ambito è quello dei reparti di ricerca di Nvidia intitolato "Interactive Reconstruction of Monte Carlo Image Sequences using a Recurrent Denoising Autoencoder" [5], che propone l'utilizzo di un autoencoder ricorrente per il denoising di immagini generate con Monte Carlo. Lo scopo dello studio era quello di partire da render generati con un solo sample per pixel per arrivare a un'immagi-

ne qualitativamente accettabile utilizzando una rete neurale con una fase di codifica e decodifica che operano su risoluzioni rispettivamente decrescenti e crescenti.

Autoencoders e denoising autoencoders Gli autoencoder sono un tipo di rete neurale che comprime i dati di input in una forma più ridotta prima di passarli al decoder [6]. In questo modo la rete è costretta a conservare solamente quelle informazioni che possono risultare nascoste, ma che sono utili alla ricostruzione accurata dell'immagine. Per questo i livelli dell'encoder sono progressivamente più piccoli, fino a raggiungere il numero minimo di features rilevanti per la decodifica.
I denoising autoencoders lavorano allo stesso modo ma pongono un'enfasi particolare nella rimozione del rumore dall'immagine di input. L'architettura proposta da Chaitanya et al. è un autoencoder con skip connection, ovvero delle connessioni tra layer corrispondenti di encoder ed encoder, e blocchi ricorrenti, che consistono in strati convoluzionali connessi a layer nascosti precedenti. Le skip connection servono per non perdere traccia di features ad alta risoluzione che chiaramente andrebbero perse in un decoder normale, mentre i blocchi convoluzionali servono per avere una maggiore stabilità temporale, poiché questa rete era stata pensata soprattutto per il denoising di interi filmati.

I risultati presentati nello studio dimostrano un significativo miglioramento nella qualità delle immagini e nella stabilità delle sequenze temporali rispetto alle tecniche esistenti. L'autoencoder ricorrente proposto permette di ottenere immagini più realistiche con un rumore notevolmente ridotto, rendendolo una soluzione promettente per il denoising nel rendering Monte Carlo.

2.2.2 Riduzione del rumore con Kernel-Predicting Convolutional Networks

Un contributo rilevante in questo ambito è il lavoro di Bako et al. [7] intitolato "Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings", che propone l'uso di reti neurali convoluzionali per il denoising delle immagini generate con Monte Carlo. L'obiettivo dello studio era migliorare la qualità delle immagini rumorose mediante una rete che prevede i pesi dei kernel di denoising applicati ai pixel, consentendo di ottenere risultati di alta qualità con tempi di calcolo ridotti.

Kernel-Predicting Convolutional Networks (KPCN) Le reti neurali convoluzionali sono potenti strumenti per l'elaborazione delle immagini, capaci di apprendere rappresentazioni complesse dai dati di input. La rete KPCN proposta da Bako et al. prevede i kernel di pesi locali per ogni pixel rumoroso, che vengono poi utilizzati per calcolare il valore denoised del pixel a partire dai suoi vicini. Questa architettura si basa su una decomposizione del problema in componenti diffuse e speculari, trattate separatamente per preservare i dettagli importanti dell'immagine. La rete è costituita da più strati convoluzionali, dove ogni strato esegue convoluzioni e applica funzioni di attivazione non lineari come le Rectified Linear Units (ReLU), eccetto l'ultimo strato che utilizza la funzione identità per produrre i kernel finali.

I risultati dello studio dimostrano un miglioramento significativo nella qualità delle immagini rispetto ai denoisers Monte Carlo esistenti. La rete KPCN addestrata su dati di produzione reali, come fotogrammi da "Finding Dory" e "Cars 3", ha mostrato una buona generalizzazione su diverse scene, producendo immagini di alta qualità con un rumore ridotto. Questo rende l'approccio KPCN una soluzione promettente per il denoising nel rendering Monte Carlo.

Capitolo 3

Dataset

Nell'ambito dell'intelligenza artificiale, i dataset rappresentano la pietra angolare su cui si fonda l'addestramento e la validazione dei modelli. Un dataset ben progettato, ampio e diversificato è essenziale per garantire che i modelli di intelligenza artificiale possano generalizzare efficacemente e gestire situazioni del mondo reale. In particolare, nella rimozione del rumore generato da tecniche di rendering come il ray tracing, la qualità dei dati di addestramento può determinare il successo o il fallimento di un approccio basato su reti neurali. Questo capitolo esplora il ruolo critico dei dataset, le loro caratteristiche fondamentali e i metodi utilizzati per generare il dataset di riferimento per lo sviluppo del progetto.

3.1 Infinigen e la generazione procedurale

Per la generazione del dataset è stato utilizzato Infinigen [8], un avanzato sistema di generazione procedurale progettato per creare una vasta gamma di dati sintetici 3D utilizzati principalmente nell'addestramento di modelli di apprendimento automatico. Sfruttando il software open-source Blender e un motore di rendering basato su path tracing chiamato Cycles, Infinigen può generare realistiche scene 3D e oggetti complessi. La sua innovazione principale risiede nell'automatizzazione della conversione dei grafici a nodi

in codice Python tramite un Node Transpiler, rendendo i processi di generazione e randomizzazione più flessibili ed espressivi. Inoltre, Infinigen include tecniche avanzate per l'estrazione della ground truth, superando le limitazioni dei metodi tradizionali e fornendo dati altamente accurati per il training di reti neurali. Questo sistema permette di creare scenari vari e dettagliati, come paesaggi naturali e animazioni di creature, dimostrando una notevole capacità di generalizzazione per modelli addestrati su dati sintetici.

3.1.1 La generazione procedurale

Con generazione procedurale facciamo riferimento alla capacità di un sistema di creare, in base a dei simulatori e a regole prestabilite, dati che possono essere sotto forma di testi, musica, immagini o ambienti tridimensionali, come ad esempio nei videogiochi. Infinigen è un motore totalmente procedurale perché non utilizza un numero finito di asset 3D ma è in grado di generare un numero infinito di forme, materiali e texture, che vengono applicati in scene altrettanto casuali e uniche.

3.1.2 Blender e Cycles

Infinigen si basa su Blender [9], un software open-source che mette a disposizione una suite completa per la creazione 3D e che è facilmente integrabile con script Python [10]. Sfruttando i node graphs di Blender, un sistema in grado di creare forme e texture procedurali attraverso un'interfaccia visuale, è possibile creare regole matematiche per la generazione di strutture uniche.

Per il rendering delle scene viene utilizzato Cycles [11], un motore di rendering unbiased integrato in Blender. Cycles, essendo un motore unbiased (che quindi non si basa su approssimazioni), si basa sul ray tracing e sul path tracing, simulando con precisione fisica il comportamento della luce e la sua interazione con gli oggetti nella scena 3D. Questo approccio consente di ottenere immagini fotorealistiche con effetti di luce complessi, riflessioni, rifrazioni e caustiche, il tutto con un elevato grado di realismo. Allo stesso

tempo, questa sua caratteristica lo rende un motore molto pesante, che può richiedere molto tempo per elaborare scene complesse e con effetti di riflessioni e trasparenza, tipiche delle scene con specchi d'acqua. Cycles integra Intel Open Image Denoiser [12], una libreria open-source basata su modelli di deep learning specializzati nel denoising di immagini. Questa libreria viene utilizzata al termine del rendering per ridurre eventuali artefatti prodotti dal ray tracing.

3.2 Creazione del dataset

Il dataset è formato da 443 immagini 1920x1080 pixel create da Infinigen, che sono divise equamente fra i 13 biomi standard. Di ogni immagine ci sono due versioni renderizzate con Cycles: una con 100 sample per pixel e su cui è stato applicato il denoising (Fig.3.1) e una con un sample per pixel senza denoising (Fig.3.2).

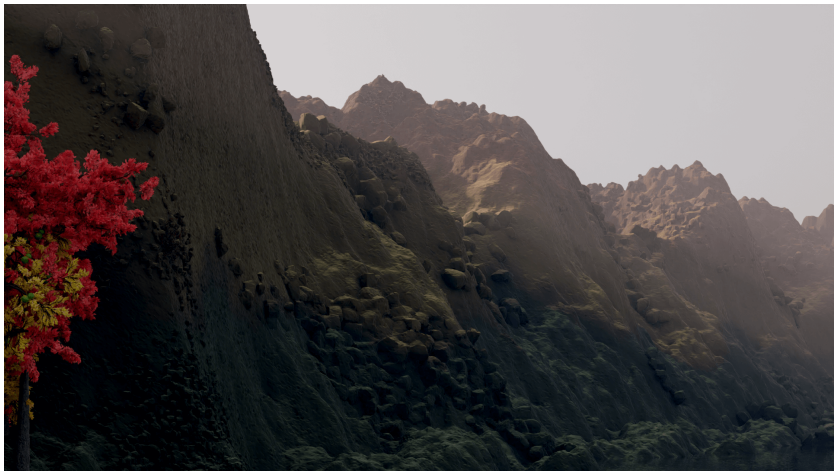


Figura 3.1: Immagine renderizzata con 100 spp e denoising.

3.2.1 La divisione in tiles

Un'immagine 1920x1080 pixel non è la scelta giusta per un qualsiasi modello, perché richiederebbe a una rete neurale con un livello di input molto

Figura 3.2: Immagine renderizzata con 1 spp.

grande e diversi milioni di parametri da ottimizzare e quindi anche a un enorme consumo in termini di memoria, oltre che a tempi di addestramento estremamente lunghi. Per questo motivo è necessario dividere le immagini in tiles più piccole, anche per evitare che un downsampling eccessivo porti a una perdita importante di informazioni in fase di upsampling. Le dimensioni delle tiles incideranno sia sui tempi, sia sulle performance del modello. Inoltre, per utilizzare il modello su immagini di grandi dimensioni, queste tiles andranno ricomposte per formare l'output finale.

Prime versioni del dataset Inizialmente, le tiles avevano dimensioni di 128x128 pixel, con una sovrapposizione fra tiles adiacenti di 64 pixel. Ogni immagine quindi produceva 480 elementi e un dataset complessivo di 212.640 elementi. Come vedremo successivamente, questo primo dataset produceva risultati qualitativamente inaccettabili e in tempi molto lunghi. Una seconda versione del dataset comprendeva tiles da 256x256 con un overlapping di 64 pixel. Anche in questo caso, i tempi per il completamento di un'immagine non portavano alcun vantaggio.

Dataset finale Il dataset finale consiste di 2480 tiles da 512x512 pixel con overlapping di 50 pixel. Il dataset è diviso in training set e validation

set. Il training set comprende il 70% delle tiles totali ed è quello che il modello utilizza effettivamente per modificare i suoi parametri, mentre il validation set, che comprende il restante 30% delle tiles, viene utilizzato per la validazione del modello in fase di training.

Per quanto riguarda il test set, ovvero quei dati che il modello non ha mai visto in fase di training per simulare dati reali, sono stati utilizzate 100 immagini mai viste dal modello e 100 frame di un video generato da Infinigen con un sample e senza denoising, che poi sono stati ricomposti per valutare l'impatto visivo in filmati.

Capitolo 4

Metodo proposto

Nel campo del rendering 3D, la continua ricerca di soluzioni che migliorino l'efficienza e la qualità visiva ha portato all'esplorazione di numerosi metodi innovativi. Questo capitolo descrive un nuovo approccio metodologico che combina l'uso delle reti neurali con tecniche avanzate di elaborazione delle immagini per ottenere render di alta qualità. Il metodo proposto utilizza una combinazione di segmentazione dell'immagine in tile e fusione dei tile, ottimizzando così sia la velocità di calcolo che la qualità del risultato finale. Inizieremo descrivendo i dati di input e i passaggi eseguiti, per poi analizzare l'architettura delle reti neurali utilizzate, i parametri di training e i diversi metodi di fusione dei tile sperimentati.

4.1 Dati di input

I dati che vengono forniti al modello sono immagini RGB in formato PNG, con risoluzione di 1920x1080 pixel, di scene generate artificialmente e proceduralmente con Infinigen. Queste immagini sono renderizzate utilizzando il ray tracing, con un solo sample per pixel e senza utilizzare alcun metodo di denoising. Per ridurre il numero di parametri della rete neurale e di conseguenza i tempi di addestramento, ogni immagine è stata suddivisa in 8 tile da 512x512 pixel, ognuna delle quali si sovrappone parzialmente alle

tile adiacenti, in modo da avere della ridondanza nei dati che servirà in fase di ricostruzione dei tile per garantire una transizione fluida fra tile vicine.

4.2 Elaborazione dell'input

I dati di input vengono elaborati da un modello di rete neurale, basato su pix2pix [13], precedentemente addestrato su un dataset di immagini di Infinigen formato da tile corrispondenti di immagini parzialmente renderizzate e complete. L'addestramento dura 200 epoche con un learning rate di 0.0002 e con batch di un solo elemento.

4.2.1 pix2pix

Pix2pix è un modello di Conditional Adversarial Networks (CAN) che si pone come soluzione generica a diversi problemi di image-to-image translation, ovvero tutti quei problemi che riguardano la trasformazione di immagini mantenendo delle caratteristiche dell'immagine originale. Le reti neurali che si occupano di questi problemi imparano la mappatura fra due domini di immagini.

Architettura

Il generatore G di pix2pix è una U-net [14], un'architettura basata su encoder-decoder e skip connections (Figura 4.1). Questa struttura permette di conservare in output tutte le caratteristiche fondamentali dell'immagine di input, che passeranno il bottleneck dopo la fase del downsampling, e manterrà il livello di dettaglio grazie alle skip connections, che connettono ogni livello i dell'encoder con ogni livello $n - i$ del decoder, dove n è il numero di layer da cui sono composte le due sezioni.

Il discriminatore D invece è una CNN che, a differenza di un discriminatore classico che prende in analisi l'intero output di G, classifica delle porzioni di dimensioni ridotte e per ognuna stabilisce se si tratta di un'immagine reale o generata. La rete viene chiamata PatchGAN e fornisce un risultato medio

fra tutte le patch che analizza. L'analisi di porzioni ridotte, di 70x70 pixel, rende il processo molto rapido perché la rete ha pochi parametri e non viene comunque penalizzata dal fatto che l'elaborazione avviene molte volte per un singolo output.

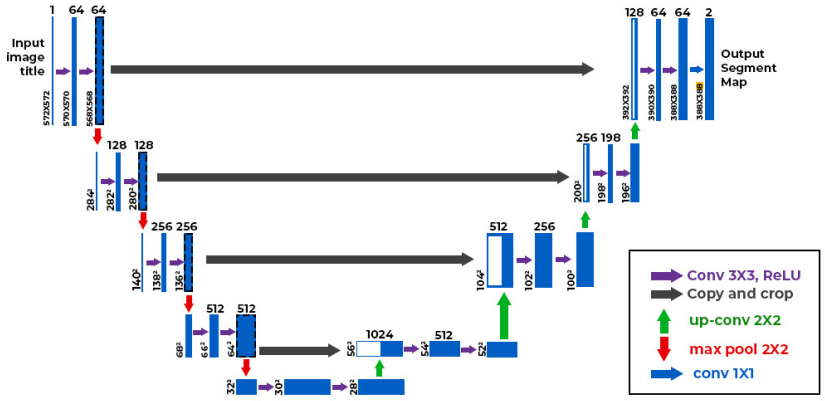


Figura 4.1: Architettura di una U-net.

Funzioni di loss

Una loss function (o funzione di perdita) nel contesto del machine learning è una funzione che misura quanto bene o male un modello predice l'output desiderato rispetto ai valori effettivi. Essa quantifica la differenza tra le previsioni del modello e i dati reali, fornendo un valore numerico che il modello cerca di minimizzare durante il processo di addestramento.

In pix2pix sono utilizzate più funzioni di loss. Una delle funzioni in gioco è la loss avversaria condizionale (cGAN loss) che è definita come:

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z)))] \qquad (4.1)$$

Nella formula, $D(x, y)$ rappresenta la probabilità che il discriminatore classifichi la coppia di immagini x e y, dove x è l'immagine di input e y è il ground truth, come reale. $D(x, G(x, z))$ è la probabilità che il discriminatore identifichi la coppia di immagini x e $G(x, z)$, dove $G(x, z)$ è l'immagine

generata a partire dall'input x e il rumore z, come reale. Questa funzione di loss viene definita avversaria perché il discriminatore cerca di massimizzare il risultato della loss, mentre il generatore lo minimizza.

Per fare in modo che il generatore non si limiti a "ingannare" il discriminatore, ma che produca anche risultati vicini al ground truth, la cGAN loss viene combinata con una L1 loss, che misura la distanza media assoluta tra l'immagine generata e l'immagine reale ed è definita come:

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}\left[\|y - G(x,z)\|_1\right] \tag{4.2}$$

L'obiettivo del generatore è quindi quello di minimizzare la somma ponderata delle due funzioni di loss:

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G,D) + \lambda \mathcal{L}_{L1}(G) \tag{4.3}$$

Dove λ è un iperparametro che bilancia l'importanza delle due loss.

Poiché il discriminatore D ha un ritmo di training più veloce rispetto a quello del generatore, la sua loss function viene dimezzata in modo che il suo addestramento sia rallentato.

$$D\_loss = 0.5 \cdot D\_loss \tag{4.4}$$

4.2.2 Output della rete e ricostruzione delle immagini

Il modello addestrato produce tile idealmente senza rumore della stessa grandezza dei tile in input (Figura 4.2). Per produrre l'immagine completa è necessario ricomporre i tile sfruttando l'overlapping.

Primi metodi

Inizialmente i tile venivano semplicemente ricomposti senza ulteriore elaborazione (Figura 4.3). Questo metodo ha evidenziato la mancanza di contesto fra frammenti della stessa immagine, che risultano infatti sconnessi e addirittura appartenenti a foto differenti.

Figura 4.2: Differenze tra immagine di input, ground truth e output del modello.

Figura 4.3: Immagine ricomposta senza elaborazione.

Un primo tentativo per risolvere questo problema è stato quello di fare una media fra i pixel sovrapposti per provare a rendere la transizione fra i frammenti più leggera (Figura 4.4). Tuttavia, questo ha creato ancora più artefatti visivi.

Metodi di blending

Una soluzione al problema è stata quella di utilizzare il blending di Poisson [15], un metodo per importare in modo fluido regioni di immagini opache e trasparenti in una destinazione e modificare l'aspetto di un'immagine all'interno di una regione mascherata specificata dall'utente, fondendola perfettamente con l'immagine di destinazione.

Questo approccio utilizza l'equazione di Poisson, un'equazione differen-

Figura 4.4: Immagine ricomposta facendo la media fra i pixel sovrapposti.

ziale parziale che, nel contesto dell'editing delle immagini, viene utilizzata per creare una transizione graduale e senza cuciture tra una regione dell'immagine che viene modificata e il resto dell'immagine.

Il metodo di Poisson Image Editing si articola in diverse fasi chiave. Si inizia selezionando manualmente una regione dell'immagine in cui si desidera effettuare l'editing. Successivamente, viene definito un campo vettoriale all'interno di questa regione: i gradienti di un'immagine sorgente vengono utilizzati se si sta eseguendo una clonazione, oppure il campo può essere scelto per ottenere specifici effetti di illuminazione o texture. L'equazione di Poisson $\Delta f = \operatorname{div} v$ viene quindi risolta numericamente con le condizioni al contorno appropriate, calcolando f che rappresenta l'immagine modificata all'interno della regione selezionata. Questa operazione viene eseguita indipendentemente per ciascun canale di colore (ad esempio, RGB). Infine, i valori dei pixel nella regione selezionata vengono sostituiti con quelli calcolati, garantendo una transizione graduale e senza soluzione di continuità con l'immagine circostante. Questo approccio permette di realizzare modifiche naturali e senza cuciture visibili, rendendo il metodo particolarmente utile per operazioni come la clonazione, la modifica del colore e della texture, e la creazione di tessere.

Le immagini ricostruite con questo metodo (Figura 4.5) non presentano evidenti artefatti grafici e sono visivamente molto vicine al ground truth.

Sebbene questo metodo produca ottimi risultati, risolvere l'equazione

Figura 4.5: Immagine ricomposta con Poisson blending.

di Poisson richiede diverse decine di secondi e ridurrebbe notevolmente il guadagno in termini di tempo dell'utilizzo della rete neurale.

Media ponderata dei pixel

Il metodo che è stato effettivamente utilizzato consiste nel fare una media ponderata dei pixel sovrapposti, ovvero assegnando a ciascun pixel un peso inversamente proporzionale alla su distanza dall'immagine di origine.

$$O(x,y) = I(x,y) \cdot \frac{tile\_size - x}{overlapping} + J(x,y) \cdot \left(1 - \frac{tile\_size - x}{overlapping}\right) \tag{4.5}$$

Nella forumula O è l'immagine finale, I e J sono le tiles che si sovrappongono, $tile\_size$ è la dimensione di un lato del frammento e $overlapping$ è il numero di pixel che si sovrappongono.

Questo processo produce risultati qualitativamente molto simili a quello che utilizza l'equazione di Poisson ma richiede una frazione di secondo per la ricostruzione di un'intera immagine.

Figura 4.6: Immagine ricomposta facendo la media ponderata.

Figura 4.7: Immagine di input e immagine ricomposta con media ponderata.

Capitolo 5

Prove sperimentali

Questo capitolo si concentra sulla descrizione dettagliata delle prove sperimentali condotte per valutare le performance delle diverse reti neurali utilizzate nel contesto del denoising e miglioramento dei render Monte Carlo. Vengono presentati gli indicatori di performance e vien descritto il protocollo di testing. Successivamente, si discutono i risultati ottenuti con vari approcci sperimentati e si confrontano con quelli presenti nella letteratura esistente. Infine, si fornisce una valutazione complessiva dei risultati, includendo un'analisi degli errori e un'esplorazione dell'usabilità delle soluzioni proposte.

5.1 Indicatori di Performance

Per valutare la qualità delle immagini generate e denoised dalle reti neurali proposte, sono stati utilizzati due principali indicatori di performance: il Peak Signal-to-Noise Ratio (PSNR) e lo Structural Similarity Index Measure (SSIM). Questi indicatori sono stati scelti per la loro capacità di fornire una valutazione quantitativa e qualitativa della qualità delle immagini, rispettivamente.

5.1.1 Peak Signal-to-Noise Ratio (PSNR)

Il Peak Signal-to-Noise Ratio (PSNR) è una misura quantitativa che viene comunemente utilizzata per valutare la qualità delle immagini e dei video compressi. PSNR è definito come il rapporto tra il valore massimo possibile di un segnale e la potenza del rumore che influenza la fedeltà della sua rappresentazione. È espresso in decibel (dB).

Matematicamente, PSNR è definito come:

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

Dove $\text{MAX}\{I\}$ è il valore massimo possibile del segnale, che corrisponde a 255 per un'immagine a 8 bit. Per immagini a colori tale definizione vale per una componente. Data un'immagine originale I e la versione con rumore K entrambe di dimensioni $M \times N$, MSE invece è l'errore quadratico medio tra le due immagini ed è definito come:

$$\text{MSE} = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \|I(x,y) - K(x,y)\|^2 \tag{5.2}$$

5.1.2 Structural Similarity Index Measure (SSIM)

Lo Structural Similarity Index Measure (SSIM) è un indicatore di qualità dell'immagine che cerca di migliorare le limitazioni degli indicatori basati su errori assoluti come il PSNR. SSIM misura la somiglianza tra due immagini in termini di luminanza, contrasto e struttura, riflettendo meglio la percezione umana della qualità visiva. SSIM è definito come:

$$\text{SSIM}(I,K) = \frac{(2\mu_I\mu_K + C_1)(2\sigma_{IK} + C_2)}{(\mu_I^2 + \mu_K^2 + C_1)(\sigma_I^2 + \sigma_K^2 + C_2)} \tag{5.3}$$

Nel calcolo del SSIM, μ_x e μ_y sono le medie locali delle intensità dei pixel delle due immagini, riflettendo la loro luminosità. Le varianze locali, σ_x^2 e σ_y^2, misurano il contrasto all'interno delle immagini, mentre σ_{xy} è la covarianza che indica quanto le deviazioni di intensità tra le due immagini siano

correlate. Le costanti C_1 e C_2 sono piccole quantità aggiunte per stabilizzare i calcoli, basate sulla scala di intensità dei pixel. La formula SSIM combina questi termini per confrontare luminosità, contrasto e struttura delle immagini.

5.2 Protocollo di Testing

Sono stati testati diversi modelli addestrati su dataset di dimensioni diverse e con tiles di grandezza variabile tra i 256 e i 512 pixel per lato.

5.2.1 Preparazione dei Dati

Per testare l'efficacia del modello addestrato è stato utilizzato un set di immagini generate da Infinigen con un solo sample per pixel e senza denoising. Queste immagini, che non sono mai state viste dalla rete neurale in fase di training, sono state precedentemente divise in tiles quadrate la cui dimensione dipende dalla grandezza delle tiles con cui è stato addestrato il modello.

5.2.2 Procedura di Esecuzione dei Test

Per l'esecuzione dei test, le tiles create a partire dalle immagini del test set vengono passate in input al modello preso in esame, che produce altrettante tiles elaborate. Queste tiles vengono inizialmente analizzate singolarmente utilizzando le misurazioni dei performance indicator e, successivamente, vengono ricomposte in modo da poter essere nuovamente studiate come immagini complete.

5.3 Risultati sperimentali

Le prove sperimentali sono state eseguite su quattro diversi addestramenti di pix2pix, denominati whole_ds_2, whole_ds_3, mini_whole_ds e desert_pix così strutturati:

- **whole_ds_2** è allenato per 200 epoche sul dataset completo, diviso in tiles da 512x512 pixel con un overlapping fra tiles vicine di 50 pixel.
- **whole_ds_3** è allenato per 200 epoche su un dataset privo di immagini in input che risultavano difficili da elaborare perché quasi completamente formate da rumore, diviso in tiles da 512x512 pixel con un overlapping fra tiles vicine di 50 pixel.
- **mini_whole_ds** è allenato per 200 epoche su un dataset completo ma diviso in tiles da 256x256 pixel con un overlapping fra tiles vicine di 64 pixel.
- **desert_pix** è allenato per 200 epoche ma su un dataset ristretto composto unicamente da immagini del bioma "deserto" di infinigen, diviso in tiles da 512x512 pixel con un overlapping fra tiles vicine di 50 pixel.

Parametri di addestramento			
Nome del modello	N° epoche	grandezza tile	overlapping
whole_ds_2	200	512x512	50
whole_ds_3	200	512x512	50
mini_whole_ds	200	256x256	64
desert_pix	200	512x512	50

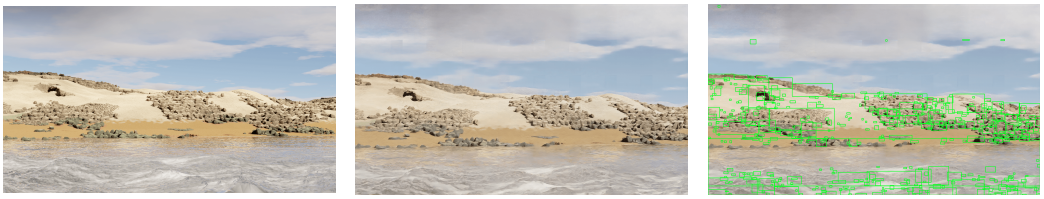
Da questi modelli è possibile estrapolare diversi dati. In particolare possiamo ricavare la differenza di PSNR tra l'immagine di input e quella di output, entrambe in relazione al ground truth. Inoltre possiamo misurare il valore di SSIM, sia per quello che riguarda le singole tiles, che per quello che riguarda l'immagine ricostruita.

Osservando i risultati dei performance indicator è possibile notare che il valore di PSNR tende a peggiorare tra input e output, introducendo del rumore aggiuntivo, allontanandosi così dal ground truth. SSIM, al contrario,

PSNR			
Nome del modello	input	tile-based	full-size
whole_ds_2	28.9528 dB	29.3738 dB	30.1938 dB
whole_ds_3	28.9528 dB	29.2908 dB	30.0684 dB
mini_whole_ds	29.1422 dB	30.0363 dB	30.1889 dB
desert_pix	28.6635 dB	28.4467 dB	31.5818 dB

SSIM			
Nome del modello	input	tile-based	full-size
whole_ds_2	0.1845	0.6606	0.1845
whole_ds_3	0.1845	0.6308	0.6384
mini_whole_ds	0.1972	0.6903	0.7010
desert_pix	0.1206	0.7523	0.7621

mostra notevoli miglioramenti fra output e input, questo perché SSIM tiene conto degli aspetti visivi dal punto di vista umano. Inoltre è bene notare che in tutti i casi, le immagini ricomposte ottengono misurazioni migliori rispetto alle tiles singole. Nonostante questo è bene notare che valori di SSIM inferiori a 0.94, indicano un alto livello di degrado tra output e ground truth 5.1.



(a) (b) (c)

Figura 5.1: Differenze strutturali (c) tra ground truth (a) e output del modello (b) con SSIM pari a 0.8156.

5.4 Analisi dei Risultati

Sebbene i risultati numerici dei test indichino che le immagini generate dal modello sono lontane dai render effettuati con 100 sample per pixel e con denoising, visivamente l'output del modello non si discosta eccessivamente dal ground truth mentre è molto lontano dall'input molto rumoroso e con difetti visivi. Lo scopo della generazione tramite AI non è quello di sostituire il rendering effettivo, ma quello di fornire un'idea del risultato finale in tempi ragionevoli, soprattutto se messi in relazione con i tempi reali di rendering. Dal punto di vista dell'efficienza, l'utilizzo della rete neurale è molto vantaggioso poiché il rendering completo da parte di Infinigen di una singola scena richiede in media 10 minuti, mentre il processo completo di elaborazione, inclusa la fase di denoising, richiede in media un singolo minuto.

5.4.1 Usabilità

Nonostante Blender sia facilmente integrabile con script Python, l'utilizzo di una rete neurale per il denoising è difficilmente integrabile direttamente nel motore di rendering e richiederebbe ulteriori lavori. Pertanto pix2pix può essere utilizzato alla fine del processo di rendering parziale.

Conclusioni

In conclusione, questa tesi ha dimostrato come l'applicazione di reti neurali e tecniche di intelligenza artificiale possa significativamente migliorare l'efficienza del processo di rendering, un settore di cruciale importanza per l'industria cinematografica e videoludica. Attraverso l'uso di Generative Adversarial Networks (GANs), è stato possibile ottenere immagini di alta qualità in tempi notevolmente ridotti rispetto ai metodi di rendering tradizionali, che spesso richiedono tempi di calcolo molto elevati.

I risultati sperimentali ottenuti mostrano che, sebbene le immagini generate dalle reti neurali non raggiungano ancora la perfezione dei rendering tradizionali con un alto numero di campioni per pixel e denoising, offrono comunque un'alternativa valida per ottenere anteprime rapide e visivamente coerenti. Questo rappresenta un notevole vantaggio, soprattutto in fasi di sviluppo e pre-visualizzazione, dove la velocità è cruciale per iterazioni rapide e decisioni tempestive.

L'uso delle reti neurali consente di ridurre i tempi di rendering da diverse ore a pochi minuti o addirittura secondi, rappresentando un notevole incremento in termini di efficienza e usabilità. Tuttavia, l'integrazione di tali tecniche nei motori di rendering esistenti non è priva di sfide. È necessario un ulteriore sviluppo per garantire una perfetta integrazione e sfruttare appieno il potenziale offerto dall'intelligenza artificiale, tenendo conto delle esigenze specifiche di diverse applicazioni industriali.

Oltre a migliorare l'efficienza del rendering, questo lavoro ha anche messo in luce nuove direzioni per la ricerca futura. Un'area di particolare interes-

se riguarda l'ottimizzazione dei modelli di rete neurale per specifici tipi di contenuti e condizioni di illuminazione, al fine di migliorare ulteriormente la qualità dell'immagine. Inoltre, l'esplorazione di tecniche di apprendimento non supervisionato e di autoapprendimento potrebbe portare a modelli ancora più versatili e capaci di adattarsi autonomamente a nuovi scenari e dati.

Bibliografia

[1] What is a neural network? https://www.ibm.com/topics/neural-networks.

[2] What is deep learning? https://https://www.ibm.com/topics/deep-learning.

[3] What are convolutional neural networks? https://www.ibm.com/topics/convolutional-neural-networks.

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[5] Chakravarty R Alla Chaitanya, Anton S Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. Interactive reconstruction of monte carlo image sequences using a recurrent denoising autoencoder. *ACM Transactions on Graphics (TOG)*, 36(4): 1–12, 2017.

[6] What is an autoencoder? https://www.ibm.com/topics/autoencoder.

[7] Steve Bako, Thijs Vogels, Brian McWilliams, Mark Meyer, Jan Novák, Alex Harvill, Pradeep Sen, Tony Derose, and Fabrice Rousselle. Kernel-predicting convolutional networks for denoising monte carlo renderings. *ACM Trans. Graph.*, 36(4):97–1, 2017.

[8] Alexander Raistrick, Lahav Lipson, Zeyu Ma, Lingjie Mei, Mingzhe Wang, Yiming Zuo, Karhan Kayan, Hongyu Wen, Beining Han, Yihan Wang, et al. Infinite photorealistic worlds using procedural generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12630–12641, 2023.

[9] Blender. `https://www.blender.org/`.

[10] Python. `https://www.python.org/`.

[11] Cycles open source production rendering. `https://www.cycles-renderer.org/`.

[12] Attila T. Áfra. Intel® Open Image Denoise, 2024. `https://www.openimagedenoise.org`.

[13] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[14] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[15] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 577–582. 2023.