ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Facoltà di Ingegneria
Corso di Laurea Magistrale di Ingegneria Meccanica
Tesi di Laurea in: Ingegnerizzazione del Prodotto

STUDIO E APPLICAZIONE DELL'INTELLIGENZA ARTIFICIALE AL REVERSE ENGINEERING DI COMPONENTI MECCANICI

Tesi di Laurea di:
Placucci Luca

Relatore:
Prof. Alfredo Liverani

Co-relatore:
Ing. Gian Maria Santi

Co-relatore:
Prof. Daniela Francia

Sessione II
ANNO ACCADEMICO 2024 – 2025

Al Pi.

Introduzione

Nell'ambito dell'ingegneria industriale, l'ampio utilizzo della modellazione 3D tramite software CAD ha prodotto un significativo miglioramento del processo di ingegnerizzazione del prodotto finito, sia in termini di risparmio di tempo sia in termini di qualità generale e diversificazione del risultato finale.

Il perfezionamento del prodotto finale ha richiesto modelli CAD sempre più performanti, al fine di ottenere rappresentazioni 3D dei prodotti finiti estremamente accurati. Questo però richiede l'impiego di notevoli risorse computazionali.

Da questa analisi si pone l'esigenza di raggiungere un compromesso tra la qualità del pezzo finale e la quantità di risolse impiegate per ottenerlo.

Per definire la risoluzione di un modello CAD si utilizzano le mesh poligonali, ovvero di piccoli poligoni in cui è stata suddivisa la superficie. La loro dimensione può variare e man mano che la dimensione dei lati si riduce, aumenta il numero di poligoni e di conseguenza la risoluzione finale del componente.

Lo scopo di questo elaborato è quindi quello di applicare l'intelligenza artificiale tramite l'utilizzo di una rete neurale, al fine di ottimizzare una mesh poligonale ottenuta a partire da un modello disegnato su un software CAD. Tale modello sarà sottoposto alla rete secondo due modalità: file del pezzo scansionato da uno scanner 3D e file del pezzo rotto tramite CAD. La rete dovrà essere in grado di ricostruire le mancanze derivate dalla scannerizzazione e le parti danneggiate.

Nei primi capitoli dell'elaborato, si affrontano i temi alla base della tesi, con un'analisi e un'introduzione alle conoscenze riguardanti il concetto di mesh, le reti neurali e il loro stato dell'arte e infine la strumentazione utilizzata.

Una volta fornite queste informazioni preliminari, si passa al corpo della tesi con la spiegazione del tipo di rete scelto e dei parametri impostati. I file CAD progettati, scansionati e rotti vengono sottoposti alla rete per la sessione di training, a seguito dalla quale si ottiene il pezzo ricostruito e ottimizzato. Vi è infine una fase di valutazione e di confronto dei risultati ottenuti.

Sommario

1. Reverse engineering

Per affrontare lo sviluppo di questo elaborato, occorre fornire delle conoscenze di base, al fine di comprendere meglio la metodologia utilizzata. In questo capitolo viene trattata la pratica del reverse engineering. È un processo industriale che consiste nel progettare un componente a partire dal risultato finale fino a riscostruire il modello digitale di input. Questo processo si cura di studiare a fondo il componente tridimensionale fisico, acquisendone le caratteristiche geometriche, le sue funzionalità e l'operatività, al fine di ottimizzare il componente in esame.

Il reverse engineering inizia quindi da un componente fisico di cui si vogliono studiare le proprietà. Al fine di poter ottenere un modello CAD digitale per poi poterlo rielaborare ed ottimizzare, occorre utilizzare degli strumenti che permettono tale acquisizione, per esempio attraverso l'uso di uno scanner 3D. Quest'apparecchiatura effettua la scansione del pezzo, registrando una nuvola di punti nello spazio e riscostruendo la superfice esterna per ottenere infine un modello CAD.

In questa maniera risulta più semplice e veloce ottenere un modello completo, in quanto lo scanner 3D può essere gestito anche da personale non qualificato, affidando il processo di scansione in massima parte allo strumento. Si riduce così la componete umana, ottenendo una buona qualità dell'output.

L'utilizzo del reverse engineering nell'ambito industriale permette un miglioramento generale del processo produttivo, presentando i seguenti vantaggi:

- Permette di apportare miglioramenti e risolvere eventuali difetti di componentistica già in produzione;
- Permette una più immediata innovazione del prodotto ottimizzandone i costi, riducendo così il rischio di obsolescenza;
- Permette di risolvere problematiche riguardanti la reperibilità dei componenti nelle scorte aziendali;
- L'analisi approfondita delle caratteristiche del prodotto permette di analizzare criticità potendone gestire in maniera più semplice la gestione a scorta.

Nel presente elaborato verrà utilizzato uno scanner 3D per la digitalizzazione dei componenti meccanici analizzati.

2. Mesh

Per rappresentare una qualsiasi geometria in un ambiente digitale, occorre scomporla in forme più semplici, che poi una volta unite insieme costituiscono la struttura del modello originale. Queste forme più semplici vengono definite con il nome di mesh poligonale e sono di fondamentale importanza nella modellazione 3D.

Le mesh poligonali trovano ampio uso nella computer grafica dove è necessario gestire sia immagini 2D, sia componenti 3D. Nell'ambito della progettazione meccanica, il reticolo assume un ruolo fondamentale sia nel design dei componenti sia nella parte della simulazione numerica, che prevede la valutazione della resistenza dei componenti. Le mesh poligonali sono inoltre sono utilizzate nell'intrattenimento e in campo artistico.

La scomposizione di un componente tridimensionale in facce aventi forme geometriche più piccole si rende necessaria, in quanto le schede video dei computer sono progettate per gestire le superfici tridimensionali secondo questa ottica di lavoro. In questo modo risulta possibile svolgere, tramite l'utilizzo di algoritmi, operazioni di modifica e rendering in maniera agevole.

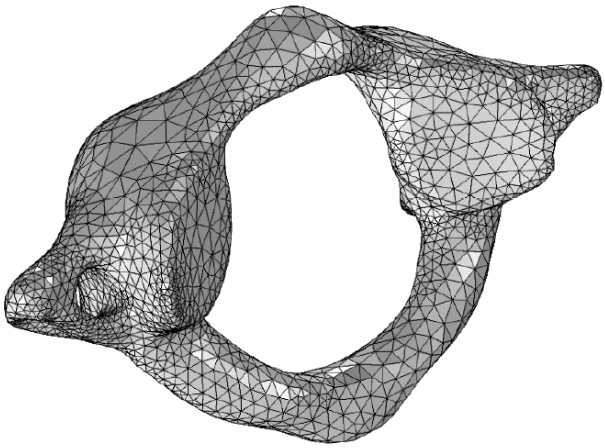


Figura 1. Esempio di mesh composta da poligoni triangolari.

Per definizione una mesh poligonale è una rappresentazione digitale di un oggetto tridimensionale composta da una collezione di poligoni e che quindi non ha proprietà di massa. Questi poligoni possono essere di varie forme, a seconda delle necessità e del capo applicativo. Le più comuni sono quella quadrangolare e triangolare per la loro facilità di utilizzo, ma possono essere impiegate anche poligoni di forma più complessa. Questi poligoni prendono il nome di facce che, essendo figure geometriche, sono composte da vertici e spigoli:

- Le facce: sono un insieme chiuso dei lati;

- I vertici: sono i vertici dei poligoni che compongono la mesh e hanno coordinate nello spazio (x,y,z);
- Gli spigoli: sono i segmenti che collegano i vertici di una faccia e ne delimitano la superficie;

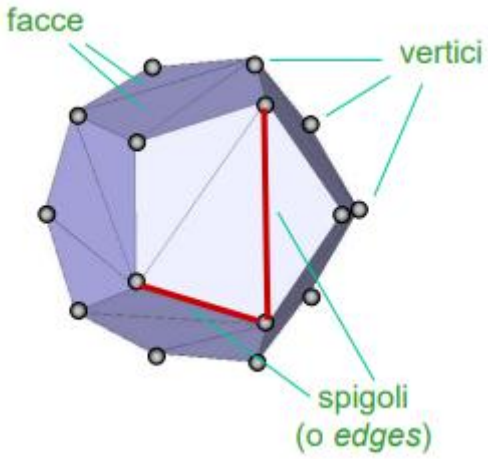


Figura 2. Facce, vertici e spigoli di una mesh.

Questi elementi geometrici, per fare parte di una rete poligonale, devono rispettare le seguenti proprietà:

1) I vertici devono essere tutti collegati da almeno due spigoli, così da avere una superficie chiusa e limitata;
2) Ogni spigolo deve essere condiviso da almeno una faccia;
3) Se due facce si intersecano, lo spigolo o il vertice dove si incontrano deve fare parte della rete poligonale;

Le facce in genere sono poligoni convessi e se sono tutte triangolari, la mesh poligonale prende il nome di *trimesh*, mentre poligoni con più vertici prendono il nome di polymesh. In particolare, nel caso in cui le facce siano tutte quadrilateri prendono il nome di *quad-mesh* o CG. Nel caso in cui invece non fossero tutti quadrilateri, vengono designate come *quad-dominant mesh*. Per altre forme geometriche generiche si usa il termine di mesh poligonale generica.

Con la mesh poligonale è possibile costruire delle forme geometriche semplici come cubi, cilindri, coni etc. e questi poligoni tridimensionali prendono il nome di *oggetto mesh*. Si definisce con il termine *tassellazione* della mesh la raccolta delle forme planari che ricoprono un oggetto mesh. Si tratta di un parametro molto importante per indicare la risoluzione della mesh, in quanto questa è data dal numero di vertici oppure di spigoli o di facce che la compongono.

Un tassellamento più grossolano implica avere una mesh meno fitta e quindi la superficie risulterà essere più approssimata. Questo implica la possibilità di perdere delle informazioni sulla geometria, nel caso in cui si abbia a che fare con forme

complesse dotate di molti dettagli. Nel caso contrario invece, con un tassellamento più fine, si ottiene una buona accuratezza della superficie, ma diventa importante l'onere computazionale impiegato.

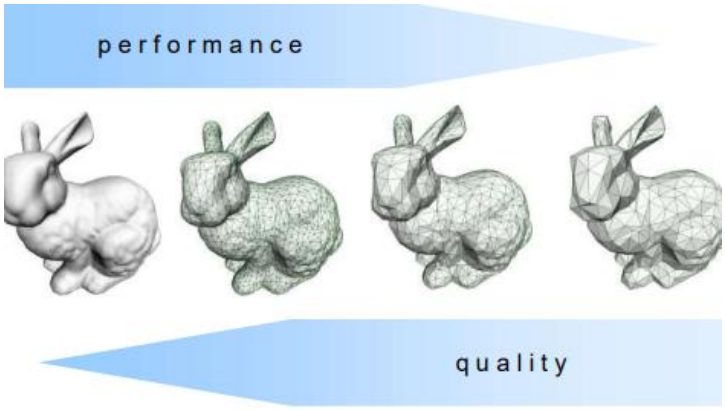


Figura 3. Compromesso tra qualità e performance.

Tuttavia, tramite opportuni software di modellazione della mesh, come Blender o Meshlab, e come si vedrà in seguito, è possibile modificare la mesh in modo da avere una risoluzione variabile sulla superficie del componente. Per esempio, si può affinare il tassellamento in parti dove sono presenti geometrie irregolari o dettagli di piccole dimensioni, dove una mesh troppo grossolana andrebbe inevitabilmente a perdere gran parte delle informazioni della geometria. In contrasto si può adottare un reticolo con un numero minore di poligoni su superfici piane, in modo da non creare un file CAD di dimensioni troppo grandi.

Un'ulteriore distinzione sulla mesh poligonale è possibile sulla base della disposizione degli spigoli e delle facce: se uno spigolo è condiviso solamente da una faccia, allora prende il nome di spigolo di bordo. In caso contrario, se uno spigolo è condiviso da due facce, allora lo spigolo è interno.

Da questa distinzione ne deriva che se una mesh poligonale non ha spigoli di bordo allora si dice chiusa, se invece ne presenta almeno uno, la mesh viene definita aperta.

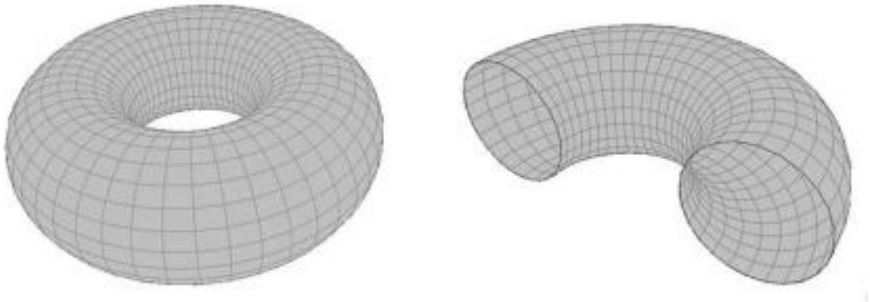


Figura 4. Mesh chiusa e aperta.

2.1 Manifold

Il termine manifold indica una caratteristica della mesh riguardante la disposizione degli spigoli in essa. Si definisce uno spigolo *two-manifolds* se questo viene condiviso da due facce, in modo da ottenere una superficie. Tuttavia, se lo spigolo è condiviso da più facce viene detto *not-two-manfolds*. Questo parametro risulta essere molto importante, in quanto a livello computazionale avere una mesh manifold potrebbe essere un requisito dell'algoritmo di elaborazione.

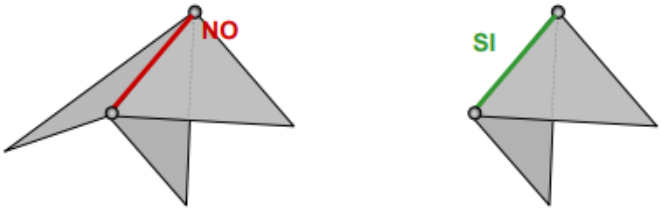


Figura 5. A sinistra spigolo not-two-manifold. A destra, spigolo two-manifold.

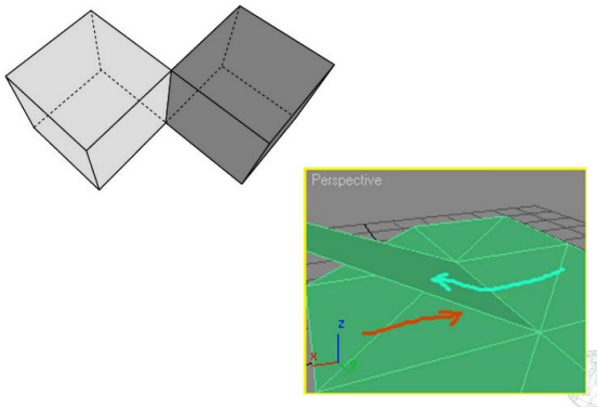


Figura 6. Ulteriore esempio di spigoli not-two-manifolds.

La definizione di manifold si può estendere anche ai vertici che costituiscono il reticolo della mesh. A partire dalle facce, queste si dicono *adiacenti* se condividono un spigolo. Viene definito *fan* l'insieme delle facce adiacenti che condividono un vertice e il vertice in questione è detto *two-manifold*. Se non sono rispettate queste condizioni il vertice è banalmente detto *not-two-manifold*.

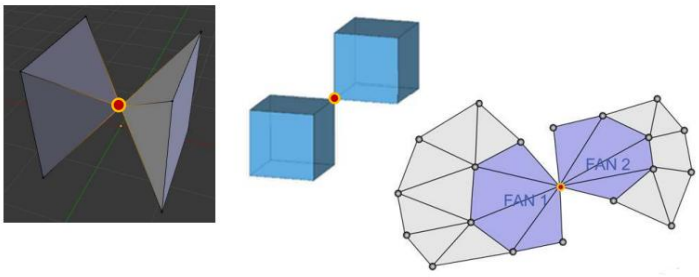


Figura 7. Vertice two-manifold.

2.2 Mesh triangolare

Come affermato in precedenza, le forme dei poligoni che compongono la mesh possono assumere varie forme, tuttavia quella più utilizzata per la modellazione risulta essere la mesh triangolare o anche detta trimesh. La scelta di questa tipologia di poligono presenta i seguenti vantaggi:

- La forma triangolare è la figura piana più semplice in cui tutti i punti risultano essere complanari (dalla geometria euclidea, per tre punti passa un solo piano);
- È sempre possibile ricondursi a triangoli in caso di poligoni più complessi;
- Permette una semplice interpretazione e interpolazione;
- Data la semplicità geometrica, esistono molti metodi per costruirla;
- Permette una renderizzazione diretta dall'hardware, così da poter riscostruire l'oggetto reale su cui è stata fatta la mesh.

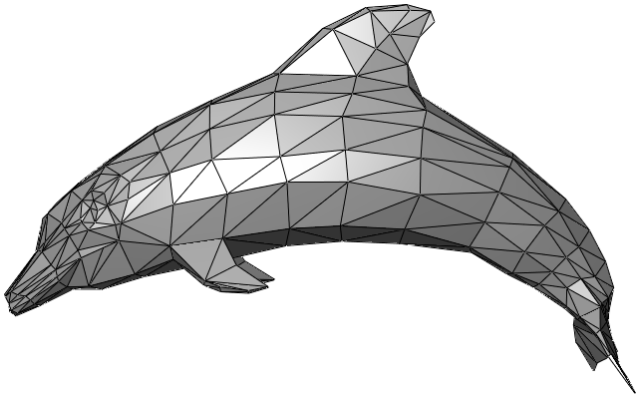


Figura 8. Mesh con poligoni triangolari.

2.2.1 Composizione della mesh triangolare

La mesh triangolare si presenta a graficamente come un insieme di triangoli senza una particolare parametrizzazione geometrica. Con questa figura geometrica è possibile definire qualsiasi punto al loro interno in funzione delle coordinate dei vertici. Infatti, dato un triangolo avente vertici [a,b,c] un generico punto p può essere scritto come:

$$p = \alpha \cdot a + \beta \cdot b + \gamma \cdot c$$

Con i parametri $\alpha + \beta + \gamma = 1 \quad \beta, \gamma \geq 0$.

Con un triangolo arbitrario aventi vertici [u,v,w], si definisce una mappatura lineare:

$$f\colon \mathbb{R}^2 \to \mathbb{R}^3 \qquad \text{con } \alpha \cdot u + \beta \cdot v + \gamma \cdot w \quad \to \quad \alpha \cdot a + \beta \cdot b + \gamma \cdot c$$

Sulla base di questa relazione, definendo per ogni vertice una posizione in 2D è possibile ottenere una parametrizzazione globale della mesh.

Questo reticolo poligonale può essere visto come una struttura a grafo formata da un insieme di vertici e facce collegate tra loro. Di seguito vengono definiti questi parametri:

- $V = \{v_1, v_2, \dots\dots\dots, v_n\}$ l'insieme dei vertici che compongono la mesh;
- $F = \{f_1, f_2, \dots\dots\dots, f_F\}$ con $f_i \in VxVxV$ l'insieme delle facce che formano il reticolo;
- $E = \{e_1, e_2, \dots\dots\dots, e_F\}$ con $e_i \in VxV$ l'insieme degli spigoli che connettono i vertici tra di loro;

Riportando quindi un insieme generico di punti p_i nel piano tridimensionale in $\mathbb{R}^3$, e avendo ciascun vertice $v_i \in V$, si ottiene la seguente relazione:

$$P = \{p_1, p_2, \dots\dots\dots, p_V\}, \qquad p_i \coloneqq p(v_i) = \begin{pmatrix} x(v_i) \\ y(v_i) \\ z(v_i) \end{pmatrix} \in \mathbb{R}^3$$

Si ha inoltre che ogni faccia $f_i \in F$ corrisponde ad un triangolo definito dalle sue posizioni dei vertici.

Una volta parametrizzati i punti contenenti un triangolo, risulta possibile approssimare una superficie con la funzione lineare f definita in precedenza. Questa approssimazione produce un errore avente funzione $O(h^2)$, dove h indica la lunghezza massima del bordo.

Si osserva dalla relazione che avendo un andamento quadratico, l'errore diventa ¼ se si dimezza la dimensione degli spigoli. Ne consegue quindi che l'errore è inversamente proporzionale al numero delle facce, in quanto questa riduzione di dimensione dello spigolo quadruplica il numero dei triangoli, affinando così il reticolo.

Inoltre, si osserva che per ottenere una migliore approssimazione occorre campionare i vertici in maniera adattativa. In altre parole, si aumenta la densità dei vertici in zone che presentano superfici curve e riducendone il numero in parti piane del reticolo.

Risulta possibile relazionare gli elementi geometrici che costituiscono un reticolo (vertici, spigoli e facce), attraverso la formula di Eulero:

$$V - E + F = 2(1 - g)$$

dove:

- V indica il numero dei vertici;
- E indica il numero degli spigoli;

- F indica il numero delle facce;
- g indica il genere della superfice e rappresenta il numero "buchi" della superficie. Un toro, per esempio, ha g=1.

Nella pratica, il termine g è trascurabile rispetto agli altri parametri della formula e di conseguenza la parte destra della relazione risulta irrilevante.

Si definisce valenza di un vertice il numero di spigoli incidenti in esso. Considerando una mesh manifold, dalla relazione di Eulero, si ricavano le seguenti considerazioni pratiche:

- $F \approx 2 \cdot V$, il numero dei triangoli è pari al doppio dei vertici;
- $E \approx 3 \cdot V$, il numero degli spigoli è pari al triplo dei vertici;
- Il numero medio di spigoli incidenti per ogni vertice è pari a 6 per quelli interni e 4 per i vertici limite.

Risulta ora utile definire la rappresentazione delle superfici implicite o volumetriche in modo da indicare se un punto si trova all'interno o all'esterno della superfice esterna di un oggetto solido. Per fare ciò si utilizza una funzione implicita a valori scalari:

$$F: \mathbb{R}^3 \rightarrow \mathbb{R}$$

F utilizza un potenziale dove si assegna il livello zero alla superficie del solido e si attribuisce un segno positivo e negativo ai punti rispettivamente all'esterno e all'interno dell'oggetto. Mentre, come affermato in precedenza, tutti i vertici che giacciono sulla superficie del solido hanno potenziale nullo.

Questa procedura trova impiego negli algoritmi di riparazione della mesh, in quanto se la funzione F è continua, la superficie implicita non presenta fori. Inoltre essendo una funzione che rappresenta un insieme di livelli potenziali, non si possono verificare intersezioni tra le facce.

In aggiunta, tramite la funzione F è possibile deformare le superfici implicite aumentando o diminuendo i valori della funzione localmente. Si può inoltre modificare facilmente la topologia e la connettività della mesh – ovvero la forma geometrica dei reticoli, come per esempio triangolare e quadrangolare – in quanto la funzione F non dipende da questi ultimi. Da un punto di vista pratico una funzione implicita può essere rappresentata da una griglia voxel Dove il voxel è definito come l'unità di base nella modellazione 3D in un sistema di rappresentazione volumetrica. Questa rappresentazione semplifica anche i calcoli e il controllo sull'errore finale in fase di elaborazione.

La limitazione di questo metodo si trova nella difficoltà di rendering e nell'utilizzo delle texture, ovvero l'insieme di dati visivi presenti (come la colorazione, il pattern, i

riflessi,…). In particolare, risulta molto complesso incollarle in superfici implicite in evoluzione.

2.3 Struttura dei dati mesh

Per utilizzare e compiere operazioni di modellazione sulla mesh tramite algoritmi, occorre creare una struttura dati in modo da immagazzinare le informazioni del reticolo in maniera efficiente. Per fare ciò, in primo luogo bisogna determinare la tipologia di mesh poligonale che si dispone e in secondo luogo, occorre conoscere quali tipologie di algoritmi si andranno a implementare nei dati e come saranno gestite le informazioni durante le elaborazioni.

Le informazioni che definiscono la mesh a cui prestare attenzione sono:

- Geometria: al cui interno sono immagazzinate la disposizione spaziale dei vertici sottoforma di coordinate (x,y,z) determinando la risoluzione del reticolo;
- Connettività: indica come sono connessi tra loro i vertici e come sono disposti i poligoni;
- Attributi: ovvero la presenza di eventuali caratteristiche aggiuntive sulla superficie come colore, texture e ombreggiature.

Facendo riferimento ad una tipologia di mesh triangolare o trimensh, essendo la più comune, si riportano di seguito le tipologie di strutture di dati più comuni.

2.3.1 Strutture dati basati sulle facce

Face-set: viene utilizzato nei file stereolitografia (STL). Si tratta della struttura dati più semplice, memorizza l'insieme delle facce poligonali dalle posizioni dei loro vertici. In questo caso si utilizza una mesh triangolare e si immagazzinano le tre posizioni dei vertici che compongono una singola faccia, che richiedono così 36 byte di memoria per ogni triangolo.

Questo sistema non tiene conto della connettività della mesh, ossia non considera la disposizione spaziale delle facce vicine. Il face-set non risulta adatto alla maggior parte delle applicazioni perché non è possibile ricavarsi la connettività della mesh in maniera diretta e questo genera ridondanza, in quanto i dati dei vertici sono replicati tante volte quanto il grado degli stessi.

Tuttavia, con la struttura dati index-face-set, in cui ogni faccia è definita attraverso un insieme di indici che fanno riferimento ai vertici di un array di vertici, si elimina la

ridondanza. Questo tipo di architettura è utilizzato nei modelli 3D aventi formato .obj e .stl.

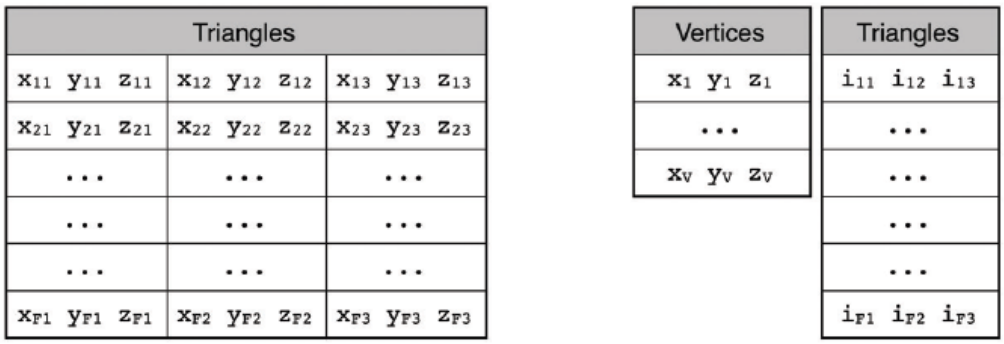


Figura 9. Esempio di face-set a sinistra. Struttura index-face-set a destra.

2.3.2 Strutte dati basati sui bordi o spigoli:

- **Winged-edge** e **quad-edge**: sono adatte anche per altri tipologie di mesh non triangolari. Come intuibile dal nome, questa struttura dati si basa sui bordi, in quanto questi forniscono le principali informazioni sulla connettività della mesh. Nel caso di architettura Winged-edge, per ogni bordo vengono memorizzate informazioni nell'intorno dello spigolo, per esempio relative ai suoi vertici di estremità, alle facce incidenti, al bordo precedente e successivo situati nelle facce in cui è compreso. Le facce e i bordi memorizzano i dati relativi a uno dei bordi incidenti. Rispetto al caso precedente la richiesta di memoria per immagazzinare tutte queste informazioni risulta maggiore.

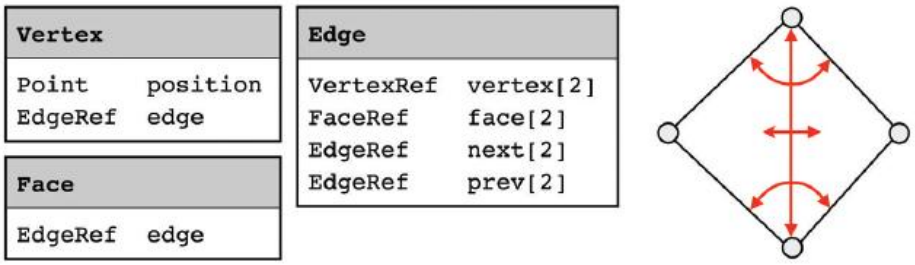


Figura 10. Struttura dati winged-edge.

- **Halfedge**: questa struttura dati divide ogni spigolo in due spigoli orientati. Questa architettura si presta alla rappresentazione di mesh poligonali che sono sottoinsiemi two-manifold orientabili, ossia non presentano spigoli o vertici complessi. Ogni halfedge è associato ad un vertice, ad una faccia e ad una direzione, creando così un collegamento bilaterale tra le facce, halfedge e vertici. Questa struttura dati risulta essere molto efficiente in termini di elaborazione e di impiego di memoria, in quanto presenta il grande vantaggio di immagazzinare poche ma significative informazioni, come per esempio sul successivo halfedge

o su quello opposto. Inoltre, implementando l'uso di indici, si riesce a ridurre ulteriormente il consumo di memoria.

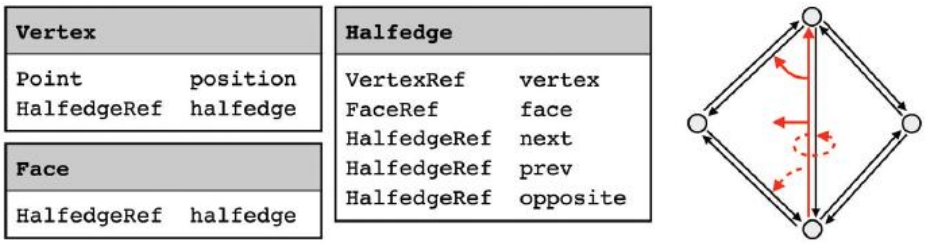


Figura 11. Struttura dati halfedge.

- **Directed edge**: consiste in una variante del halfedge e ottimizzata per la mesh triangolare. La struttura si basa su indici che fanno riferimento a ciascun elemento della mesh. In aggiunta, questi indici codificati in maniera opportuna possono contenere dati sulla connettività della rete.

2.4 Elaborazione e ottimizzazione della mesh

Dal momento in cui sono note le strutture dei dati su come vengono rappresentate e immagazzinate le informazioni sulle mesh, si può procedere all'elaborazione e alla manipolazione del reticolo poligonale. Questa procedura si rende necessaria per ottenere un buon compromesso tra risoluzione della mesh e carico computazionale.

Occorre considerare che i dati sulla mesh contenuti nella struttura possono presentare delle alterazioni come rumore o mancanze dovute principalmente alla provenienza del file. Per esempio, un file proveniente da una modellazione CAD esclusivamente su software presenterà meno discontinuità di un file ottenuto da una scansione 3D di un pezzo fisico.

La rielaborazione e l'ottimizzazione della mesh hanno pone come obiettivi quindi l'ottenere una buona risoluzione, mantenere basso l'onere computazionale e aggiustare il reticolo dove danneggiato.

Per risolvere questi problemi si utilizzano approcci differenti in base alla tipologia di problema che si va ad affrontare. Lo smooting, per esempio, si utilizza per lisciare il più possibile le superfici e si avvale di due ulteriori tecniche:

1. Denosizing → per la riduzione del rumore;
2. Hole filling → per la chiusura di eventuali buchi nella mesh.

Per quanto riguarda l'ottimizzazione del reticolo, si utilizzano approcci come:

1. Remeshing → riguarda la ricostruzione di una nuova mesh sostituendola alla precedente;
2. Decimazione → si diminuisce il numero di poligoni della mesh, riducendo così la memoria richiesta;
3. Riparazione della mesh → si utilizza per riparare eventuali mancanze o irregolarità attraverso l'uso di opportuni algoritmi.

Queste tecniche vengono analizzate in maniera più approfondita di seguito.

2.4.1 Smoothing

Basandosi su una mesh triangolare lo smoothing si occupa di levigare le superfici attraverso il calcolo di funzioni del tipo:

$$f: S \rightarrow \mathbb{R}^d$$

Questa operazione sul reticolo poligonale ricopre un ruolo fondamentale per le successive attività di rielaborazione della mesh, in quanto produce una geometria di input corretta per gli algoritmi che competono alle attività successive di ottimizzazione.

Figura 12. Tecnica di smooting.

La funzione f ha lo scopo di descrivere le posizioni dei vertici, le coordinate della trama e gli spostamenti dei vertici, in modo da parametrizzare il reticolo poligonale.

Come affermato in precedenza la levigatura della mesh si compone di:

- **Denosizing**: si occupa di ridurre il rumore ad alta frequenza nella funzione f. Il rumore ad alta frequenza potrebbe danneggiare le posizioni dei vertici, alterando così il reticolo. Questa problematica si riscontra quando si effettua una scansione in 3D di un componente fisico. Per risolvere questo problema e mantenere solo i dati a basse frequenze che mantengono la forma complessiva, occorre effettuare una posizione di filtraggio per mezzo di filtri passa-basso che vengono applicati alla funzione f. Da qui viene l'esigenza di utilizzare la trasformata di Fourier in modo da applicarla alla mesh poligonale.
- **Carenatura**: con questa operazione si rimuove il rumore ad alta frequenza della f e si leviga il più possibile la superficie così da renderla liscia.

Per quanto riguarda la tecnica di **denosizing**, ci si avvale dell'utilizzo della trasformata di Fourier, che definisce una funzione $f : \mathbb{R} \to \mathbb{C}$ dalla sua rappresentazione nel dominio spaziale al dominio delle frequenze Questa trasformazione può essere scritta attraverso le formule di trasformata e antitrasformata:

$$F(\omega) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i \omega x} dx$$

$$f(x) = \int_{-\infty}^{\infty} F(\omega) e^{2\pi i \omega x} d\omega$$

In questo caso la trasformata può essere visto come un cambiamento di base tramite la proiezione ortogonale del vettore f sui vettori di base $e\omega$.

$$f(x) = \sum_{\omega=-\infty}^{\infty} \langle f, e\omega \rangle \; e\omega$$

Dove il coefficiente scalare $\langle f, e\omega \rangle$ corrisponde al termine $F(\omega)$ e descrive quale ampiezza della ω è contenuta nel segnale $f(x)$. Dato che le coordinate $F(\omega)$ corrispondono alle frequenze, si può pensare di applicare un filtro passa-basso che semplicemente tagli tutte le frequenze sopra un certo valore definito come ω_{max}. Quello che si ottiene è la funzione filtrata $\tilde{f}$:

$$\tilde{f}(x) = \int_{-\omega_{min}}^{\omega_{max}} \langle f, e\omega \rangle \; e\omega \; d\omega$$

dove tutte le frequenze $|\omega| < \omega_{max}$.

Tuttavia non è possibile applicare direttamente la trasformata di Fourier sulle funzioni che operano con manifold, si osserva però che le onde complesse $e\omega$, così come le funzioni seno e coseno, sono autofunzioni dell'operatore di Laplace:

$$\Delta\left(e^{2\pi i \omega x}\right) = \frac{d^2}{dx^2} \; e^{2\pi i \omega x} = -(2\pi\omega)^2 \; e^{2\pi i \omega x}$$

Data una funzione ei, questa è autofunzione del laplaciano con autovalore λi se $\Delta ei = \lambda iei$. Da qui scegliendo le autofunzioni dell'operatore Laplace-Beltrami Δ, essendo possibile il calcolo della sua discretizzazione, l'operatore fornirà la generalizzazione della trasformata di Fourier su mesh triangolari discrete.

Si passa ora alla discretizzazione della mesh triangolare. Data una funzione continua $f(x)$, si sostituiscono i valori con il vettore di n vertici:

$$f : S \to \mathbb{R} \;\; \to \;\; \{f(v_1), f(v_2), \dots \dots \dots, f(v_n)\}^T$$

Si procede con il calcolo del laplaciano per ogni vertice tramite l'operatore di Laplace-Beltrami Δ:

$$\begin{pmatrix} \Delta f(v_1) \\ \vdots \\ \Delta f(v_n) \end{pmatrix} = \mathbf{L} \begin{pmatrix} f(v_1) \\ \vdots \\ f(v_n) \end{pmatrix}$$

Dove il termine L indica la matrice di Laplace-Beltrami, che contiene in ogni riga i-esima, i pesi per discretizzare il vertice v_i.

$$\Delta f(v_i) = \sum_{v_j \in \mathcal{N}_1(v_i)} w_{ij} \left(f(v_j) - f(v_i) \right)$$

I pesi sono indipendenti dalla valenza del vertice e sono scelti in modo che la matrice L sia simmetrica. Gli autovalori di L prendono il nome di *frequenze naturali*, mentre gli autovettori si definiscono come *vibrazioni naturali* della maglia triangolare.

Data che L è semidefinita positiva e simmetrica, i suoi autovettori costituiscono una base ortogonale di $\mathbb{R}^n$ così da rappresentare ogni vettore $f=\{f_1, f_2, \dots \dots \dots, f_n\}^T$ con questa base:

$$f = \sum_{i=1}^{n} \langle ei, f \rangle \, ei \qquad con \ \langle ei, f \rangle = e_i^T f$$

Si può ottenere una funzione filtrata con solo le funzioni a bassa frequenza costituita da $m < n$ autovettori:

$$\tilde{f} = \sum_{i=1}^{m} \langle ei, f \rangle \, ei$$

Con questi passaggi si è ottenuto un filtraggio della geometria della mesh.

Al livello computazionale, esistono altri metodi più pratici. Dato che la trasformata di Fourier inversa di una gaussiana nel dominio delle frequenze fornisce una gaussiana nel dominio spaziale, si può evitare di passare dalla trasformata e calcolare lo smoothing direttamente sul dominio spaziale. In altre parole l'elaborazione viene effettuata direttamente sulla mesh triangolare. In questo caso si procede con lo smorzamento delle frequenze moltiplicandole per un kernel gaussiano e questo metodo prende il nome di **flusso di diffusione**.

Il flusso di diffusione è un modello matematico descritto dall'equazione:

$$\frac{\partial f(x,t)}{\partial t} = \lambda \Delta f(x,t)$$

Si può applicare l'equazione di diffusione in modo semplice per livellare una funzione $f: S \to \mathbb{R}$ su una superficie maniforld S, tramite la sostituzione dell'operatore di Laplace regolare con l'operatore di Laplace-Beltrami Δ.

La posizione dei vertici di Laplace-Beltrami corrisponde alla normale della curvatura media e tutti i vertici si muovono di una quantità definita da questo valore di curvatura e l'equazione di diffusione prende il nome di **flusso della curvatura media**.

Si possono implementare anche flussi laplaciani di ordine superiore, ma richiedono un più ampio bacino di vertici per fornire buoni risultati e perciò si tratta di operazioni onerose a livello computazionale che tuttavia forniscono migliori proprietà di filtraggio.

La **carenatura** superficiale, come affermato in precedenza, si occupa di ottenere una superficie più liscia possibile. Per arrivare a questo obiettivo, si segue il principio della forma più semplice, cioè la superficie non deve presentare dettagli e oscillazioni eliminabili. Per fare ciò si inseriscono dei vincoli estetici e si introduce una funzione di energia da minimizzare.

In questo modo con opportune sostituzioni si passa da un sistema non lineare ad uno lineare, più semplice da risolvere. Inoltre, si ottiene un risultato migliore della mesh, in quanto è meno dipendente dalla topologia iniziale.

2.4.2 Remeshing

Le mesh ottenute attraverso software di modellazione non sempre forniscono una mesh adatta alle successive attività di analisi. Questo perché potrebbero presentare problematiche dovute ad un eccessivo campionamento, che potrebbe richiedere un onere computazionale troppo gravoso, oppure presentare una topologia della mesh incompatibile con l'algoritmo di elaborazione (come nel caso in cui venga presentata una mesh quadrangolare a un algoritmo che necessita di una mesh di input triangolare).

Per risolvere questi problemi, occorre ricavare una nuova mesh e questa attività viene definita remeshing, che oltre a ricostruire una nuova mesh, deve anche fornire una buona approssimazione della superfice originale.

Per ricavare una nuova mesh da una esistente si presume che il reticolo di input sia triangolare manifold e per l'elaborazione si utilizzano algoritmi dedicati.

Per capire come affrontare il problema occorre considerare la mesh di partenza sotto vari aspetti, in base a:

- **Topologia del poligono**: in generale quelli più usati sono mesh triangolari e quadrangolari. In fase di remeshing le quadrangolari di input possono essere

facilmente trasformate in triangoli aggiungendo una diagonale. Mentre per effettuare il passaggio inverso, risulta più complessa l'operazione e nella maggioranza dei casi si ottiene una dominanza quadrupla non totale.

Figura 13. Da quadrimesh a trimesh.

- **Forma dell'elemento**: i poligoni che compongono il reticolo possono essere isotropi o anisotropi. Gli elementi isotropi mantengono una geometria uniforme in tutte le direzioni tendendo alla forma dell'equilatero. Mentre in poligoni anisotropi, la dimensione dei lati varia considerevolmente nelle diverse direzioni. Forme isotrope ottengono migliori risultati e sono preferibili per le simulazioni numeriche, mentre forme anisotrope sono da preferire per una migliore approssimazione della forma e impiegano meno facce a parità di risoluzione rispetto a quelle isotrope.

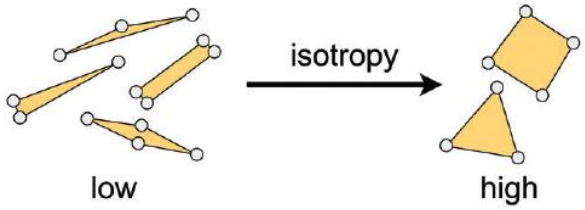


Figura 14. Facce a bassa isotropia a sinistra e alta isotropia a destra.

Durate il processo di remeshing assume un aspetto importante anche come le facce si dispongono nel reticolo. Può essere utile, per risparmiare il consumo di memoria, adottare un nuovo reticolo a campionamento non uniforme, in maniera da avere poche facce su una superficie piana e disporre un campionamento di vertici maggiore su aree che presentano dettagli e forte curvatura. Se si adottasse una mesh uniforme, si avrebbero su tutta la superficie la stessa distribuzione dei vertici, aumentando così il numero delle facce e l'impiego di memoria risulta maggiore. Quest'ultimo aspetto si ripercuote a cascata su tutte le operazioni successive.

Inoltre, come si può osservare nella pratica, ricavare una nuova mesh implica un processo di ricampionamento dei vertici, per cui riveste un ruolo importante l'allineamento e l'orientamento degli elementi per rappresentare al meglio la superficie originale.

Quanto visto riporta le caratteristiche locali della mesh, in particolare analizzando i poligoni singoli che costituiscono il reticolo. Passando ad una visione globale della mesh è possibile fare le seguenti distinzioni:

- **Mesh regolari**: se i vertici interni hanno tutti valenza 6, mentre quelli limite hanno tutti valenza 4 nel caso di trimesh. Mentre se la maglia è quadrangolare la valenza vale rispettivamente 4 e 3;
- **Mesh irregolari**: non presentano alcuna regolarità nella loro connessione;
- **Mesh semiregolari**: derivano dalla suddivisione di una mesh grossolana e il numero delle maglie irregolari è molto contenuto e costante dopo un raffinamento uniforme;
- **Mesh molto regolare**: in questo caso i vertici sono a maggioranza regolari e non implica che derivino da una suddivisione a differenza delle semiregolari;

Le mesh regolari sono adatte ad essere generate per poche forme geometriche regolari, per tutti gli altri casi si necessita di una fase di preelaborazione della mesh; tuttavia, hanno il vantaggio di essere utilizzate facilmente per la simulazione numerica.

Le mesh semiregolari trovano impiego nelle modellazioni con multi-risoluzione, riuscendo ad ottenere una parametrizzazione della mesh a partire da un reticolo grossolano di input.

Si procede ora all'analisi degli algoritmi che svolgono l'attività di remeshing. La logica di funzionamento con cui si svolge il processo, consiste in una prima fase di calcolo dei vertici della mesh originale e in seguito l'algoritmo riposiziona i nuovi vertici per la qualità della mesh, mantenendo però una corrispondenza con i punti del reticolo di input. Proprio mantenere una corrispondenza tra questi vertici rappresenta il problema maggiore per questi algoritmi. Per ovviare a questa difficoltà si utilizzano vari metodi:

- Parametrizzazione globale: si effettua in primo luogo una parametrizzazione del reticolo poligonale in 2D, dove i vertici vengono riposizionati e in seguito vengono trasportati in 3D;
- Parametrizzazione locale: si mantiene la parametrizzazione locale nell'intorno del punto della nuova mesh, se si posiziona un nuovo vertice e questo ricade fuori dall'intorno del primo punto si ricalcola un nuovo intorno;
- Proiezione: il nuovo vertice viene proiettato sull'elemento più vicino della mesh iniziale;

In generale la parametrizzazione globale è più onerosa e risente molto delle discontinuità del reticolo, producendo in output distorsioni della geometria. La parametrizzazione locale invece risulta essere più stabile ottenendo una più alta qualità finale, ma di contro richiede un onere di memoria molto alto.

Si introducono ora le strutture dati utilizzate dagli algoritmi per il remeshing:

- **Diagrammi di Voronoi**: dato un insieme di punti $P = \{p_1, \dots \dots, p_n\} \in \mathbb{R}^d$, si associa ad ogni punto p_i la sua regione di Voronoi $V(p_i)$, tale che:

$$V(p_i) = \{x \in \mathbb{R}^d : \|x - p_i\| \leq \|x - p_j\|, \qquad \forall j \neq i\}$$

  L'insieme delle regioni non vuote di Voronoi e delle loro facce, insieme alle loro relazioni di incidenza, prende il nome di diagramma di Voronoi.
- **Triangolazione di Delauney**: rappresenta la struttura duale al diagramma di Voronoi. In altre parole, in due dimensioni, ogni triangolo di Delauney aventi vertici [p,q,r], è duale a un vertice di Voronoi dove $V(p), V(q), V(r)$ si incontrano.

La triangolazione di Delauney è importante perché essendo duale al diagramma di Voronoi, gode di diverse proprietà che nella generazione della mesh risolve il problema di avere angoli tra gli spigoli troppo piccoli che causerebbero inevitabilmente problemi nelle analisi agli elementi finiti.

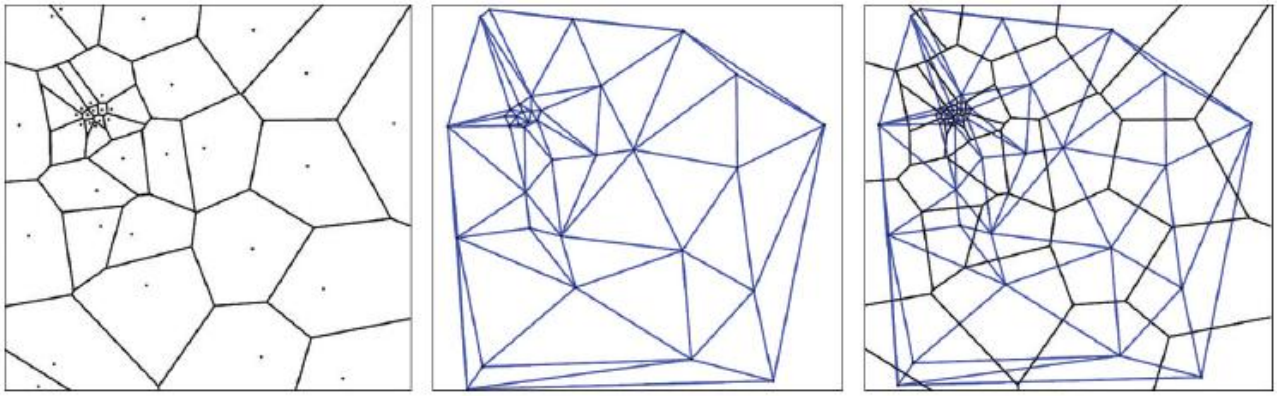


Figura 15. A sinistra, diagramma di Voronoi. Al centro, triangolazioni di Delauney. A destra, la loro sovrapposizione per lo stesso set di vertici.

Considerando ora il caso di una **mesh triangolare isotropa**, nella quale i lati dei triangoli tendono ad avere dimensioni simili con un campionamento uniforme su tutta la superficie, il remeshing può essere effettuato modificando gradualmente le facce. Questa operazione avviene tramite tre principali tipologie di algoritmi:

- **Greedy remeshing**: in questi algoritmi si esegue una modifica locale alla volta, inserendo dei vertici fino al raggiungimento dell'obiettivo. Più nel dettaglio il remeshing si basa sulla perfezione e sul filtraggio della triangolazione di Delauney. In particolare in ogni fase di affinamento, viene inserito nella triangolazione un punto preso dalla superficie di input. Il punto in questione viene scelto tra l'intersezione della superficie di iniziale e gli spigoli di Voronoi. Il diagramma di Voronoi viene utilizzato quindi per sondare la superficie di input durante il processo di remeshing. Il vantaggio di questo metodo risiede nella garanzia dell'output. In altre parole, la mesh che si ottiene garantisce il non autointersecarsi dei triangoli proprio per il fatto di provenire da una

triangolazione di Delauney tridimensionale. Inoltre, è un metodo robusto perché è indipendente dalla parametrizzazione locale o globale del reticolo.

- **Remeshing variazionale**: questo metodo oltre a ricavare una nuova mesh, ricerca anche una qualità migliore del reticolo. Come abbiamo visto in precedenza, per l'ottimizzazione occorre determinare una funzione energetica e trovare i valori che minimizzano questa energia per mezzo di un risolutore.
  Con il remeshing si è osservato come campionamenti isotropi in 2D portino a una triangolazione ben modellata. Cosa che non accade in tre dimensioni, dal momento che possono verificarsi tetraedri deformati. Con un remeshing isotropo ci si può ricondurre al campionamento di punti isotropi, che consiste nel distribuire uniformemente un insieme di punti della mesh di partenza. Per questa assegnazione dei vertici si utilizza la tassellazione di Voronoi centroidale, nella quale ogni sito coincide con il centroide dei suoi Voronoi della regione. Il centroide viene definito come centro di massa. L'algoritmo svolge in maniera iterativa le fasi di costruzione della tassellatura di Voronoi corrispondenti ai punti p_i, calcolo dei centroidi c_i e spostando i punti p_i, facendoli coincidere con c_i, fino ad arrivare a convergenza.
- **Remeshing incrementale**: questo metodo si basa sull'utilizzo di un algoritmo ad alta efficienza per costruire una mesh triangolare isotropa. La logica di elaborazione consiste nel determinare una lunghezza target di uno spigolo e si modifica la lunghezza dei bordi fino a quando tutti non raggiungono approssimativamente la lunghezza target. Da notare che per ottenere in output una mesh triangolare isotropa, occorre normalizzare anche le valenze dei vertici durante il remeshing.
  Questo approccio è facile da applicare e robusto, in quanto non necessita di parametrizzazione del reticolo.
  Un aspetto importante riguarda la conservazione delle caratteristiche originali della mesh durante l'operazione di remeshing. Questa problematica viene risolta implementando alcune regole pratiche basate sulla connettività del reticolo nell'algoritmo di elaborazione.
  Risulta inoltre possibile utilizzare dei pesi per ottenere una distribuzione dei vertici più adattativa alla superficie e migliorando sensibilmente la qualità del risultato finale.

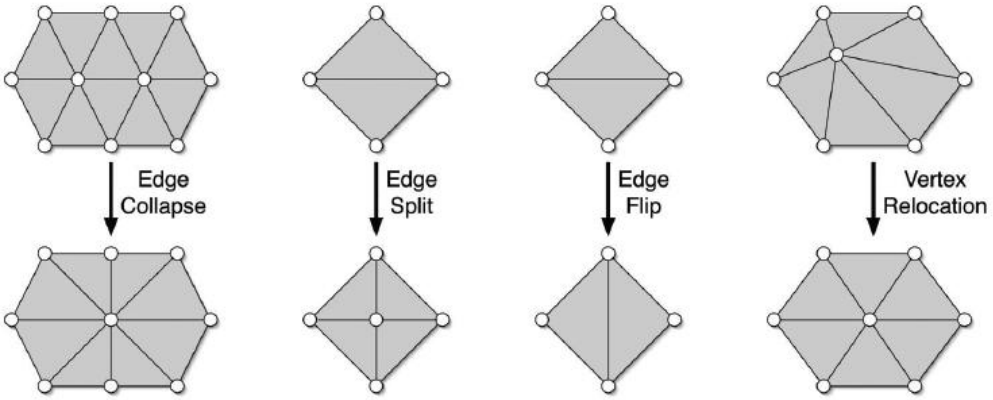


Figura 16. Tecniche di remashing incrementale.

2.4.3 Decimazione della mesh

La decimazione della mesh si pone come obiettivo quello di modificare il reticolo mesh diminuendo il numero di vertici, spigoli e facce. In sostanza si ricava una nuova mesh a partire da una esistente, ma che non è ottimizzata per le successive attività di analisi numerica.

Questa operazione si rende necessaria quando si ha un'eccessiva risoluzione della mesh, la quale presentando un alto campionamento può richiedere un ingente quantitativo di risorse computazionali, oppure nel caso si vogliano ridurre i tempi di elaborazione mantenendo però un'elevata qualità dell'output.

In maniera analoga a quanto visto per il remeshing, la decimazione del reticolo poligonale avviene mediate l'utilizzo di algoritmi. Inoltre, la riduzione del numero di componenti del reticolo permette di poter adattare la computazione in funzione della macchina di cui si dispone. In altre parole si può scalare la complessità del modello per adattarlo a vari computer aventi capacità di calcolo variabili.

Gli algoritmi che vengono utilizzati per questa attività, in generale si applicano a mesh di tipo manifold e di solito la semplificazione della mesh viene ricavata in maniera iterativa rimuovendo un vertice alla volta. Il processo può avvenire anche in senso inverso: cioè a partire da una mesh decimata si ricostruiscono i dati originali. Quest'ultima applicazione viene impiegata nel caso di una raffinazione del reticolo.

La riduzione di complessità della mesh può essere vista come un'unica operazione nella quale i vertici della mesh decimata fanno parte di un sottoinsieme del reticolo originale. Le tecniche impiegate per questa operazione si classificano come algoritmi:

- **Clustering di vertici**: sono algoritmi molto efficienti e robusti. La complessità computazionale aumenta linearmente all'aumentare del numero dei vertici. L'unico svantaggio è riferito alla qualità finale dell'output che può essere non

soddisfacente, in quanto può accadere che non si ottengano sempre mesh di tipo manifold.

Il processo di decimazione della mesh avviene prima definendo una tolleranza di approssimazione ε e in base a questo valore avviene la suddivisione in celle con dimensione inferiore alla tolleranza. Dopo una fase di calcolo di posizione dei vertici di ogni cella, le facce originali degenerano su angoli che giacciono nella stessa cella e la mesh decimata si ottiene rimuovendo tutte le facce degenerate. Le facce rimanenti corrispondono ai triangoli originali i cui angoli giacciono su celle diverse.

- **Decimazione incrementale**: rispetto al caso precedente porta ad una migliore qualità della mesh. La rimozione dei vertici viene eseguita in maniera iterativa sulla base di criteri definiti dall'utente. Ad ogni rimozione, si è un cambiamento del reticolo nelle vicinanze che richiede un ricalcolo dei criteri di qualità.

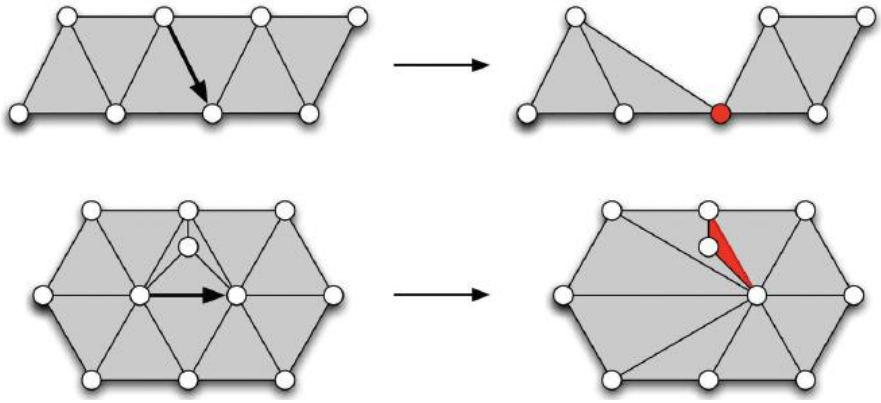


Figura 17. Decimazione incrementale.

- **Ricampionamento**: in questo approccio, gli algoritmi distribuiscono i nuovi punti della nuova mesh sulla superficie. Una volta collegati, quello che si ottiene è una nuova mesh. In questo modo risulta possibile manipolare le connessioni per costruire mesh multirisoluzione. Lo svantaggio di questo metodo risiede nella sua sensibilità alla presenza di aliasing dovuto a un non perfetto allineamento del modello con la geometria originale. Per evitare questo problema si ricorre ad un processo preliminare e manuale di segmentazione dei dati.
- **Approssimazione della mesh**: questi algoritmi oltre a decimare il reticolo, si concentrano sulla riduzione dell'errore di approssimazione attraverso strategie di ottimizzazione della mesh.

2.4.4 Riparazione della Mesh

Dall'analisi generale fatta sulla mesh, a partire dallo smoothing fino alla decimazione, emerge il fatto che la mesh per poter essere manipolata e permettere le successive operazioni di elaborazione, non deve presentare mancanze o discontinuità. In altre parole, deve rappresentare una superficie chiusa e senza intersezioni dei poligoni. Questo aspetto assume un ruolo determinate per quanto riguarda la qualità dell'output, dal momento che gli algoritmi di elaborazione non sono in grado di gestire queste discontinuità. Ciò rende necessario un processo preliminare di riparazione della mesh che può essere eseguito manualmente o tramite un algoritmo di riparazione.

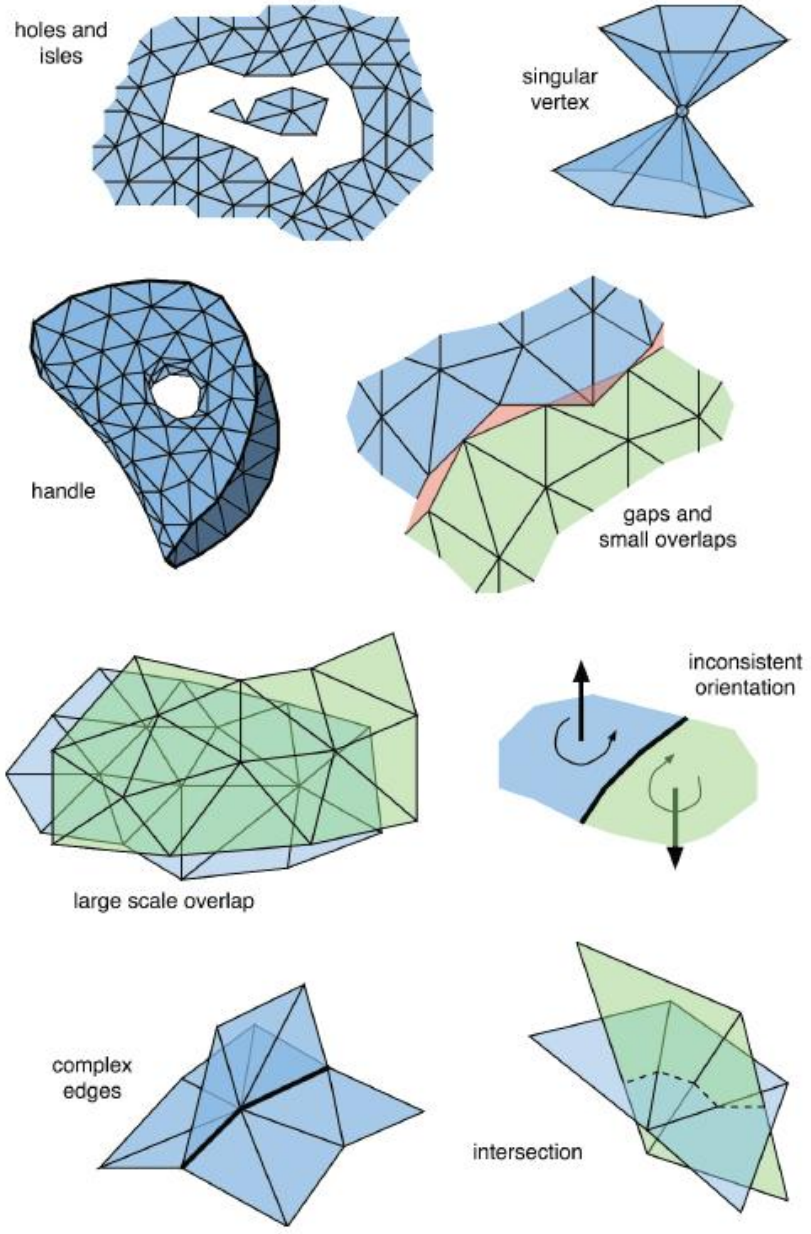


Figura 18. Esempi di discontinuità.

La mesh mancante si riscontra maggiormente nelle scansioni 3D di pezzi fisici ed è principalmente dovuta al fatto che lo scanner non riesce a definire in maniera completa parti del pezzo perché vengono schermate o nascoste da altre superfici. Anche i modelli CAD però presentano delle discontinuità: le più comuni sono le curve autointersecanti che non giacciono sulla loro primitiva geometrica. Si possono inoltre riscontrare intersezioni tra le facce della mesh e anomalie geometriche nei triangoli. Le prime possono essere difficili da rilevare, mentre le seconde possono essere agevolmente rimossi attraverso la decimazione e il remeshing.

Anche in questo caso per la riparazione della mesh poligonale si ricorre all'uso di algoritmi, in quali possono essere classificati come:

- **Orientati alla superficie**: sono algoritmi che cercano di risolvere le discontinuità del reticolo in modo semplice, lavorando direttamente sulla superficie. Gli eventuali spazi vuoti della mesh vengono risolti semplicemente collegando tra di loro gli elementi di contorno formando così nella parte mancante strisce triangolari. Questi algoritmi modificano leggermente il modello di input e sono in grado di conservare le caratteristiche della mesh poligonale proprio perché inseriscono un numero limitato di facce per riparare il reticolo.

  Tuttavia, per essere implementati, il modello di partenza deve soddisfare i requisiti di qualità, ossia la mesh di partenza deve avere un errore di approssimazione noto, rendendo necessario un processamento manuale preliminare. Inoltre, la risoluzione delle discontinuità su vaste aree della mesh risulta difficoltosa e alcune di queste non possono essere rilevate e risolte, come nel caso di due solidi separati ma vicini tra di loro.
- **Orientati al volume**: questi algoritmi trasformano il modello di input in una geometria volumica da cui si estrae il modello di output. La generazione di questo volume implica l'impossibilità di avere discontinuità di vario tipo come intersezioni dei triangoli o sovrapposizioni o spazi vuoti, creando così una mesh chiusa. Questi algoritmi presentano il vantaggio di avere una buona qualità dell'output e non richiedono una fase di manuale di processamento. Allo stesso tempo la costruzione di un volume porta ad una ridefinizione del modello di partenza, potendo perdere caratteristiche e funzionalità dello stesso. Inoltre, l'elevata qualità della mesh ottenuta richiede un più elevato consumo di risorse computazionali e un ulteriore trattamento di decimazione.

3. Reti Neurali

Per il raggiungimento dell'obiettivo di questo elaborato, è necessario manipolare la mesh poligonale tramite l'utilizzo di una rete neurale, che si basa sui concetti di intelligenza artificiale e di machine learning.

L'**intelligenza artificiale** viene definita come il ramo dell'informatica che consente di programmare sistemi hardware e software aventi capacità similari alla mente umana, implementando percezioni visive spazio-temporali e capacità decisionali.

Il **machine learning**, noto come apprendimento automatico, è un sottoinsieme dell'intelligenza artificiale che racchiude metodi che consentono all'algoritmo di risolvere problemi sulla base di regole definite nel codice.

Questi concetti cardine sono alla base del **deep learning**, o apprendimento profondo, è a sua volta un sottoinsieme del machine learning e basa le sue fondamenta sul generare algoritmi che ricreano il comportamento del neurone umano, costruendo così una struttura stratificata. Questo metodo si propone quindi di addestrare i computer per riconoscere ed elaborare dati senza che siano fornite istruzioni precise, aumentando così la flessibilità dell'algoritmo. In altre parole, tutti i sistemi deep learning sono costituiti da reti neurali.

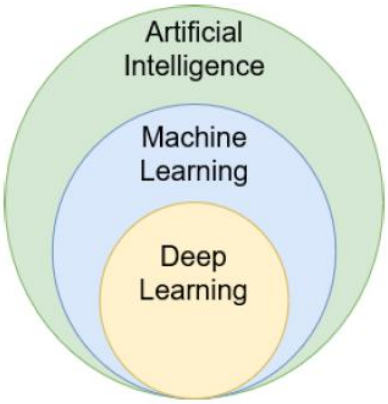


Figura 19. Struttura dell'intelligenza artificiale.

Le reti neurali sono definite come modelli matematici costituiti da neuroni artificiali, i quali replicando il funzionamento del neurone umano, si propongono di migliorare la performance in termini di riconoscimento dati e capacità di azione, attraverso l'apprendimento delle informazioni fornite.

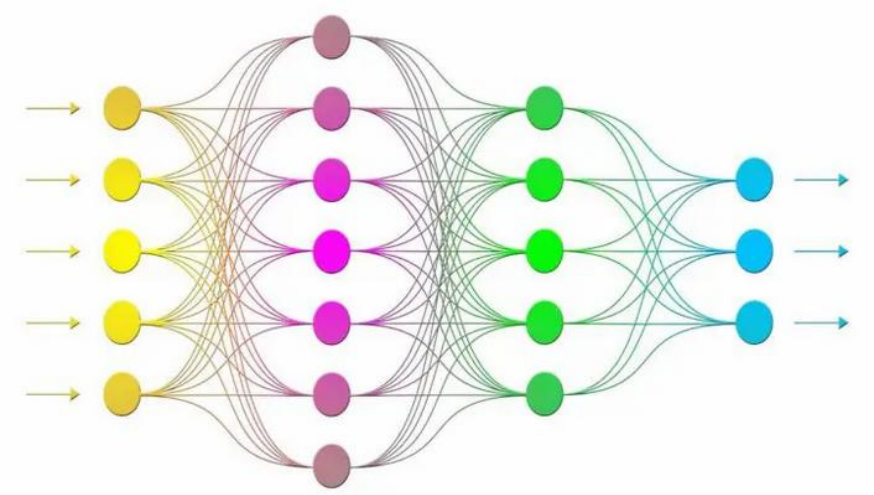


Figura 20. Rete neurale.

Per capire che cos'è una rete neurale e come funziona, occorre prima analizzare il funzionamento di un neurone umano per poi analizzare il neurone artificiale.

Il neurone umano è una cellula nervosa facente parte del sistema nervoso centrale, che viene posta in comunicazione con altre simili al fine di costruire una rete neurale. Questa è in grado di svolgere molteplici funzioni di elaborazione dei segnali. Per poter svolgere questo compito, ogni neurone deve essere dotato di:

- Recettori (dendriti) che gli permettono di acquisire il segnale;
- Nucleo per elaborare il segnale secondo una specifica funzione;
- Assone, un percorso per trasportare il segnale fino al collegamento;
- Sinapsi, il collegamento con il successivo neurone.

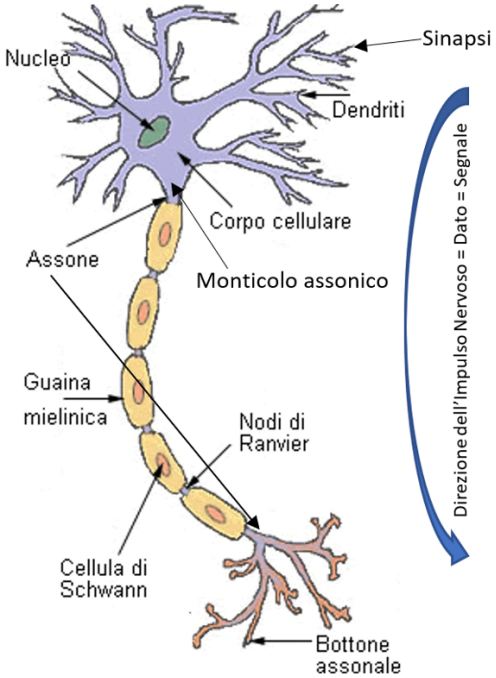


Figura 21. Neurone umano.

Una rete neurale artificiale è costituita da un insieme di neuroni in maniera analoga al cervello umano. Questi neuroni artificiali prendono il nome di nodi, ognuno dei quali possiede un proprio dataset di conoscenza che ha appreso da elaborazioni precedenti, o da regole determinate dal nodo, oppure programmate in fase di costruzione della rete. Questi nodi sono delle unità computazionali che, in maniera analoga alla cellula nervosa umana, ricevono un segnale di input, lo elaborano tramite una funzione di attivazione e producono un output, se il segnale elaborato supera la soglia di attivazione. Ogni neurone artificiale possiede una propria costante di guadagno nota come *bias*.

La struttura di una rete neurale è composta da strati chiamati *layer* e ognuno di questi è composto da un numero ben preciso di neuroni. Il numero di queste unità computazionali fornisce la caratteristica di complessità della rete. In base alla posizione che ciascun layer occupa, si può fare la seguente distinzione:

- **Input layer**: rappresenta il primo livello della rete e prende l'input direttamente dall'ambiente esterno;
- **Hidden layer**: costituiscono i livelli intermedi della rete neurale, dove il layer successivo riceve come input l'output del layer precedente;
- **Output layer**: rappresenta l'ultimo strato della rete.

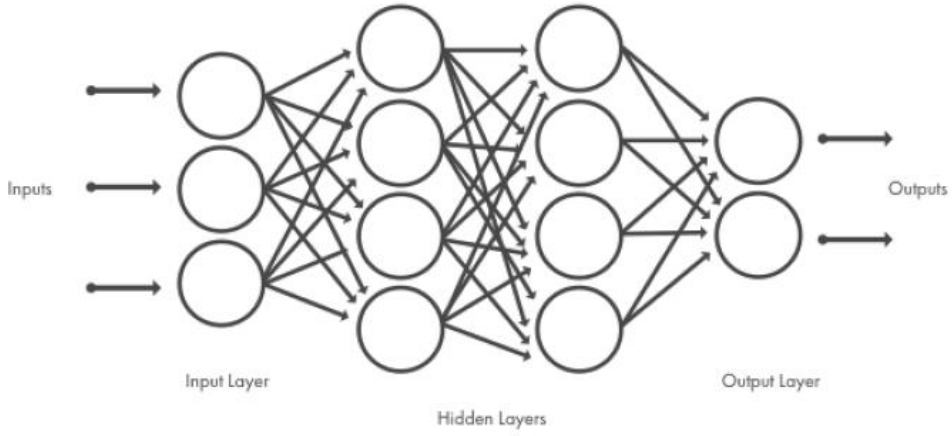


Figura 22. Struttura delle rete in layer.

Occorre evidenziare che i layer sono indipendenti tra di loro e non è presente alcun vincolo sul numero di nodi al loro interno. Tuttavia, come nota pratica, il numero dei neuroni di input deve essere inferiore al numero di nodi degli strati intermedi.

Una volta definiti i vari livelli di cui è composta una rete neurale, si analizzano le connessioni tra i neuroni, le quali assumono una struttura complessiva a grafo, che descrive il modo in cui le informazioni vengono veicolate all'interno della rete. Ogni collegamento tra un due neuroni prende il nome di *edge* o spigolo e ad ognuna di queste connessioni, viene assegnato un peso, *weight*.

Analizzando più nel dettaglio il funzionamento dei neuroni artificiali, il segnale di input che ricevono viene analizzato da una funzione di attivazione, la quale decide se utilizzare o meno l'unità computazionale. In altre parole, la funzione determina se il passaggio per quello specifico neurone sia importante per il processo di elaborazione.

Queste funzioni di attivazione sono funzioni matematiche continue e possono essere di varie tipologie, come ad esempio:

- Funzione **sigmoide** o **logistica**: $f(x) = \frac{1}{1+e^{-x}}$;

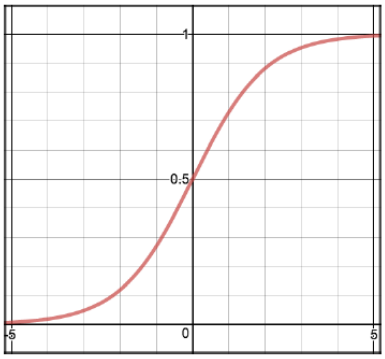


Figura 23. Funzione di attivazione sigmoide.

- Funzione a **tangente iperbolica**: $f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$;

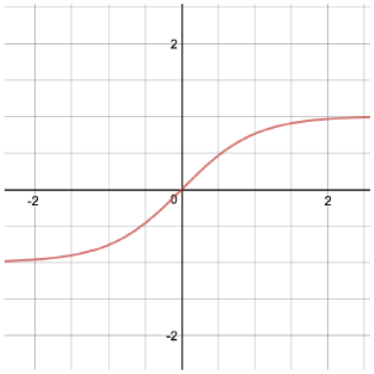


Figura 24. Funzione di attivazione tangente iperbolica.

- Funzione di rettifica lineare: nota come **ReLu**, $f(x) = \max(0, x)$;

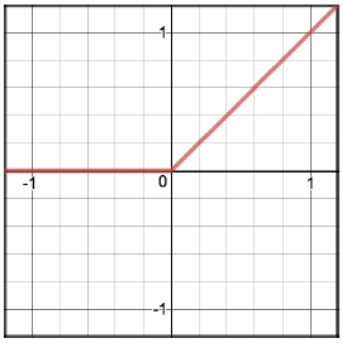


Figura 25. Funzione di attivazione ReLu.

- Funzione **Leacky ReLu**: rispetto alla precedente si introduce un valore a per definire il limite inferiore e la funzione che ne risulta è la seguente:

$$f(x) = \max(ax, x)$$

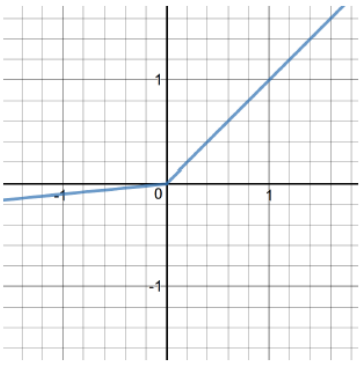


Figura 26. Funzione di attivazione Leaaky ReLu.

3.1 Tipologie di reti neurali

Nota la struttura delle reti neurali artificiali, risulta possibile farne una distinzione in base alla profondità della rete. Questa caratteristica dipende dal numero di hidden layer e dal numero di nodi. Più una rete possiede un numero di livelli intermedi e un alto numero di neuroni, più la profondità della rete aumenta.

Considerando la funzionalità delle reti, in particolare analizzando il movimento del flusso dei dati, si identifica la seguente classificazione:

- **Convolutional neural network (CNN)**: questa tipologia si basa sulle operazioni di convoluzione per produrre l'output. La convoluzione si basa sull'applicazione di filtri (kernel), i quali influiscono sul valore finale prodotto dal neurone. I filtri possono non agire in maniera mirata sui neuroni, i quali se appartengono allo stesso layer e condividono lo stesso filtro possiedono lo stesso valore di peso.

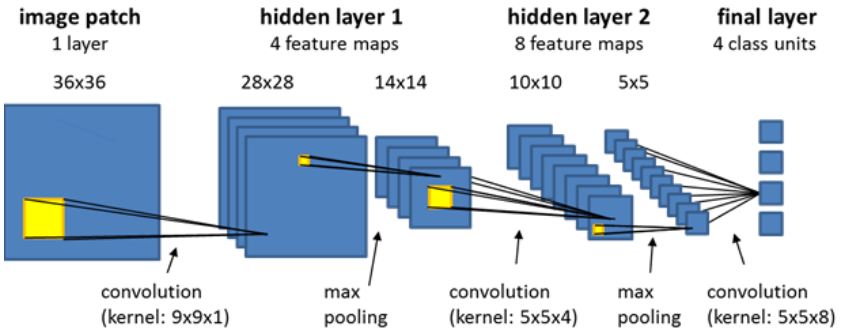


Figura 27. Convolutional neural network.

Inoltre, questa tipo di rete presenta un'architettura identica a quella generale con le tre tipologie di layer (input, hidden e output), con la differenza che gli strati intermedi vengono suddivisi in base alla loro tipologia secondo la seguente classificazione:

- **Convolutional layer**: in questo primo hidden layer sono presenti filtri per le varie operazioni di convoluzione. L'obiettivo di questa tipologia di livello è quello di estrarre le informazioni principali, riducendo così la mole di dati da elaborare. Inoltre, in questa fase la rete CNN impara a costruire in maniera autonoma dei filtri appropriati alla tipologia dei dati;

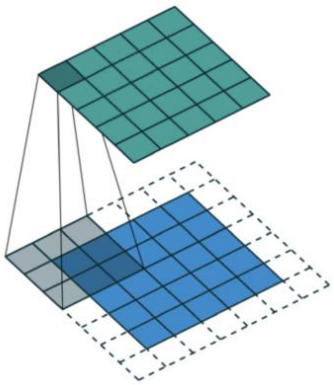


Figura 28. Convolutional layer.

- **ReLu layer**: rappresenta il passo successivo alle operazioni di convoluzione. Come visto in precedenza, ReLu è una funzione di attivazione e viene implementata per il fatto di essere una funzione non lineare e quindi più adatta ai casi reali. In questa fase di elaborazione vengono messi in risalto maggiormente i dettagli;
- **Pooling layer**: una volta evidenziati in maniera opportuna i dettagli, si procede con il ridimensionamento dell'input per ridurre l'onere computazionale. In aggiunta, questo layer ha il compito di immagazzinare le caratteristiche dominanti e invarianti. In base al tipo di pooling che si applica, si può estrarre il valore massimo fornito dalla matrice del filtro, oppure il valore medio;

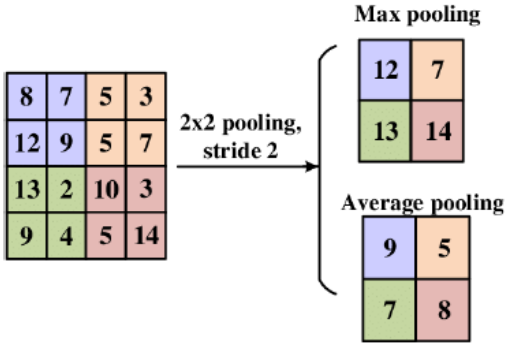


Figura 29. Pooling layer.

- **Fully connetted layer**; effettuata la fase di pooling, il passo successivo prende il nome di flattering, ovvero l'azione di trasformare il dato in uscita dai layer precedenti e dotato di più dimensioni (altezza, larghezza, profondità), in un vettore monodimensionale. In questo modo, al neuronefully connected viene dato in ingresso un singolo valore come input, che risulta quindi "appiattito".
  In questa fase si ha l'apprendimento della rete delle caratteristiche non lineari estratte nelle fasi precedenti. Le reti convoluzionarie presentano un numero variabile di questi layer fully connected fino all'output.

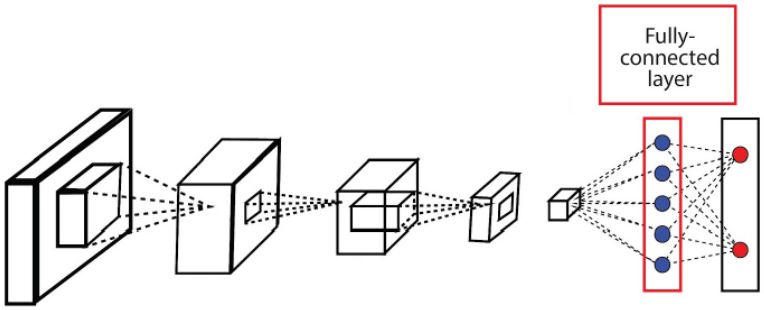


Figura 30. Fully connected layer.

Gli strati intermedi sono quindi costituiti da una combinazione e successione di layer sopra mostrata.

Le CNN trovano la loro applicazione nel classificare immagini ed oggetti e in generale si applicano bene ai casi in cui la struttura dei dati è a griglia, ma possono essere utilizzate anche per previsioni e language processing.

- **Feedforward neural network (FNN)**: in questa categoria il flusso dei dati all'interno della rete è unidirezionale: a partire dal layer di input, attraverso gli strati intermedi raggiunge quello di output. Questa architettura non presenta quindi loop e sono le reti di più semplice implementazione.
  Di questa categoria fanno parte le **Multilayer Perceptrons** (MLP) che trovano la loro applicazione nel riconoscimento di oggetti all'interno delle immagini. Le MLP sono reti aventi come struttura almeno tre layer: uno di input, uno intermedio ed uno di output. Le reti FNN si compongono di layer multipli di MLP e risultano adatti per classificazioni di ampi dataset.

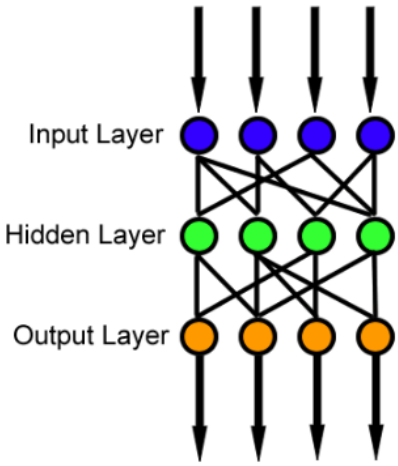


Figura 31. Feedforward neural network.

- **Recurrent neural network (RNN)**: a differenza delle precedenti, il flusso dei dati non è unidirezionale, ma presenta fasi di elaborazione nelle quali i dati ottenuti da strati precedenti vengono utilizzati per modificare le elaborazioni dei layer successivi. Inoltre, sono presenti dei loop nelle connessioni tra i vari neuroni.

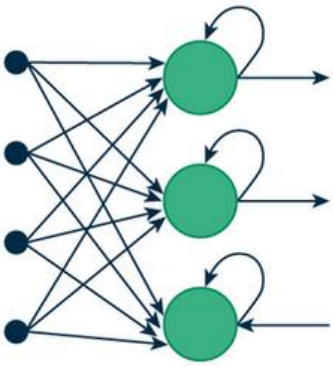


Figura 32. Recurrent neural network.

- **Fully Connetted**: sono note come reti neurali completamente connesse. Questa architettura si costruisce connettendo ogni nodo di uno strato con tutti i neuroni dello strato successivo. In questo modo si ottiene una struttura della rete più flessibile in termini di dati input, permettendo così una più ampia applicazione. A livello di prestazioni, queste si rivelano però inferiori rispetto a reti progettate appositamente per la specifica funzione.

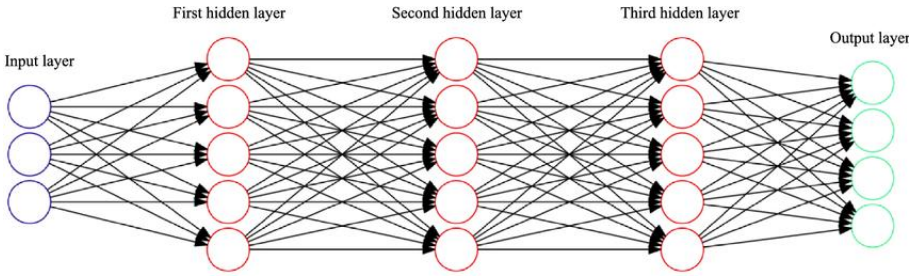


Figura 33. Fully connected neural network.

- **Reti neurali autoencoder**: queste reti vengono utilizzate per l'elaborazione dei dati e prendono anche il nome di codificatori, agendo in modo da evidenziare i dati importanti e ridurre l'importanza delle informazioni non rilevanti. Il layer di output svolge la funzione di decodifica e all'aumentare della profondità della rete si aggiungono più strati verso il layer di decodifica.

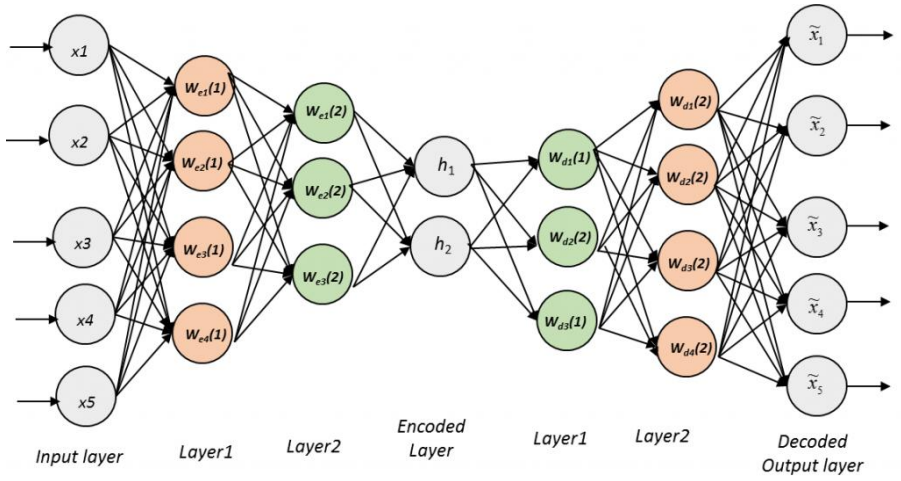


Figura 34. Autoencoder neural network.

- **Graph Convolutional Network (GCN)**: si tratta di un tipo di rete neurale che, contrariamente alle CNN che utilizzano dati strutturati a griglia, come immagini, richiede in ingresso dati strutturati come grafi, come ad esempio una mesh poligonale.

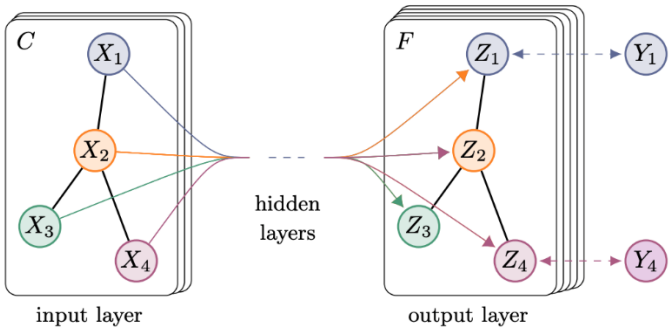


Figura 35. Graph convolutional network.

3.2 Metodi di apprendimento della rete

Riveste un ruolo importante la fase di apprendimento o di training della rete. Le reti neurali, infatti, sono composte da neuroni artificiali, i quali nel corso del processamento dei dati cercano di produrre valori di output sempre migliori.

Per questo processo delicato si utilizzano degli algoritmi di apprendimento, che sulla base della logica di funzionamento possono essere distinti in:

- **Approccio supervisionato**: con questo metodo, vengono forniti alla rete dei dati di input a cui corrispondono output noti. L'algoritmo permette alla rete di capire la connessione tra input e output. In seguito, la rete oltre a generalizzare le connessioni, ne costruisce di nuove potendo così acquisire dati che non sono presenti nel set di input iniziale.
  L'apprendimento della rete avviene durante il processo di elaborazione dati attraverso la continua modifica dei pesi. In altre parole, ad ogni collegamento tra i neuroni viene assegnato un peso iniziale, durante il processamento dei dati l'algoritmo varia questi pesi in modo che la produzione dell'output venga direzionata in modo da aumentare la qualità del risultato finale. I pesi vengono incrementati nel caso l'output sia corretto e diminuiscono di valore se la risposta non fornisce valori veritieri.
  L'efficienza dell'apprendimento dipende dai parametri in ingresso. Si riscontra che aumentandoli si ha una crescente difficoltà di apprendimento da parte della rete, in quanto risulta sempre più complessa la generalizzazione. Questo problema si ripropone anche nel caso di un set di dati con scarso numero di parametri, poiché lo scarso numero di variabili non permette di generalizzare.
  Questo modello trova applicazione nelle reti feedforward.
- **Approccio non supervisionato**: a differenza del modello precedente, riceve in ingresso delle variabili di input, la rete le analizza e impara a riconoscere i vari modelli e a trovare le relazioni tra i dati, il tutto in maniera completamente autonoma. I pesi delle connessioni variano in maniera molto dinamica e la loro modifica è assegnata completamente ai neuroni. In questo caso l'ottimizzazione dell'output non tiene conto dell'errore. Questo approccio consente di non catalogare i dati di input.
- **Approccio supervisionato con rinforzo**: l'apprendimento della rete avviene per mezzo della sua interazione con l'ambiente esterno. Il rinforzo è rappresentato dalle azioni effettivamente utili a raggiungere il risultato finale che viene predeterminato. La rete risulta quindi in grado di determinare quali azioni siano utili o positive e quali inutili o negative, le quali potrebbero portare ad errori e a un aumento tempo di elaborazione. Le azioni reputate negative possono essere eliminate dall'algoritmo, aumentando così la qualità dell'output.
- **Approccio semi-supervisionato**: si tratta di un approccio ibrido tra quello supervisionato e non supervisionato. Per l'apprendimento si utilizza un set di dati etichettati e una grande quantità di dati non classificati. Combinando i due approcci si riesce a utilizzare le potenzialità di entrambi i metodi mirando ad ottimizzare le prestazioni della rete. Questo metodo trova applicazione nel caso in cui si ha un enorme set di dati con etichettatura incompleta e alta complessità.

3.3 Training della rete

Questo capitolo si concentra sull'analisi di addestramento della rete neurale. La fase di training della rete riveste un ruolo chiave in quanto impatta in maniera rilevante sulla qualità del risultato finale. Come evidenziato in precedenza la fase di training dall'andamento della variazione del valore dei pesi, dove questi rappresentano le connessioni tra i nodi.

La variazione dei valori dei pesi durane l'addestramento della rete, viene determinata dalla funzione di errore che prende il nome di *loss*. Questo parametro rappresenta l'indice della performance generare della rete, in quanto più è contenuto e più l'output tende al valore ideale. In letteratura la funzione di loss L, prende anche il nome di funzione di costo e può essere definita come:

$$L = L(D, \omega)$$

Dove:

- D indica l'insieme dei dati, ovvero rappresenta il dataset;
- ω indica i pesi;

Come affermato in precedenza, la funzione di loss fornisce un valore che indica la differenza tra il valore di output elaborato dalla rete e il valore di riferimento. Da questo risulta chiaro che l'errore vada ridotto al minimo possibile.

In funzione della tipologia e della classificazione del dataset esistono in letteratura diverse funzioni di loss a cui occorre poi ridurre al minino questo valore.

Per minimizzare la funzione di costo, il metodo più comunemente utilizzato è quello della **discesa del gradiente** che consiste nel calcolo di un valore minimo locale di una funzione differenziale. Inizialmente i valori dei pesi ω sono assegnati in maniera casuale. In seguito, si procede con il calcolo del gradiente della funzione di loss e i pesi vengono poi aggiornati secondo la direzione decrescente del gradiente della funzione di costo secondo la legge:

$$\omega_{l+1} = \omega_l - r\, \nabla_\omega\, L$$

Dove i parametri sono definiti come segue:

- ω_l : rappresenta il valore del peso precedente;
- ω_{l+1} : indica il valore successivo che assumerà il peso;
- r : viene definito come **learning rate**, questo parametro indica di quanto muoversi nella direzione indicata dal gradiente. Riveste un ruolo fondamentale nell'ottimizzazione della rete;
- $\nabla_\omega\, [L(\omega)]$: indica il gradiente della funzione di costo;

Durante il processo, il valore dei pesi viene aggiornato sulla base dell'errore fino a quando non si raggiunge il valore minimo della funzione di loss. L'aggiornamento dei pesi può avvenire secondo due tipi di approcci:

1. **Batch**: nel quale le modifiche dei pesi vengono apportate solo una volta noto il dataset. In altre parole, prima di modificare i pesi si valuta l'errore sull'intero insieme di modelli. In questo modo si apportano poche ma significative modifiche;
2. **On-line**: le modifiche avvengono dopo la presentazione di ogni modello. Questo metodo presenta un numero maggiore di modifiche ma di entità contenuta.
   Nella pratica quest'ultimo metodo è quello che si preferisce applicare, in quanto fissato un learning rate opportuno, consente una migliore conoscenza dell'andamento della funzione di costo.

Infine, occorre definire la grandezza temporale in cui avviene il processo. Si introduce quindi il parametro che prende il nome di **epoca**, la quale rappresenta l'unità di misura del tempo di addestramento. Al termine di ogni epoca la rete neurale produce un valore di loss che può essere graficato. Il numero di epoche può essere scelto manualmente da codice.

L'ottimizzazione di questi parametri è orientata a ridurre il loss e quando la rete ha determinato il valore minimo si dice che è stata raggiunta la **convergenza**. Per raggiungere questo obiettivo si possono fare delle considerazioni sul learning rate. Come definito in precedenza, il learning rate rappresenta la variazione della correzione eseguita sul peso. Il suo valore riveste quindi un ruolo rilevante in termini di prestazione della rete, in quanto determina la velocità con cui la rete converge. Rispetto al valore di default iniziale si possono distinguere due casistiche opposte:

1. Con un bassissimo valore di learning rate, la convergenza della rete impegna un tempistiche rilevanti;
2. Con un valore molto elevato del tasso di apprendimento, può risultare difficoltoso raggiungere valori minimi del loss e ciò potrebbe portare ad un andamento oscillatorio.

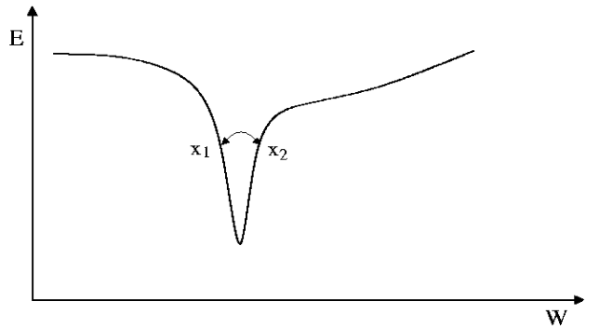


Figura 36. Andamento del learning rate in funzione dell'errore (E) e del peso (W).

La durata dell'addestramento dipende direttamente anche dal numero di epoche che si è in precedenza settato. Anche questo parametro influenza il valore finale della funzione di costo, per cui occorre trovare anche in questo caso il numero ottimo di epoche. Si riscontra infatti che aumentando il numero di epoche il valore della funzione di costo assume un andamento decrescente.

Durante la fase di training si possono riscontrare diversi problemi che peggiorano le prestazioni della rete, spesso legati all'adattabilità e alla generalizzazione dell'algoritmo sui dati:

1. **Over-training**: aumentare il numero di epoche può non portare benefici, in quanto dopo un certo valore di soglia la loss tende ad aumentare. Questo si verifica a causa dell'overtraining, dove la rete perde la capacità di generalizzare i dati e risulta necessario interrompere l'addestramento.

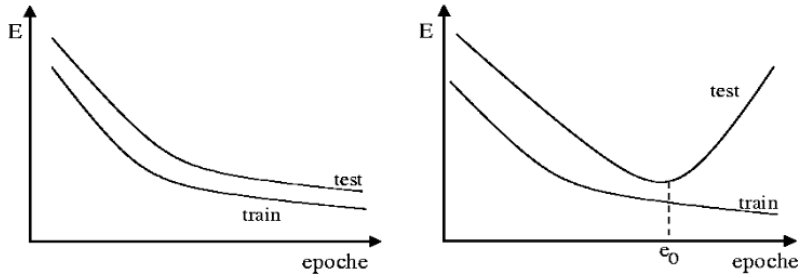


Figura 37. Andamento corretto (sinistra) e over-training (destra).

   Facendo riferimento alla figura precedente, è possibile notare come nel grafico a sinistra, la rete riesca a generalizzare correttamente, in base al valore delle epoche e dell'errore (E), così da riuscire a minimizzare quest'ultimo. Nel grafico a destra, invece, dopo un certo numero di epoche (e_0), l'errore sul dataset di test comincia a crescere e si verifica una situazione di over-training, durante la quale la rete non è più in grado di generalizzare ed è quindi necessario interrompere l'addestramento.
2. **Overfitting**: questo problema si presenta quando la rete si adatta troppo ai dati iniziali, ottenendo un buon valore di loss; tuttavia, fornendo all'algoritmo un nuovo set di dati, la rete fallisce fornendo valori di output incoerenti. In altre parole, è come se la rete imparasse a memoria tutti i dati senza ricavare delle connessioni tra di essi. Questa problematica emerge soprattutto quando i dati in ingresso sono complessi, cioè presentano molti dettagli da elaborare;
3. **Underfitting**: rappresenta il problema contrario all'overfitting, cioè la rete incontra difficoltà ad imparare a causa dei dati iniziali, i quali essendo troppo semplici con pochi dettagli non risultano rappresentativi della complessità dei dati e non permettono quindi una generalizzazione degli input. L'underfiting può emergere anche quando si ha un dataset insufficiente e i dati forniti non influenzano la variabile di target, oppure quando le caratteristiche delle

informazioni fornite non risultano dimensionate correttamente. L'underfiting si può presentare anche come conseguenza dell'overfitting, in quanto per evitare quest'ultimo si potrebbero ridurre eccessivamente i dettagli del modello di input.

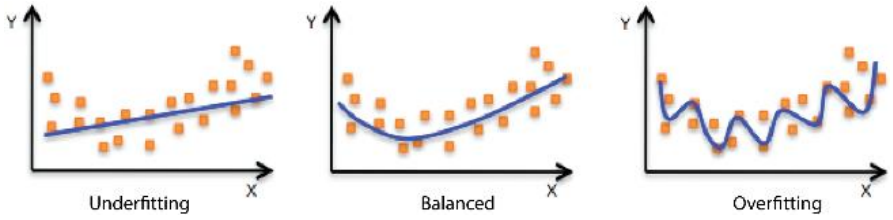


Figura 38. Underfitting e Overfitting

3.4 Stato dell'arte

Dopo una panoramica generale sulla mesh poligonale e sulle reti neurali, è necessario procedere con la ricerca di una rete neurale esistente e preallenata ottimale per elaborare modelli CAD in formato .stl e .obj. Inoltre, dato che l'ambiente informatico è in continua evoluzione, occorre filtrare la ricerca alle sole reti recenti che abbiano non più di due anni dalla compilazione di questo elaborato.

I requisiti per l'individuazione della rete più idonea per lo scopo di questo elaborato si possono riassumere come segue:

1. La rete neurale deve essere già esistente, completa di codice e di facile installazione;
2. La rete deve essere già preallenata, in quanto il training risulta essere una fase delicata e dispendiosa in termini di risorse computazionali e di tempo;
3. Si selezionano reti neurali la cui ultima modifica non deve essere anteriore al 2022;
4. L'algoritmo deve ricevere in ingresso file CAD in formato .stl oppure .obj;
5. La rete deve poter gestire ed effettuare operazioni di modifica sulla mesh poligonale.

Data la vastità di reti presenti in letteratura, la ricerca è stata prevalentemente svolta su GitHub. Si tratta di una libreria opensource al cui interno oltre ad essere caricati dai vari utenti codici completi e liberi, oppure con licenza vincolata, sono presenti anche articoli che descrivono una panoramica della rete in questione.

Di seguito vengono riportate le principali reti neurali candidate per poter raggiungere lo scopo di questo elaborato:

- **SeMIGCN**: questa rete neurale ha come ultimo aggiornamento giugno del 2024, così da soddisfare il terzo requisito, inoltre il codice fornito possiede una licenza MIT. Ciò significa che è possibile apportare modifiche ai vari script presenti nel

sito. Inoltre, risulta di facile installazione. Questa rete neurale progettata in linguaggio Python necessita di Linux come sistema operativo per poter funzionare. SeMIGCN riceve in ingresso file CAD in formato .obj e necessita esclusivamente di un'unica mesh incompleta di input per effettuare la fase di training. Il risultato finale risulta essere una nuova mesh poligonale completa e impermeabile della stessa topologia di quella iniziale. Questa rete neurale è di tipo CNN.

- **Polygen**: questo modello data l'ultimo aggiornamento al 2022. Si tratta di un'implementazione Pytorch alla rete Polygen. La rete neurale si propone di generare una mesh 3D a partire da vari dataset. A differenza della rete precedente, vengono generati dei modelli 3D a partire da immagini, quindi la nuova mesh che si ricava dipende dall'immagine di input. Nonostante il modello caricato nel sito riceva in ingresso delle immagini, si nota che sono forniti nel dataset anche file di tipo .obj. Tuttavia, il codice non risulta disponibile con licenza MIT.
- **Neural Mesh simplification**: la rete neurale si propone di semplificare una mesh poligonale 3D, mantenendone la topologia ma senza crearne una nuova. Il procedimento avviene a partire da un insieme di vertici della mesh di input, si effettua il ricampionamento casuale e si procede con l'addestramento della rete per produrre nuovi triangoli in base alla connettività dei bordi campionanti. Si conclude con una classificazione dei triangoli per capire quali includere nel reticolo finale. Questa rete è stata aggiornata di recente, il codice è scaricabile dal sito e riceve in ingresso file di formato .obj.
- **MeshGPT**: è rete neurale aggiornata di recente, il codice è disponibile sul sito con licenza MIT. Tuttavia, non è utilizzabile per via della licenza presente. Questo modello si propone di generare mesh triangolari adottando un approccio basato sul generare mesh triangolare come sequenza di triangoli. Per fare ciò si parte da un dataset iniziale da cui apprendere le varie caratteristiche della geometria e la topologia della mesh. Queste proprietà vengono elaborate dall'algoritmo per poi venire decodificate in triangoli al fine di ricostruire correttamente la mesh. Quello che si ottiene in output è quindi una nuova mesh triangolare compatta che approssima al meglio la geometria iniziale. Lo svantaggio di questa rete risiede nel suo allenamento che risulta complesso da implementare e dispendioso in termini di tempo.
- **Mesh Anything**: questa rete neurale necessita di un modello 3D su cui viene costruita e fatta aderire una nuova mesh. A partire da una nuvola di punti, questi vengono campionati, codificati attraverso un decoder e in seguito avviene la generazione della nuova mesh poligonale. Il modello genera la mesh secondo una topologia ottimizzata della forma, in modo da ridurre l'onere computazionale. Tuttavia, la rete neurale presenta delle limitazioni, come ad

esempio il limite massimo di facce pari a 800 – sia per quanto riguarda il modello di input sia per la nuova mesh poligonale – inoltre la mesh di input deve risultare nitida con i bordi ben definiti. La variante Mesh Anything V2 porta il limite da 800 a 1600 facce. La rete riceve in input file di tipo .obj, risulta aggiornata di recente. La licenza non è di tipo MIT però il codice è scaricabile.

3.4.1 Scelta della rete neurale

Dopo aver fornito una panoramica generale delle reti neurali candidate per svolgere questo elaborato, si può concludere che la scelta migliore ricade sulla rete SeMIGCN.

La rete neurale selezionata risulta quella che meglio soddisfa i requisiti necessari, in quanto rispetto alle alternative proposte risulta di più facile installazione, avendo una procedura guidata semplice che richiede l'utilizzo di Docker sul sistema operativo Linux.

Docker è una piattaforma opensource che permette di creare, eseguire e distribuire applicazioni in ambienti isolati che prendono il nome di container. Questo sistema permette di evitare di installare e utilizzare una macchina virtuale che presenta un peso computazionale maggiore rispetto ai singoli container. All'interno di questi container sono contenuti tutti file necessari per farlo funzionare, questo permette di poter utilizzare i vari container su qualsiasi computer che abbia Docker installato. Inoltre, avendo creato un sistema isolato, risulta possibile eseguire i vari contenitori in parallelo senza che questi entrino in conflitto tra di loro.

Inoltre, l'utilizzo della rete SeMIGCN richiede pochi intuitivi comandi che ne consentono una facile compressione del processo. In particolare i parametri della rete sono evidenziati all'interno degli script secondo un elenco ben leggibile, permettendo così di modificarli rapidamente.

Un ulteriore elemento che ha portato alla scelta della rete neurale SeMIGCN è rappresentato dalla sua facilità d'uso, in quanto per la sua logica di funzionamento e architettura non richiede un enorme dataset di dati, ma riceve in ingresso una sola mesh poligonale danneggiata, restituendole in output una nuova mesh rielaborata.

4. SeMIGCN

Data l'importanza della rete neurale per questo elaborato, risulta necessario farne un approfondimento. Come affermato in precedenza, la rete neurale riceve in ingresso una sola mesh incompleta, non richiedendo così nessun dataset per il training. La rete si concentra sul processo di inpainting – ovvero il riempimento delle mancanze della mesh incompleta fornita in ingresso – basato sulla definizione di auto-priorità.

La particolarità della rete SeMIGCN risiede nel fatto che durante l'elaborazione, viene mantenuta la struttura della mesh poligonale senza trasformazione del reticolo in formati intermedi. Per formati intermedi si intende la trasformazione della mesh in una nuvola di punti, una griglia voxel o funzioni implicite. Queste rappresentazioni di dati mesh sono tenuti come riferimento, in quanto sono facilmente gestibili da reti neurali profonde, cioè con molti hidden layer.

A livello di strumentazione, la rete SeMIGCN utilizza le seguenti specifiche:

- La rete è stata costruita utilizzando i programmi Pytorch e Pytorch geometric;
- Il computer utilizzato è dotato di una CPU intel Core i7 5930K(3.5 GHz, 6 core), GPU NVIDIA GeForce TITAN X avente 12 Gb;
- Il training è stato settato a 100 epoche e con un learning rate iniziale pari 0.01, inoltre nell'algoritmo è stato implementato il dimezzamento del learning rate dopo 50 epoche;

Dai test condotti dagli autori sulle diverse geometrie di varia complessità, si è riscontrato che la rete riesce bene a riparare piccoli fori nella mesh, ma all'aumentare della dimensione di questi, la qualità peggiora.

Nei casi di le scansioni 3D dei vari test, alcune caratteristiche geometriche come protuberanze o dettagli complessi possono perdersi e la rete SeMIGCN non risulta in grado di colmare geometrie totalmente mancanti.

L'utilizzo della rete neurale SeMIGCN presenta quindi le seguenti limitazioni:

- La mesh prevista di output dopo l'elaborazione della rete deve essere chiusa e impermeabile. Questa problematica è dovuta all'uso di MeshFix, che considera le mesh a dimensione maggiore a discapito delle piccole entità nel caso di reticoli separati, perdendo così delle caratteristiche sulla geometria;
- La difficoltà di recupero delle forme caratteristiche nelle regioni che rimangono nascoste. La rete infatti non risulta in grado di generare una geometria mancante, in quanto si ha solo una mesh incompleta come input alla rete.

La rete SeMIGCN è strutturata in tre fasi:

1. Preprocess;
2. Training;
3. Evaluation.

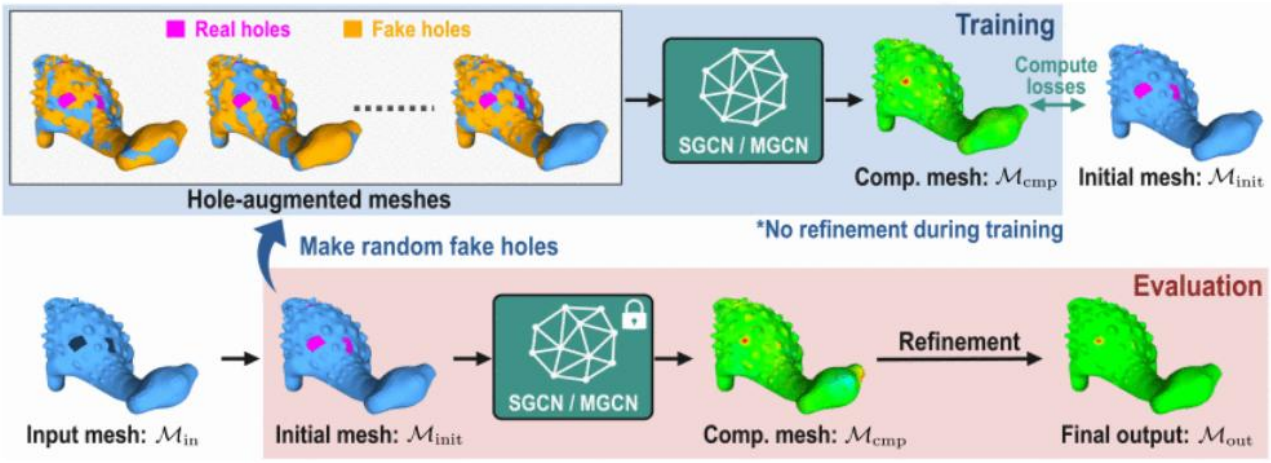


Figura 39. Rete SeMIGCN.

4.1 Preprocess

Per rendere idonea la mesh incompleta in input per le successive fasi di training, è necessaria una preelaborazione (*preprocess*) perché possa essere data alla rete neurale. Questa esigenza deriva dalla difficoltà di generazione di nuove facce, spigoli e vertici. Per ovviare al problema si utilizza MeshFix, ovvero un software implementato su Python, che riceve in ingresso una mesh poligonale e genera una copia del reticolo di input correggendone i difetti. MeshFix viene fornito dagli autori della rete SeMIGCN all'interno del file prepare.py. MeshFix ripara le mancanze della mesh inserendovi nuovi vertici, ottenendo così un riempimento iniziale dei fori.

Segue poi con una fase di *remeshing* per uniformare il campionamento dei vertici su tutta la superficie. Vi è la *normalizzazione della scala* e infine si completa con uno *smoothing* laplaciano uniforme per estrarre le caratteristiche geometriche locali. Questa preelaborazione della mesh iniziale permette di avere un miglioramento della qualità dell'output rendendo inoltre indipendente il risultato ottenuto dal riempimento operato da MeshFix.

4.2 Training

La gestione delle operazioni sulla mesh da parte della rete SeMIGCN avviene mediante l'introduzione di due reti convoluzionarie a grafi di tipo GCN: la prima avente risoluzione multipla nota come MGCN, e la seconda a risoluzione singola SGCN.

Per quanto riguarda la fase di training delle due reti neurali sovracitate, per utilizzare una sola mesh di input, l'addestramento avviene secondo il metodo autosupervisionato. Il risultato finale consiste quindi nell'ottenere una mesh di output completa a partire da una mesh in ingresso impermeabile, dove le varie mancanze del reticolo sono state chiuse. La mesh poligonale che la rete neurale riceve come input deve essere di tipo triangolare, in quanto risulta essere la struttura del reticolo più utilizzata, specialmente nel campo delle scansioni 3D. Prendendo in esame proprio questo ambito, per trasformare un pezzo fisico in un modello digitale, si utilizza uno scanner 3D, in cui il risultato finale che si ottiene risulta essere una mesh poligonale incompleta del componente, perché può accadere che per la disposizione e colorazione delle superfici, alcune aree non vengono scansionate correttamente. Il reticolo poligonale incompleto risulta così inutilizzabile per eventuali operazioni successive di analisi.

L'obiettivo della rete SeMIGCN è quello di ricostruire in maniera fedele e naturale le parti del reticolo mancanti per ottenere un modello impermeabile, al fine di fornire un input di alta qualità per le elaborazioni future. In altre parole, si cerca di aggiungere alla mesh iniziale i triangoli mancanti attraverso l'utilizzo di una rete neurale.

La rete scelta per questo elaborato riceve in ingresso una sola mesh di input, evitando di dover effettuare l'addestramento con un ampio dataset, risparmiando così tempi e risorse. Per fornire questa caratteristica alla rete, gli ideatori hanno proposto un metodo di apprendimento "self prior" o auto-priorità, che consiste nell'implementare una conoscenza pregressa che la rete acquisisce dal singolo input incompleto. Questo sistema ha fornito buoni risultati nella ricostruzione di immagini. Per questo motivo gli autori lo hanno riadattato anche alla ricostruzione di geometrie 3D.

L'apprendimento **self prior** durante l'inpaintig della mesh mantiene il formato e la topologia iniziale del reticolo poligonale, in questo modo si evita il problema di convertire la mesh in altri formati, come per esempio una nuvola di punti. Si evita così di ottenere una geometria di output inappropriata, causata da possibili distorsioni del reticolo durante il processo di elaborazione. L'addestramento utilizza quindi solo i dati di input, il che permette di applicare due reti di tipo convoluzionarie GCN con struttura a grafo:

- **SGCN**: presenta una struttura di strati di convoluzione dove i grafi sono impilati in maniera rettilinea sui grafi della mesh in ingresso. La rete si compone di 13 blocchi di convoluzione e da un layer complementare connesso;
- **MGCN**: differisce dalla struttura precedente per la presenza di un sistema codificatori-decodificatori con strati di pooling e unpooling, i quali sono definiti su una mesh utilizzando una serie precalcolata di collassi di spigoli. Queste due operazioni riducono l'onere computazionale nel caso di mesh poligonale di

grandi dimensioni con centinaia o migliaia di vertici. Anche in questo caso l'ultimo layer è di tipo fully connected.

L'addestramento della rete neurale avviene quindi attraverso l'aumento dei dati sulla mesh di input, evidenziando alcune zone della mesh scelte casualmente come buchi falsi e le posizioni corrette dei vertici sono note dalla supervisione durante l'addestramento. Utilizzando l'addestramento self prior viene risolto il problema del riempimento dei buchi nella mesh spostando i vertici di un reticolo levigato ottenuto tramite il processo di smoothing, nelle loro posizioni corrette. Questo significa che il processo dell'inpainting avviene definendo dei vettori di spostamento dei vertici anziché delle loro posizioni.

Durante la fase di training e dopo avere mappato alcune zone continue come buchi falsi, si introduce un vettore a quattro dimensioni, dove le prime tre indicano lo spostamento del vertice nelle tre dimensioni. Il quarto elemento invece fornisce informazioni di tipo binario sulla presenza o meno di un buco nel reticolo, assegnando il valore 0 nel caso vi sia una mancanza, mentre nel caso in cui non sia presente nessun buco si inserisce 1. Se si ricade nell'ultima ipotesi, anche gli altri valori sono nulli. In questa maniera si trattano in modo identico i buchi falsi e quelli veri. La rete neurale GCN si addestra così a prevedere i vettori di spostamento nelle aree dove sono presenti dei buchi e tutto questo permette di conoscere lo spostamento corretto dei vertici nei buchi falsi. La mesh di output si ottiene applicando gli spostamenti corretti al reticolo ottenuto tramite lo smoothing.

Per valutare la qualità del risultato finale della rete, si utilizza la funzione di perdita o **Loss function**. Questa funzione indica la differenza tra l'output della rete neurale e il modello ideale. Per ottenere una qualità migliore la funzione di loss deve fornire il risultato minimo possibile. La rete SeMIGCN utilizza tre tipi di funzione di loss:

- Termine dati per le **posizioni dei vertici**: l'errore di posizione X_{cmp} viene calcolato tramite l'errore quadratico medio tra le posizioni dei vertici della mesh originale e le posizioni dei vertici che hanno riempito i buchi. Per definire il termine dati, si introduce una maschera binaria $\tilde{m}^i_{vtx} \in \{0,1\}$ dove il valore 0 corrisponde a buchi reali, mentre 1 si riferisce a buchi falsi. La funzione di loss si calcola come segue:

$$E_{\text{pos}}(\mathbf{X}_{\text{cmp}}, \mathbf{X}_{\text{init}}) = \left(\frac{1}{\sum_i \tilde{m}^i_{\text{vtx}}} \sum_{i=1}^{|\mathcal{V}|} \tilde{m}^i_{\text{vtx}} \| \mathbf{x}^i_{\text{cmp}} - \mathbf{x}^i_{\text{init}} \|_2^2 \right)^{\frac{1}{2}}$$

  Dove $\|.\|$ indica la norma del vettore.
- Termine dati per le **normali alle facce**: le direzioni normali definiscono le caratteristiche del reticolo poligonale, quali per esempio gli spigoli vivi. Risulta quindi possibile definire le posizioni dei vertici tramite le componenti normali

alle facce. Per garantire che le normali tra la mesh in ingresso e quella di output siano corrisposte, si introduce il temine dati per le normali alle facce, definito tramite l'errore medio assoluto. Anche in questo caso si inserisce una maschera binaria $\tilde{m}_{nrm}^{j} \in \{0,1\}$ dove il valore zero indica facce con fori reali, mentre si assegna il valore uno alle facce con fori falsi. La funzione di loss si calcola come segue:

$$E_{\text{nrm}}(\mathbf{N}_{\text{cmp}}, \mathbf{N}_{\text{init}}) = \frac{1}{\sum_j \tilde{m}_{\text{nrm}}^j} \sum_{j=1}^{|\mathcal{F}|} \tilde{m}_{\text{nrm}}^j \|\mathbf{n}_{\text{cmp}}^j - \mathbf{n}_{\text{init}}^j\|_1$$

Il valore E_{nrm} permette quindi di riprodurre le caratteristiche della mesh poligonale di input.

- Termine di **smoothing** per le normali alle facce: si introduce un termine di regolarizzazione come termine di smoothing per le normali di sfaccettatura che avviene mediante un filtraggio normale bilaterale (BNF). In questo modo si riesce a mantenere la geometria inalterata. Il regolarizzatore dello smoothing per le normali di sfaccettatura viene definito dalla seguente funzione:

$$E_{\text{reg}}(\mathbf{N}_{\text{cmp}}) = \frac{1}{|\mathcal{F}|} \sum_{j=1}^{|\mathcal{F}|} \|\mathbf{n}_{\text{cmp}}^j - S_{\text{bnf}}^{(t)}(\mathbf{n}_{\text{cmp}}^j)\|_1$$

Dove la funzione di smoothing $S_{bnf}^{(t)}$ utilizza il filtraggio BNF. Il valore E_{reg} viene calcolato solo per mesh a risoluzione più fine per ridurre l'onere computazionale.

Definite le tre funzioni di perdita, si procede con il calcolo della **funzione di loss globale** attraverso cui si effettua il training della rete:

$$E = \sum_{r=0}^{R} \lambda_{\text{pos}}^{(r)} E_{\text{pos}}^{(r)} + \lambda_{\text{nrm}} E_{\text{nrm}} + \lambda_{\text{reg}} E_{\text{reg}}$$

Il termine R nella sommatoria indica il numero di risoluzioni e utilizzando le due reti convoluzionarie a disposizione, assume i seguenti valori:

- Utilizzando SGCN, avendo risoluzione singola abbiamo $R = 1$;
- Utilizzando MGCN, avendo risoluzione multipla, abbiamo $R = 3$.

I parametri λ indicano dei pesi che vengono poi modulati per ottenere la corretta influenza del proprio valore di loss sul risultato globale di loss.

4.3 Evalutation

Definita la funzione di loss globale si pone ora il problema della ottimizzazione dei parametri della rete. Il processo di ottimizzazione richiede tempi abbastanza lunghi per arrivare alla convergenza della rete: si inserisce quindi un'ulteriore fase di raffinazione. Questa procedura viene eseguita dalla rete dopo aver terminato la fase di training e consiste nel ripristinare le posizioni dei vertici nelle aree senza buchi, senza attendere la convergenza dell'ottimizzazione.

Con l'aggiunta di questa fase si riscontra un miglioramento del reticolo rispetto a quello generato dalla rete GCN, secondo il metodo nella fase di training, mostrando in particolare delle lievi distorsioni della mesh nelle regioni senza buchi. La fase di raffinazione invece riduce sensibilmente queste deformazioni, migliorando quindi la qualità dell'output.

5. Strumentazione utilizzata

Oltre all'utilizzo di software CAD per la modellazione di componenti meccanici, per svolgere l'attività di reverse engineering, occorre utilizzare altre strumentazioni e apparecchiature, quali scanner e stampanti 3D. Una volta completata la parte relativa all'acquisizione dati dei file CAD, si procede con la loro elaborazione tramite la rete neurale. Quest'ultimo passo può richiedere un ingente onere computazionale, come emerge anche dal report fornito dagli autori della SeMIGCN, i quali per sviluppare la rete neurale hanno utilizzato un computer le cui specifiche sono riportate nel capitolo precedente e possono fornire un'utile informazione sulla capacità computazionale effettivamente richiesta.

5.1 Software

La rete neurale è stata implementata su linguaggio Python e per questioni di compatibilità si è utilizzato Linux come sistema operativo per svolgere tutte le operazioni che competono alla rete. Per quanto riguarda la parte di modellazione CAD e della mesh si sono utilizzati programmi compatibili con Windows 11.

Per la parte del lavoro riguardante la modellazione 3D e la conseguente elaborazione della mesh poligonale, si sono utilizzati i seguenti software:

- **PTC Creo parametric student 11**: questo software CAD, permette la modellazione 3D di componenti meccanici anche di forme complesse. La licenza del programma risulta gratuita solo per gli studenti universitari. È stato utilizzato per la progettazione dei pezzi dati in pasto alla rete;
- **Blender 4.2**: questo software di modellazione 3D è completamente opensource, molto utilizzato nell'ambito della computer grafica, quali creazione di scene di animazione per la realizzazione di contenuti cinematografici. Rispetto a Creo, permette un migliore controllo della mesh poligonale, consentendo operazioni quali il remeshing e la decimazione. È stato utilizzato per l'elaborazione della mesh, così da correggere eventuali problemi di lettura da parte della rete, ma anche per simulare rotture nel reticolo che la rete avrebbe poi corretto.

Relativamente alla fase della scansione di componenti tridimensionali si utilizza il software **RevoScan 5**, che risulta scaricabile gratuitamente dal sito del costruttore. Revoscan 5 realizza un'interfaccia tra lo scanner e l'utente, permettendo in primo luogo una visualizzazione grafica in tempo reale della scansione, potendo così correggere eventuali problematiche che incorrono. In secondo luogo il software permette una

elaborazione post-process, ottenendo così una qualità di output sensibilmente maggiore.

Tramite il software Revoscan 5 risulta possibile modificare l'esecuzione della scansione 3D al fine di ottenere il risultato migliore. Risulta quindi possibile modificare le impostazioni di acquisizione dati. Considerando ora l'allineamento, Revoscan 5 offre due modalità di tracciamento tramite:

- **Feature**: lo scanner si concentra sulla rilevazione di caratteristiche del pezzo. Le feature rappresentano dei punti di riferimento unici sulla superficie dell'oggetto, per esempio possono essere curve, angoli o dettagli tipici.
- **Marker**: durante la scansione la posizione dei punti risulta in funzione dei marker, ovvero puntini adesivi attaccati in zone strategiche. Il vantaggio di questo metodo consiste nel poter effettuare più scansioni conservando le informazioni sulle posizioni dei vari punti, in questo modo risulta possibile muovere a piacimento la testa dello scanner, senza rischiare la sovrapposizione dei punti.

Data la versatilità dell'ultima metodologia presentata, tutte le scansioni svolte in questo elaborato sono state fatte tracciando i marker.

Una volta ultimata la mappatura 3D del pezzo in esame, la nuvola di punti acquisita viene elaborata dal software unendo i vari punti per ottenere una superficie. Le funzioni implementate nel programma permettono un trattamento aggiuntivo al fine di perfezionare il risultato finale. Nei passaggi finali si procede con la costruzione della mesh su componete, potendo anche in questo caso scegliere la misura target per raggiungere il grado di finezza desiderato.

L'output prodotto può presentare una mesh incompleta con aree dove i punti non sono stati registrati per via delle zone d'ombra. Per cercare di risolvere queste mancanze Revoscan 5 possiede una specifica funzione di riempimento ma non controllabile da parte dell'utente, la quale permette solo di selezionare le superficie che il software indica come aperte, senza però poter modificare la metodologia di riempimento.

Per quanto riguarda invece la rete neurale, si procede ora con l'analisi dei contenuti forniti dalla libreria opensource. Su GitHub gli autori della rete neurale SeMIGCN hanno fornito tutto il necessario per poterla rendere utilizzabile da altri utenti con accesso gratuito. Sul sito vengono indicati i requisiti minimi dell'ambiente per poter utilizzare la rete neurale:

- Versione Python 3.10;
- Versione Torch 1.13.0;
- Torch-geometric 2.2.0;

Come evidenziato dalle specifiche, la rete neurale viene implementata attraverso il linguaggio di programmazione Python. Per come risulta strutturato questo linguaggio, è possibile implementare funzioni built-in disponibili su internet, in modo da semplificare e rendere più leggibile la sequenza delle istruzioni.

Di seguito vengono elencate tutte le librerie Python utilizzate dalla rete neurale SeMIGCN:

- **Pytorch**: al cui interno sono contenuti gli strumenti per svolgere attività di deep learning, come la costruzione e l'addestramento di modelli. Il suo nome abbreviato è Torch;
- **Pytorch geometric**: questa libreria open source rappresenta un'estensione di Pytorch e al suo interno sono contenute funzioni built-in adatte per lavorare con dati che non strutturati, ma che possiedono una struttura a grafo.

Per quanto riguarda nello specifico gli script, nella parte iniziale vengono caricate una serie di librerie per permettere alla rete di funzionare in maniera corretta. L'elenco di queste librerie si trova nel file requirement.txt, fornito dagli autori di SeMIGCN. Tali librerie vengono caricate mediante l'esecuzione del file **Dockerfile**.

Per una panoramica completa si propone di seguito un elenco dei principali moduli d'interesse:

- Dockerfile: in esso si trova la versione di Docker compatibile per eseguire l'installazione della rete neurale.
- Set di dati.zip: all'interno di questa cartella sono presenti dei file di tipo .obj da utilizzare come prove per vedere se la rete neurale funziona;
- Pre-elaborazione: in questa cartella è presente il file prepare.py che si occupa della parte di preelaborazione della mesh, in quanto non è possibile fornire la mesh incompleta di input direttamente alla fase di training. All'interno sono contenute le istruzioni per svolgere le seguenti fasi:
  - MeshFix, per un riempimento iniziale dei fori;
  - Remeshig, sul reticolo per uniformare il campionamento su tutta la superficie;
  - Normalizzazione della scala;
  - Smoothing, per ottenere una superficie più liscia possibile.
- sgcn.py: in questo file è contenuto l'algoritmo per eseguire la fase di training per quanto riguarda la rete convoluzionaria a risoluzione singola SGCN;
- mgcn.py: all'interno di questo file si trova l'algoritmo con le istruzioni necessarie per eseguire la fase di training per quanto riguarda la rete convoluzionaria a risoluzione multipla MGCN;

Ogniuna di queste funzioni vengono attivate mediante specifici comandi.

Per la valutazione finale dei risultati ottenuti, si esegue il confronto tra le mesh di input e i modelli ottenuti al termine delle rispettive elaborazioni. Per effettuare tali paragoni, non risulta possibile l'utilizzo del software Blender, in quanto non permette questa funzionalità, per ciò questa analisi viene svolta tramite il software **MeshLab 2023.12**.

Questo programma, facilmente scaricabile e installabile dal sito internet e completamente opensource, è dotato di varie funzionalità di elaborazione, quali:

- Allineamento di mesh in input in un sistema di riferimento comune per permettere la successiva di comparazione, al fine di ottenere un'analisi tra i modelli a confronto;
- Ricostruzione del pezzo in ingresso al fine di ottenere una mesh poligonale di output completa e chiusa;
- Applicazione di filtri automatici per modificare la resa visiva del componente e permettere la rimozione di eventuali errori. È inoltre possibile ottenere informazioni riguardanti il colore;
- L'esportazione non solo in formato .stl, così da consentirne la stampa tridimensionale, ma anche in altre modalità che ne permettono l'esportazione su altri software;
- Modifica spaziale del pezzo, tramite una modellazione delle dimensioni, della posizione e dell'orientamento. È inoltre possibile applicare una modifica strutturale alla mesh poligonale.

5.2 Hardware

La parte hardware ha richiesto l'utilizzo di uno scanner 3D e una stampante 3D.

Attraverso l'uso di uno scanner 3D **Revopoint Mini**, risulta possibile la ricostruzione di un modello digitale di un pezzo tridimensionale, tramite l'acquisizione dei punti appartenenti alla superficie del componente e di una successiva elaborazione in cui viene generata la mesh poligonale tramite un software dedicato. Il Revopoint Mini è stato fornito dall'università di Bologna e costituisce un buon compromesso in termini di prestazioni e costi.

Revopoint Mini è uno scanner 3D a luce blu con alta precisione, che può acquisire misure reali minime pari a 0.2 mm. Questo prodotto trova applicazione in ambito odontoiatrico per la realizzazione di protesi, per la realizzazione di gioielli e nel reverse engineering.

Lo scanner acquisisce la superficie in esame mediante una nuvola di punti e grazie alla notevole precisione fornita dallo strumento, risulta possibile rilevare minimi dettagli ottenendo così un elevata qualità nel modello di output.

Per realizzare un'alta risoluzione a fronte delle problematiche ambientali, come l'effetto della luce dell'ambiente dove si sta svolgendo la scansione, Revopoint utilizza una luce blu di classe 1 ad alta risoluzione, resistente alla luce ambientale e innocua per il corpo umano.

In termini di prestazioni, riveste un ruolo importante anche la velocità di scansione, che risulta pari a 10 fotogrammi al secondo (10 fps).

Revopoint Mini è costituito dai seguenti strumenti:

- **RevoScan 5**: software proprietario di interfaccia tra lo scanner 3D e il computer, scaricabile gratuitamente dal sito del costruttore. Permette di visualizzare il grafico della scansione potendo modificarne i parametri e strategie di acquisizione. Inoltre, una volta completata la digitalizzazione del modello 3D, permette una rielaborazione del risultato;
- **Testa scanner** : costituisce lo strumento principale per la scansione 3D, il quale si occupa di acquisire e digitalizzare la superficie del pezzo in esame. Di piccole dimensioni e peso contenuto, possiede nella parte inferiore una guida per inserire il treppiede;

Figura 40. Testa scanner.

- **Treppiede**: è un dispositivo che provvisto di un braccio telescopico, permette di variare l'altezza della testa scanner;

Figura 41. Treppiede.

- **Stabilizzatore portatile**: funge da supporto per utilizzare lo smartphone come dispositivo scanner. Questo stabilizzatore, inoltre, è provvisto di un giroscopio ad alta precisione integrato ad un dispositivo di stabilizzazione, in modo da permettere una scansione più fluida possibile;

Figura 42. Stabilizzatore portatile.

- **Giradischi a doppio asse**: il modello viene designato dal costruttore con la sigla A230. Svolge la funzione di base girevole e inclinabile dove posizionare i pezzi per ottenere una migliore stabilità della scansione. La tavola rotante permette una rotazione completa su 360°, mentre l'inclinazione massima risulta essere di 30°. Questi due gradi di libertà si possono controllare attraverso specifici comandi all'interno del software fornito dal costruttore, risultano infatti realizzabili piccole rotazioni orizzontali e inclinazioni della tavola. L'utilizzo di questa tavola migliora sensibilmente il risultato finale, in quanto permette l'acquisizione dei punti in zone difficili da visualizzare e il completamento di porzioni mancanti sulla superficie;

Figura 43. Giradischi.

- **Tester**: si tratta di una statuina di colore bianco con forme molto complesse. Il su compito è quello di testare il funzionamento dello scanner;
- **Accessori vari**: sono disponibili degli accessori ausiliari quali una bomboletta contenete vernice bianca per ridurre la riflessione della superficie dei pezzi che si vuole scannerizzare e dei puntini adesivi chiamati marker, che una volta attaccati alla tavola rotante o al componente in esame, permettono l'applicazione in una specifica modalità di acquisizione dati impostabile tramite il software.

Al termine dell'elaborazione risulta possibile esportare il pezzo scansionato con formato .obj o .stl.

Lo scanner viene collegato al computer mediante un cavo USB già disponibile nella scatola. La tavola rotante viene collegata al computer via Bluetooth. Attraverso il software risulta possibile verificare che tutto sia connesso correttamente.

Di seguito vengono riassunte le caratteristiche tecniche di Revopoint Mini:

- Precisione di acquisizione reale durante la scansione pari a 0.2 mm;
- Precisione del singolo fotogramma pari a 0.05 mm;

- Velocità di scansione massima pari a 16 fps (fotogrammi al secondo);
- Intervallo della singola acquisizione pari a 168x132 mm;
- Risoluzione della fotocamera di profondità e telecamera RGB pari a 2MP;
- Possibilità di scansione a colori;
- Peso dello strumento pari a 175g;
- La distanza tra lo scanner e la superficie può variare da 120 mm fino a 250 mm;
- Lo scanner possiede un processore dual core con 1.8 GHz;

La stampa 3D è una tecnologia che negli ultimi anni ha conosciuto un grande sviluppo, semplificando notevolmente il processo di ingegnerizzazione del prodotto riducendone anche le tempistiche.

Questa tecnologia è stata impiegata per svolgere l'attività di reverse engineering all'interno dell'elaborato. In questa fase risulta quindi di fondamentale importanza avere a disposizione il pezzo fisico per poter eseguire l'analisi del componente tramite scanner 3D, al fine per verificare la reale efficacia dell'applicazione della rete neurale al reverse engineering.

Questo scopo si può raggiungere mediante l'impiego di una stampante 3D, la quale attraverso un software di interfaccia, riceve in input il modello CAD da trasformare in un componete fisico. Il software inoltre permette di impostare la modalità di stampa potendo risolvere problematiche come sottosquadri.

In questo elaborato la stampa 3D è stata commissionata all'Università di Bologna e quindi non saranno approfondite le parti riguardanti l'utilizzo del software e la gestione del processo di fabbricazione.

Con rifermento all'apparecchiatura utilizzata, il modello di stampante 3D impiegato prende il nome di Bambulab x1-carbon.

Figura 44. Bambulab x1-carbon.

Questo modello di stampante 3D risulta uno dei più avanzati sul mercato e presenta funzionalità di alta velocità di deposizione del polimero ed è in grado di stampare in multicolore. Inoltre, da catalogo presenta le seguenti specifiche generali:

- Alta qualità del prodotto finito tramite risolutore Lidar di 7 μm;
- Stampa multicolore e possibilità di stampa multimateriale;
- Ampia flessibilità e personalizzazione della stampante;
- Core XY ad alta velocità con accelerazione massima pari a 20 m/s^2 ;
- Doppio livellamento automatico.

Il costruttore fornisce le seguenti specifiche tecniche del prodotto:

Specifiche tecniche Bambulab X1-carbon	
Volume di costruzione (L×P×A)	256 × 256 × 256 mm³
Ugello	0.4 mm Acciaio temprato incluso
Hoteend	Interamente in metallo
Temperatura massima hot-end	300 ℃
Diametro filamento	1.75mm
Filamento supportato	PLA, PETG, TPU, ABS, ASA, PVA, PET Ideale per PA, PC, polimeri rinforzati con fibra di carbonio/vetro
Superficie piastra di costruzione	Piatto Bambu Textured PEI **o** Bambu Cool Plate (Preinstallato, casuale, entrambi compatibili con Micro Lidar)
Temperatura massima piastra di costruzione	110℃@220V, 120℃@110V
Velocità massima della testa dell'utensile	500mm/s
Accelerazione massima della testa dell'utensile	20 m/s²
Dimensioni fisiche	X1C ： 389 × 389 × 457 mm³ Dimensioni del pacco 480 × 480 × 535 mm³ Peso netto 14,13 kg, Peso lordo 18 kg X1C Combo: Dimensioni del pacco 480 × 480 × 590 mm³ Peso lordo (incluso AMS) 22,3 kg
Requisiti elettrici	100-240 VAC, 50/60 H, 1000W@220V, 350W@110V

Sul sito Bambulab, inoltre viene offerta ampia scelta di materiali e colori per i filamenti. Nel caso specifico, per la realizzazione dei pezzi meccanici di questo elaborato, si è ritenuto più idoneo scegliere come materiale il PLA basic. Per quanto riguarda il colore, la scelta è ricaduta sul bianco, in quanto la cromia della superficie riveste un ruolo fondamentale, perché la qualità finale della scansione 3D risulta pesantemente influenzata da questo aspetto.

Di seguito vengono riportate le proprietà del polimero PLA basic, fornite a catalogo:

Impostazioni di stampa consigliate	
Impostazioni di asciugatura (Forno di asciugatura)	55 °C, 8 h
Stampa e mantenimento umidità del contenitore	< 20% RH (sigillato, con essiccante)
Temperatura Nozzle	190 - 230 °C
Temperatura piatto (con colla)	35 - 45 °C
Velocità di stampa	< 300 mm/s

Proprietà Fisiche	
Densità	1.24 g/cm³
Temperatura di rammollimento Vicat	57 °C
Temperatura Deflessione Termica	57 °C
Temperatura di fusione	160 °C
Indice di fusione	42.4 ± 3.5 g/10 min

Proprietà meccaniche	
Resistenza alla trazione	35 ± 4 MPa
Tasso allungamento alla rottura	12.2 ± 1.8 %
Modulo flessione	2750 ± 160 MPa
Resistenza alla flessione	76 ± 5 MPa
Forza d'urto	26.6 ± 2.8 kJ/m²

Figura 45. Specifiche PLA.

5.3 Server Unibo

Nel capitolo precedente, la scelta della rete è ricaduta su SeMIGCN. In seguito ad un approfondimento realizzato grazie all'articolo messo a disposizione dagli autori, emerge che per poter utilizzare la rete neurale, risulta necessario un'elevata potenza di calcolo e un'ampia capacità di memoria per quanto riguarda la scheda grafica.

Di seguito vengono riproposte le specifiche del computer utilizzato dagli autori per lo sviluppo di SeMIGCN:

- CPU intel Core i7 5930K(3.5 GHz, 6 core);
- GPU NVIDIA GeForce TITAN X avente 12 Gb;

Da ciò risulta evidente che per non incorrere in problemi di Hardware, si deve impiegare una macchina con specifiche similari. Anche in questo caso l'Università di Bologna ha messo a disposizione un server con collegamento da remoto avente le seguenti specifiche:

Specifiche tecniche server Unibo	
Modello	Lenovo ThinkSystem ST650 V2
RAM	125 GB
Processore	Intel® Xeon(R) Silver 4314 CPU @ 2.40GHz × 32
Scheda video	NVIDIA RTX A2000 12GB/PCIe/SSE2
Hard Disk	4 TB

Il processo di training della rete neurale risulta essere una procedura delicata dal punto di vista computazionale, con il rischio di non poter portare a termine l'elaborazione. Risulta possibile monitorare le risorse impiegate tramite l'implementazione del comando:

```
/watch nvidia-smi
```

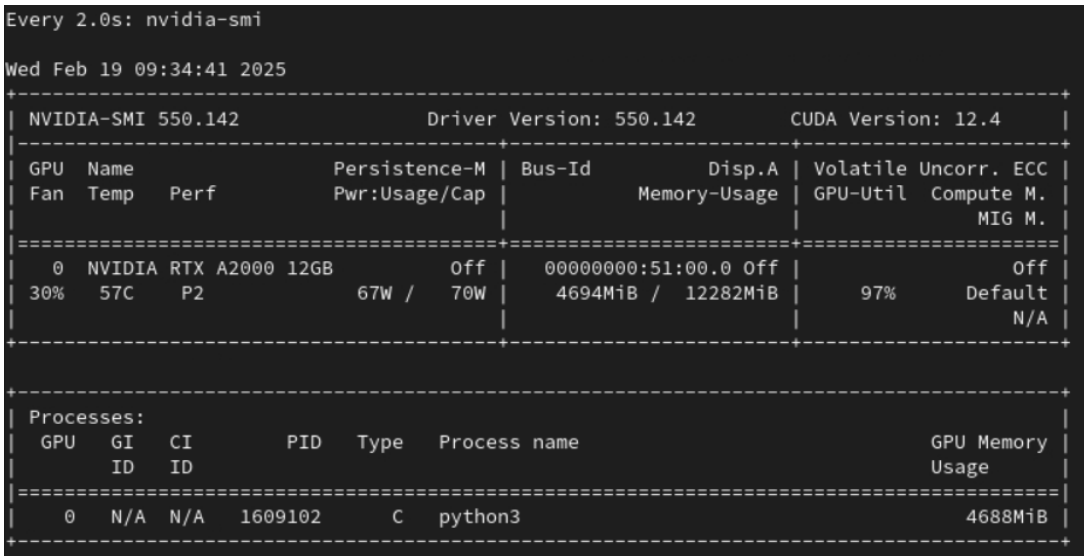


Figura 46. Schermata di monitoraggio risorse del computer.

6. Metodologia

Si può ora procedere con la presentazione accurata della metodologia implementata in questo elaborato. Si è partiti sottoponendo alla rete i file forniti su GitHub dai creatori così da verificarne il corretto funzionamento:

- `{mesh-name}_original.obj`: è il modello con la mesh incompleta. È il file di input dato in pasto alla rete neurale, così da verificare come questa si comporta nel caso di mesh incomplete;
- `{mesh-name}_gt.obj`: corrisponde al ground truth, ovvero il modello di riferimento corretto e completo, in cui non sono presenti errori nella mesh. Viene utilizzato nella fase di evaluation, al fine di confrontare ciò che la rete fornisce in output dopo l'elaborazione del file `{mesh-name}_original.obj`.

Da questi test preliminari risulta evidente che la rete può ricevere in ingresso solo mesh di tipo triangolare ed esclusivamente in formato .obj. In aggiunta, come specificato dai creatori, al fine di elevare la qualità del risultato finale, occorre fornire alla rete una mesh il più possibile uniforme.

6.1 Fase preliminare di testing

Una volta verificato il funzionamento della rete neurale, sono stati forniti a SeMIGCN file CAD creati appositamente tramite il software Creo. È bene fare questo primo test, al fine di verificarne i risultati con altri componenti. In questo modo, è possibile avere un'idea preliminare di ciò che si otterrà prima di dare in ingresso il file ottenuto tramite la stampa 3D e lo scanner 3D, così da evitare uno spreco di materiale.

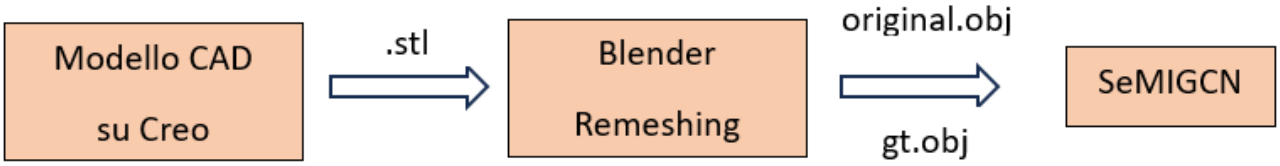
I file CAD così creati e dati in pasto alla rete hanno evidenziato due problematiche:

1. Il limite di memoria grafica, dovuto all'eccessivo numero di triangoli nella mesh;
2. La lettura corretta dei fori da parte della rete nella fase di preprocess.

Per quanto riguarda il problema dell'utilizzo della scheda grafica è emerso da diverse prove che con una scheda nvidia G-force da 4096 Mb di memoria, il numero massimo di triangoli gestibile risulta intorno ai 10.000. Per risolvere questo problema e utilizzare un campionamento della mesh più fitto, si passa all'utilizzo di dispositivi hardware più performanti, facendo ricadere la scelta su server messo a disposizione dall'Università di Bologna. Con questo computer il limite massimo dei vertici non è stato raggiunto, ma dai test effettuati si potrebbe pensare un numero superiore ai 100.000.

Al termine della fase di preprocess, l'algoritmo fornisce in output il file `{mesh-name}_smooth.obj`, dove la mesh è stata riparata, uniformemente ricampionata e levigata. Questo file CAD rappresenta l'input per la successiva fase di training. Nel caso in cui la mesh sia troppo danneggiata, come nel caso di ampie porzioni mancanti, la rete neurale non riesce a ricostruire in maniera corretta il pezzo, chiudendo anche parti come fori, in quanto non li riconosce come tali, interpretandoli come un difetto da correggere.

Dopo una serie di test su modelli CAD di varia geometria, la soluzione a questa seconda problematica ha portato alla definizione della seguente pipeline di lavoro:



Per risolvere, non è stata fatta alcuna variazione al codice della rete, quindi non è stato modificato il file prepare.py, che permette preelaborazione della mesh. Si è invece intervenuti variando la procedura con cui vengono creati i modelli CAD.

Il modello CAD è stato progettato tramite il software Creo Parametric 11, dal quale si ottiene un file in formato stereolitografica .stl, che viene modificato tramite il software Blender, intervenendo sulla mesh poligonale. Blender fornisce file in formato .obj, dati in pasto alla rete neurale.

Tramite Creo Parametric è possibile generare una mesh triangolare del ground truth, attraverso l'esportazione in formato .stl e settando a video in primo luogo un parametro di profondità del reticolo: 0.02, come mostrato nell'immagine seguente. Il software imposta di default un valore pari a 0.2 ma diminuendolo è possibile ottenere un campionamento più fitto.

Salvando tali impostazioni è possibile accedere alla schermata successiva, dove si può impostare i parametri geometrici della mesh. Un buon compromesso tra qualità di esportazione e consumo di memoria è rappresentato dai valori mostrati:

Parametro	Valore
Altezza corda	0.3
Controllo angolo	0.5
Dimensione passo	1

Tali valori sono stati impostati e mantenuti per tutti i file groud truth dei componenti presentati nei casi studio mostrati di seguito.

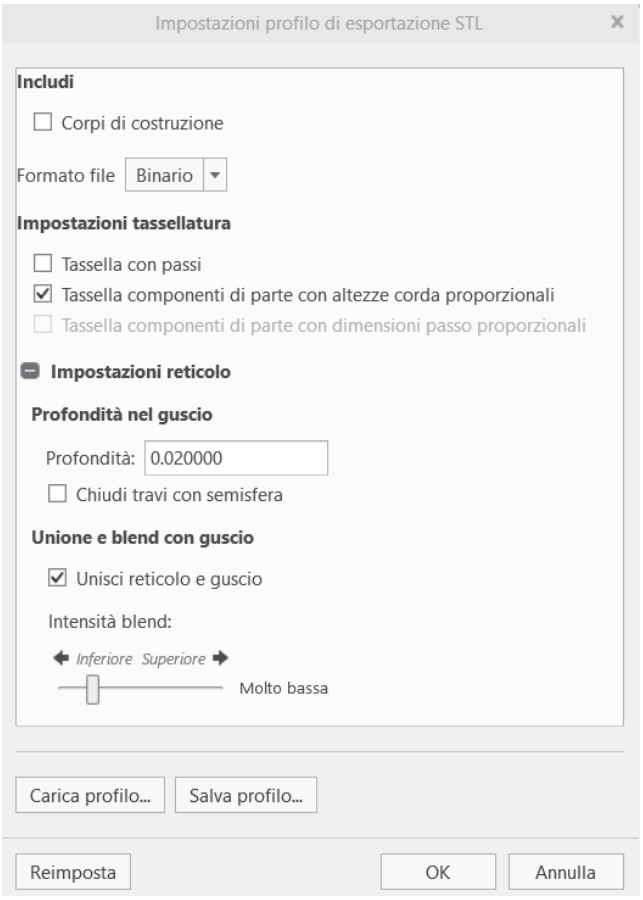


Figura 47. Schermata per settare la profondità del reticolo.

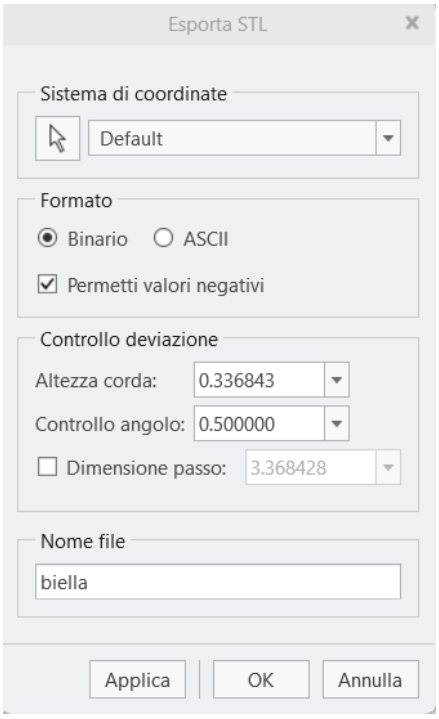


Figura 48. Parametri dimensionali della mesh poligonale.

La scelta di utilizzare due software è dovuta, oltre alla problematica in esame, anche alle seguenti motivazioni:

1. Creo permette di creare in maniera semplice il componente desiderato e contiene molte più funzionalità di modellazione rispetto a Blender.
2. Blender permette una migliore elaborazione del reticolo poligonale rispetto a Creo.
3. Dopo l'esportazione in .stl su Creo il modello tridimensionale viene ricostruito tramite un reticolo poligonale definito da parametri di default non controllabili. Su Blender si riesce invece, tramite i modificatori, a ottenere una mesh poligonale ottimizzata con un campionamento definibile dall'utente.
4. Tramite Blender è sempre possibile ricavare una nuova mesh, modificandone i parametri, indipendentemente da quella precedente ricavata. Con Creo, invece, una volta ottenuto il file .stl non è più modificabile. Eventuali modifiche richiedono la rielaborazione del pezzo originale.

Riguardo al punto 4, è bene tenere conto che ricavare una nuova mesh nella maggior parte dei casi genera un reticolo eccessivamente fitto, che può rendere dispendiose le successive operazioni. Ne consegue l'esigenza di un ulteriore elaborazione di decimazione della mesh, rendendo così compatibile il reticolo generato con le prestazioni del computer a disposizione. Queste operazioni sulla mesh sono già implementate all'interno del programma e non richiedono alcuna programmazione.

Il modificatore *Ricava nuova mesh*, presente su Blender, permette di scegliere tra varie opzioni. Quelle utilizzate risultano essere Nitido e Voxel:

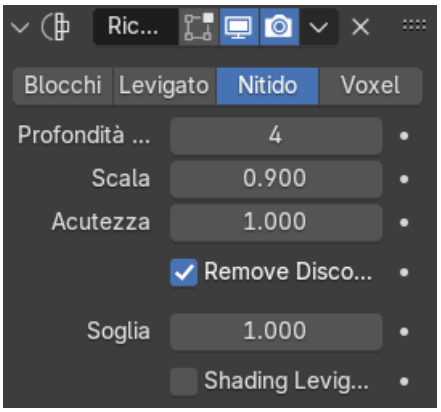


Figura 49. Modificatore Nitido.

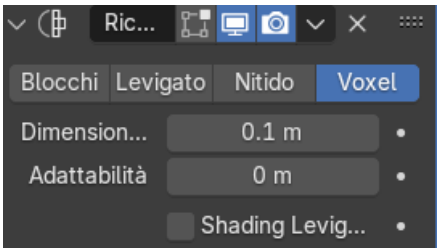


Figura 50. Modificatore Voxel.

La scelta tra quale opzione utilizzare dipende dal risultato visivo che meglio soddisfa le specifiche di qualità. Variando il valore di profondità nel primo caso e di dimensione

nel secondo, si ottiene una variazione nel campionamento della superficie del componente. Si lasciano inalterati gli altri parametri.

Si è verificato che seguendo la procedura descritta dalla pipeline, la problematica legata alla lettura corretta dei fori è stata risolta in maniera robusta.

6.2 Procedura operativa di utilizzo della rete

Nei capitoli precedenti si è trovata e analizzata a livello teorico la rete disponibile pre-addestrata che meglio soddisfa le specifiche di questo elaborato. Si prosegue in questa parte con l'utilizzo pratico della rete neurale. Quanto segue viene svolto interamente sul sistema operativo Linux, inserendo i comandi direttamente dalla console.

Nell'utilizzo della rete neurale gli autori forniscono una linea guida formata da tre fasi:

1. Preprocess;
2. Training;
3. Evaluation.

Per poterle eseguire, occorre svolgere i passaggi come di seguito.

6.2.1 Creazione dell'ambiente di lavoro

Una volta installata la rete seguendo le indicazioni fornite su GitHub, per poter utilizzare a livello paratico la rete neurale, è necessario implementare delle funzioni aggiuntive per preparare l'ambiente di lavoro, semplificando ulteriormente l'utilizzo di SeMIGCN. Tuttavia, occorre specificare che questa operazione non è obbligatoria per il corretto funzionamento della rete.

I tre comandi aggiuntivi vengono definiti come segue:

- **start.sh**: si trova all'interno della cartella SeMIGCN. Per utilizzarlo da console una volta raggiunta la sua posizione nella directory, si utilizza il comando `./start.sh`.
  Il compito di start.sh è quello di avviare il contenitore Docker, determinare il collegamento alla porta 8081 e utilizzare tutte le GPU disponibili. Inoltre costruisce la directory del contenitore Docker. Lo script è stato implementato utilizzando le ultime due righe della parte relativa all'installazione del Docker su GitHub.
  Il codice utilizzato è il seguente:

```
#!/bin/bash
sudo systemctl start docker
```

```
sudo docker run -itd --gpus all -p 8081:8081 --name semigcn -v .:/work astaka-pe/semigcn
sudo docker exec -it semigcn /bin/bash
```

- **update.sh**: una volta avviato lo start.sh si procede con la rimozione e reinstallazione delle librerie utilizzate negli script successivi. Questa operazione di disinstallazione e reinstallazione risulta necessaria per risolvere eventuali conflitti o problemi di compatibilità, nel caso alcune di esse siano già presenti e installando le versioni corrette. Per avviare lo script si utilizza da console il comando `./update.sh.`
  Il codice viene mostrato come segue:

```
#!/bin/bash
pip uninstall numpy pymeshfix
pip install numpy pymeshfix
pip uninstall numpy scikit-learn
pip install numpy scikit-learn
pip install "numpy<2"
```

  Una volta lanciato il programma vengono visualizzate a terminale le richieste di autorizzazione all'installazione dei vari pacchetti Python, si fornisce in ogni caso risposta affermativa con il comando `Y`.
- **stop.sh**: rappresenta la funzione che ferma i comandi contenuti all'interno del Docker e in seguito rimuove fisicamente la cartella semigcn. Anche in questo caso per eseguire la funzione si utilizza il comando `./stop.sh.`
  Il codice è presentato dalle seguenti istruzioni:

```
#!/bin/bash
sudo docker stop semigcn
sudo docker rm semigcn
sudo systemctl stop docker
```

  Una volta eseguito lo stop e la rimozione del Docker per avviare nuovamente l'ambiente di lavoro occorre lanciare in successione prima start.sh e poi il comando update.sh. Non è possibile eseguire nessun comando della rete neurale senza prima aver lanciato questi comandi di iniziazione e di aggiornamento.

6.2.2 Preparazione della cartella per i test

Una volta implementato correttamente l'ambiente di lavoro, il passo successivo è quello di caricare correttamente i modelli CAD, in modo che la rete neurale possa acquisirli correttamente. La procedura in questione è definita dagli autori su GitHub. Come affermato in precedenza la rete neurale non necessita di un dataset di dati per l'addestramento, ma riceve in ingresso un'unica mesh incompleta. Per cui in una nuova cartella chiamata `{mesh-name}`, si inseriscono due modelli CAD in formato .obj, secondo la designazione:

- `{mesh-name}_gt.obj`: indica il nome del file CAD corrispondente al modello ideale o ground truth dove la mesh poligonale non presenta problematiche provenendo direttamente dal software di modellazione.
- `{mesh-name}_original.obj`: il modello CAD in questione presenta una mesh incompleta o danneggiata. Questo il file rappresenta l'input della rete neurale su cui vengono svolte le successive operazioni di elaborazione.

6.2.3 Preprocess

Avendo caricato i file in formato .obj nella directory selezionata e avendo prestato attenzione al nome dei modelli CAD, si procede con l'elaborazione della mesh.

Per l'esecuzione di questa fase si utilizza il comando:

```
python3 preprocess/prepare.py -i datasets/**/{mesh-name}/{mesh-name}_original.obj
```

Con questa istruzione viene eseguita una elaborazione preliminare sulla mesh incompleta, in modo da renderla idonea per la successiva fase di training. In questa fase, si genera una mesh levigata e impermeabile a partire dal reticolo incompleto. Si hanno le seguenti fasi:

1. Riempimento dei buchi per mezzo di MeshFix;
2. Remeshing per uniformare il campionamento su tutta la superficie del pezzo;
3. Normalizzazione della scala;
4. Smoothing della mesh per ottenere un reticolo poligonale più liscio possibile.

Nell'istruzione sopra mostrata, risulta possibile definire dall'utente la lunghezza target del reticolo di remeshing aggiungendo alla fine della riga `-r{float}`, per esempio inserendo `-r 0.5`. Il valore di default è pari a 0.6. Se si imposta un valore più piccolo la mesh risulta più fine, con un valore maggiore il reticolo appare più grossolano. Il limite inferiore dei valori è dato dalla memoria della scheda grafica a disposizione e in generale non si settano valori inferiori a 0.5.

```
[MeshFix]
[Remeshing]
[Normalizing scale]
[Smoothing]
[FINISHED]
root@din097080:/work# python3 preprocess/prepare.py -i datasets/test5/boccola/boccola_original.obj -r 0.5
input       : datasets/test5/boccola/boccola_original.obj
remesh      : 0.5
```

Figura 51. Comandi da terminale.

6.2.4 Training

Ottimizzata la mesh di input e resa idonea per il training, tramite l'elaborazione precedente, risulta possibile eseguire il training della rete. Questa fase può essere effettuata mediante la scelta di utilizzare una tra le due tipologie di reti neurali a disposizione, la prima a risoluzione singola nota come SGCN e la seconda a risoluzione multipla chiamata MGCN.

Su GitHub sono disponibili entrambe le istruzioni per lanciare l'addestramento:

```
python3 sgcn.py -i datasets/**/{mesh-name} # SGCN
python3 mgcn.py -i datasets/**/{mesh-name} # MGCN
```

La scelta se utilizzarne una piuttosto che un'altra rete deriva dalla morfologia dei componente in questione:

- SGCN: si tratta di un metodo a risoluzione singola. Si presta bene per la ricostruzione di piccole parti mancanti in superfici lisce dove la mesh non è molto complessa, ottenendo una buona qualità dell'output;
- MGCN: è una rete neurale multi-risoluzione, risulta più adatta alla ricostruzione di superfici più complesse potendo apprendere anche dettagli intricati, proprio per il fatto di essere multiscala e multiview.

Nei casi applicativi relativi a questo elaborato, i pezzi in esame sono tutti componenti meccanici con geometrie semplici, quindi si è scelto di implementare la rete neurale SGCN.

L'algoritmo oltre che ad elaborare e riparare la mesh, fornisce a intervalli regolari (ogni 10 epoche) il risultato grafico della rete tramite il collegamento presente nel Docker 8081. In questo modo risulta possibile avere un controllo più marcato del processo. Inoltre la rete neurale, sempre nel medesimo intervallo di epoche, stampa a video il valore di loss globale, ovvero l'errore.

Si riscontra che a terminale non tutti i valori della loss rimangono a video, per poterli immagazzinare e graficare si è apportata una modifica al codice, tramite la quale questi valori di loss vengono salvati mediante un file di testo chiamato `losses.txt`, posizionato all'interno della cartella del modello preso in esame. Si setta la directory a codice come segue:

```
# Percorso del file per salvare le perdite
loss_file_path = f"datasets/test5/boccola/losses.txt"
```

Per immagazzinare i valori della funzione di loss con un intervallo temporale di 10 epoche si utilizza il comando:

```
# Stampa della loss ogni 10 epoche

if epoch % 10 == 0:            # verifico che l'epoca sia multiplo di 10;
    print(f"Epoch {epoch} - Loss Pos: {epoch_loss_p}, Loss Norm: {epoch_loss_n}, Total Loss: {epoch_loss}")
```

Al fine di migliorare la qualità del risultato finale della rete neurale, gli autori permettono di aggiungere alle istruzioni di lancio di SGCN e MGCN la tipologia del modello CAD che si sta esaminando, potendo scegliere tra diverse opzioni:

- CAD: se il modello 3D proviene da un software CAD;
- Real: nel caso il modello sia stato ottenuto da una scansione tridimensionale;
- Cache: opzione utile per ridurre il tempo di calcolo;
- Mu: rappresenta il peso di raffinazione.

Il mancato inserimento di queste informazioni aggiuntive non compromette il funzionamento del processo di training. A video vengono visualizzati nella raccolta dei parametri dove se non vengono implementati viene mostrata la scritta `False`, mentre al contrario presenta la scritta `True`. Nel caso in cui si tiene conto della tipologia del file, la qualità finale del componete risulta sensibilmente migliorata, in quanto le forme come gli spigoli risultano essere ben definiti rispetto allo scenario in cui si hanno a video tutti `False`.

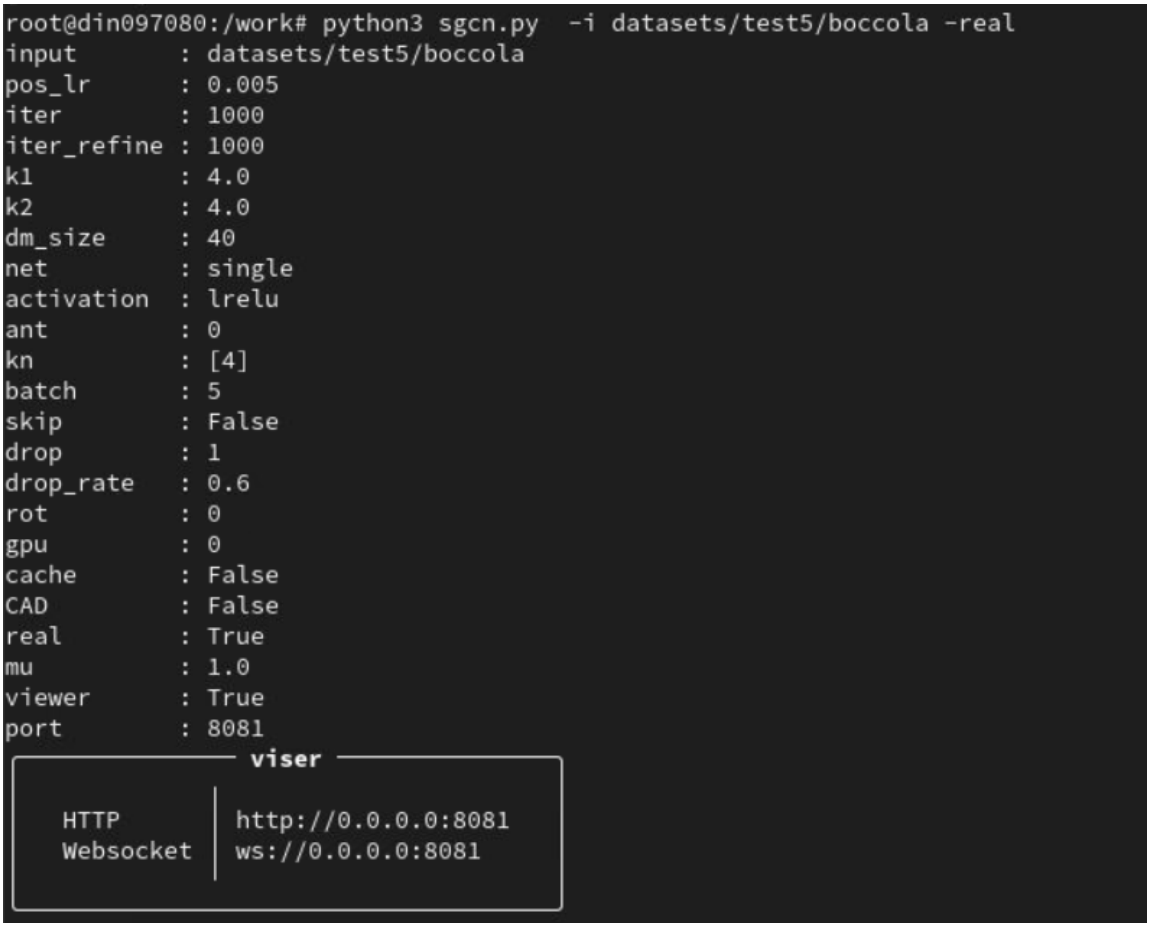


Figura 52. Parametri training da terminale.

6.2.5 Evaluation

Una volta ultimato il training della rete neurale, segue la fase della valutazione, che consiste nel comparare le mesh per essere ponderate. Si tratta di un'operazione resa opzionale in questo elaborato.

L'algoritmo genera una nuvola di punti in corrispondenza delle parti mancanti della mesh del file `{mesh-name}_original.obj`, costruita nella precedente fase di training dalla rete. Questa nuvola dei punti si trova all'interno del file `inserted.obj` nella cartella chiamata `{Colored}`.

Per la fase di valutazione, occorre disporre correttamente i file secondo una directory ben precisa. Si crea manualmente una cartella di nome `{Comparison}` all'interno della folder corrispondente al componete in esame `{mesh-name}` e vi si posizionano i file con la seguente designazione:

- `{mesh-name}_original.obj` deve essere rinominato con il nome `original.obj`;
- `{mesh-name}_gt.obj` deve essere rinominato con il nome `gt.obj`;

Posizionati i file e rinominati nel modo corretto, da console si esegue il seguente comando:

```
python3 check/batch_dist_check.py -i datasets/**/{mesh-name}
```

Questa istruzione crea la cartella `{Colored}` e al suo interno si genera il file `inserted.obj` contente la nuvola di punti corrispondenti alle parti mancanti della mesh incompleta di input. Per visualizzare il risultato finale, si importa su Blender i file .obj sopra citati e nelle parti in cui la mesh risulta mancante, si dispongono in maniera automatica i vertici ottenuti dalla fase di training.

6.3 Workflow

Dopo aver risolto le problematiche generali sulla lettura della geometria del componente da parte della rete neurale SeMIGCN e aver definito come questa si utilizza nella pratica, si procede con sperimentazione effettiva.

Si procedere come mostrato nella seguente figura.

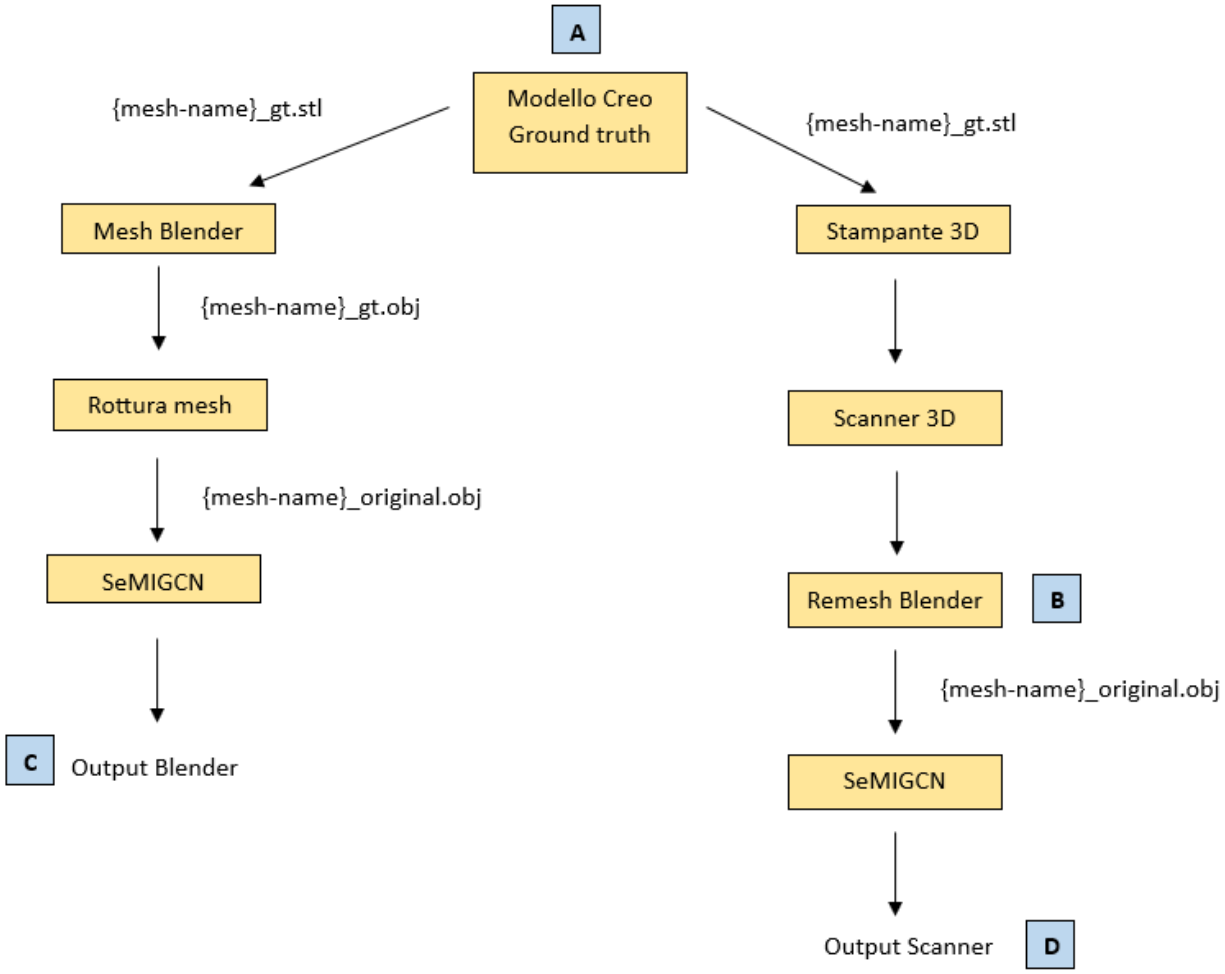


Figura 53. Pipeline di lavoro.

La metodologia generale seguita consiste nelle due seguenti procedure:

1. Modellazione del componente su Creo, per generare il `{mesh-name}_gt.stl`;
   - Mesh tramite Blender per ottenere il file `{mesh-name}_gt.obj`;
   - Tramite Blender si opera la rottura controllata del reticolo poligonale, ottenendo il file `{mesh-name}_original.obj`;
   - Il file `{mesh-name}_original.obj` avente quindi mesh incompleta viene elaborato dalla rete neurale.
2. Modellazione del componente su Creo, per generare il `{mesh-name}_gt.stl`;
   - Stampa in 3D del pezzo tramite Bambulab X1-Carbon;
   - Scansione tridimensionale tramite lo scanner Revopoint Mini e successiva elaborazione attraverso il software Revoscan 5;
   - Creazione del file `{mesh-name}_original.obj` ottenuto dal risultato della scansione 3D e dal remesh fatto tramite Blender, al fine di correggere eventuali imperfezioni nate dalla scannerizzazione;
   - Il file viene dato in pasto alla rete neurale per l'elaborazione.

Si esegue infine il confronto tra i risultati ottenuti tramite l'utilizzo di MeshLab. I confronti vengono eseguiti tenendo conto della seguente nomenclatura:

- A → ground truth, file in formato .stl;
- B → risultato scansione 3D;
- C → output rottura Blender;
- D → output scansione.

Sono quattro i confronti che vengono eseguiti:

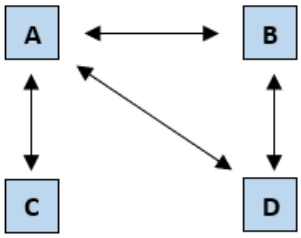


Figura 54. Confronti da eseguire tramite MeshLab.

L'intenzione è quella di ottenere una discrepanza dimensionale tra il ground truth (A) e l'output della scansione (D) paragonabile alla risoluzione dello scanner 3D ($\sim 0.2mm$). Il filtro utilizzato per effettuare il confronto tra le mesh è Distance from Reference Mesh, che confronta i vertici tra le elaborazioni in esame, tenendo come riferimento quella di groud truth (A), oppure il file originato dalla scansione (B), nel caso del confronto B-D.

6.4 Ricerca dei parametri di ottimo

Per quanto riguarda la parte relativa all'utilizzo della rete, come affermato in precedenza, si ricercano i valori di ottimo dei parametri per minimizzare la funzione di loss. Per determinare questi valori si utilizza l'algoritmo di training sgcn.py, all'interno del quale i principali parametri della rete sono raccolti in un elenco e facilmente modificabili. Al fine di assicurarsi che le modifiche siano salvate correttamente, al lancio del training della rete, da terminale si ha la visualizzazione dei parametri e dei corrispondenti valori utilizzati dall'algoritmo.

Per poter effettuare le opportune modifiche, occorre conoscere cosa rappresentano i parametri interessati e come la loro variazione impatta sulla loss:

- `pos_lr`: rappresenta il learning rate. Se si impostano valori troppo alti, la rete neurale diverge, mentre con valori più bassi si raggiunge la convergenza, ma il tempo di computazione aumenta;

- `iter`: indica il numero di epoche, deve essere un numero elevato per ottenere una loss più bassa possibile. Tuttavia, se troppo alto si rischia l'overtraining, mentre se troppo basso si potrebbe avere un peggioramento della qualità dell'output;
- k_1 e k_2: sono parametri per configurare la profondità della rete;
- k_n: tramite questo parametro si indica il numero dei nodi vicini da considerare per ogni neurone della rete;
- `batch`: questo parametro corrisponde alla dimensione del batch, ossia il numero di dati che la rete neurale elabora prima di aggiornare i pesi. Valori alti stabilizzano l'apprendimento, ma richiedono un maggiore consumo di memoria;
- `drop_rate`: il suo valore si riferisce alla percentuale del numero di nodi da saltare durante l'addestramento. In generale si imposta un valore basso per migliorare l'apprendimento, ma se il training presenta overfitting, occorre settare un valore più elevato;
- `dm_size`: rappresenta un indice della quantità di dati che un singolo neurone può elaborare.

Di seguito vengono riportati i valori dei parametri assegnati dagli autori della rete neurale SeMIGCN. Questi attributi offrono un buon compromesso tra la minimizzazione della funzione di loss e il consumo di risorse computazionali:

Parametro	Valore
Learning rate	0.01
N°epoche	100
Batch size	5
k1	4
k2	4
kn	4
Dm_size	40
Drop_rate	0.6

l fine di ottenere la qualità del risultato migliore possibile, nella pratica si agisce variando i parametri della rete, quali il numero di epoche e il learning rate, che risultano essere i più importanti da quanto emerso dal capitolo relativo al training della rete. Oltre a questi valori, sulla qualità dell'output finale impatta in maniera rilevante il campionamento della mesh fornita in input. Quanto affermato deve essere tenuto in considerazione per la sperimentazione sui componenti meccanici.

6.5 Qualità delle scansioni 3D

Al fine di facilitare l'utilizzo dello scanner 3D per l'acquisizione dei modelli tridimensionali, occorre tenere in considerazione la colorazione della superficie. La tonalità e la riflettività della superficie dei componenti – come anche il livello di illuminazione presente nella stanza – influisce pesantemente sulla qualità finale della scansione riflettendosi poi a cascata sull'output finale della rete neurale.

Da queste considerazioni risulta quindi necessario valutare attentamente il colore del polimero con cui si realizzano i pezzi reali tramite la stampa 3D. Per fare ciò si sono eseguiti varie scansioni di componenti reali aventi varie forme geometriche e varie tonalità di colorazione, al fine di valutare il risultato finale.

Dalla scannerizzazione del salvadanaio mostrato nella figura seguente, emerge come una superficie chiara, regolare e opaca riesca produrre un risultato soddisfacente.

Figura 55. Scannerizzazione 3D salvadanaio.

La scannerizzazione della statuetta egizia ha prodotto un risultato scarso, dato dall'eccessiva riflessione della colorazione.

Figura 56. Scannerizzazione statua egizia.

La statuetta di Agrippa – fornita come tester insieme allo scanner 3D – fornisce un ottimo risultato, dovuto alla natura della colorazione, bianca e opaca.

Figura 57. Scannerizzazione Agrippa.

Dalle prove effettuate, emerge la difficoltà dello scanner Revopoint Mini ad acquisire la nuvola di punti da modelli aventi tonalità scure e troppo riflettenti, per questo motivo tutti i componenti meccanici presi in esame in questo elaborato sono stati tutti stampati in colore bianco opaco.

7. Risultati

In questo capitolo si analizzano nel dettaglio i modelli CAD utilizzati come tester per valutare il comportamento della rete neurale SeMIGCN.

7.1 Flangia forata

Questo pezzo, essendo il primo analizzato, è stato preso come riferimento per trovare i valori ottimali dei parametri della rete neurale che minimizzano il loss. La minimizzazione della funzione di perdita è stata eseguita mediante una serie di test.

Con riferimento alla tabella seguente si definiscono i seguenti parametri:

- Mesh_gt: indica il numero di triangoli presenti nel modello CAD ground truth;
- Mesh_original: indica i triangoli presenti nel file original.obj, proviene dalla scansione 3D e non da una rottura della mesh del ground truth su Blender;
- Tipologia: indica la tipologia del file:
  - CAD fa riferimento al fatto che il modello in esame è stato generato da un software CAD;
  - Real invece indica che il file è il risultato di una scansione 3D. Questo parametro viene settato manualmente nello script;
- Learning rate: indica il tasso di apprendimento per il test in questione;
- N°epoche: parametro temporale proporzionale alla durata totale del singolo test;
- Loss: indica il valore della funzione di perdita al termine del training.

Test	Mesh_gt	Mesh_original	Tipologia	Learning rate	N°epoche	Batch size	k1	k2	kn	Dm_size	Drop rate	loss
Test1	24282	26660	CAD	0,01	100	5	4	4	4	40	0,6	0,898
Test2	24282	26660	CAD	0,01	300	5	4	4	4	40	0,6	0,774
Test3	24282	26660	real	0,01	100	5	4	4	4	40	0,6	0,616
Test4	24282	26660	real	0,01	300	5	4	4	4	40	0,6	0,481
Test5	24282	26660	real	0,008	1000	5	4	4	4	40	0,6	0,472
Test6	24282	26660	real	0,005	1000	5	4	4	4	40	0,6	0,465
Test7	24282	26660	real	0,004	2000	5	4	4	4	40	0,6	0,466
Test8	24282	26660	real	0,001	2000	5	4	4	4	40	0,6	0,590
Test9	24282	26660	real	0,005	2000	5	4	4	4	40	0,6	0,469
Test10	24282	26660	real	0,0035	2000	5	4	4	4	40	0,6	0,475

Test11	38690	53128	real	0,0035	2000	5	4	4	4	40	0,6	0,390

Per poter confrontare i vari test tra di loro, si sono sempre utilizzati gli stessi modelli CAD `{mesh-name}_original.obj` e `{mesh-name}_gt.obj`. Nella tabella la prima riga indica i parametri di default impostati inizialmente dagli autori della rete neurale.

Dai valori di loss ottenuti si possono fare le seguente considerazioni sui parametri di ottimo:

- Il valore di loss si può ridurre attraverso l'aumento del **numero di epoche**, ma un aumento troppo marcato porta ad un incremento della funzione di perdita;
- Come affermato nella fase del training si inserisce la **tipologia del file** a video tramite la compilazione dell'apposita istruzione di lancio. Questa informazione è opportuno indicarla sempre per avere un output a migliore qualità. Dai risultati si nota che a parità del numero di epoche e di learning rate, impostando il parametro `real`, si ottiene un valore di loss inferiore rispetto all'utilizzo di `CAD`.
- Discorso più articolato invece per il settaggio del **learning rate**. Dai capitoli precedenti, si ricorda che per minimizzare la funzione di perdita, occorre ridurre il learning rate. Tuttavia, da sola questa variazione non risulta sufficiente, in quanto riducendo la variazione con cui vengono aggiornati i pesi, per raggiungere l'obiettivo preposto, si richiede più tempo di elaborazione, che si traduce nella necessità di aumentare il numero di epoche. Anche in questo caso però non aumentare troppo il numero di epoche porta ad un nuovo incremento del valore di loss.
- L'ultimo test (Test11) è stato eseguito in maniera più grafica, aumentando il **numero dei triangoli** della mesh original di input. Si nota che rispetto ai casi precedenti, il valore di loss risulta il più basso, nonostante si è incorsi nelle problematiche dei punti precedenti. Ne consegue che il campionamento della mesh poligonale del file original impatta maggiormente rispetto ai parametri della rete neurale modificati a codice. Il limite superiore all'eccessivo infittimento della mesh è dato in primo luogo dalla scheda grafica disponibile e in secondo luogo dal tempo maggiore di elaborazione anche per la fase di preprocess.

I restanti valori della tabella (batch size, k_1, k_2, k_n, Dm_size, Drop_rate) non sono stati modificati per via del loro ruolo tecnico nel funzionamento della rete, ritenendo accettabili i valori impostati dagli autori di SeMIGCN.

Per i successivi test dei modelli CAD, si è ritenuto un buon compromesso, in termini di minimizzazione della funzione di perdita e di tempistiche, i seguenti valori:

Parametro	Valore
Tipologia	real
N°epoche	1000
Learning rate	0.005

7.1.1 Flangia forata ground truth

In figura viene mostrato il modello CAD nominato `flangia_forata_gt.obj`, come prevede la designazione vista precedentemente.

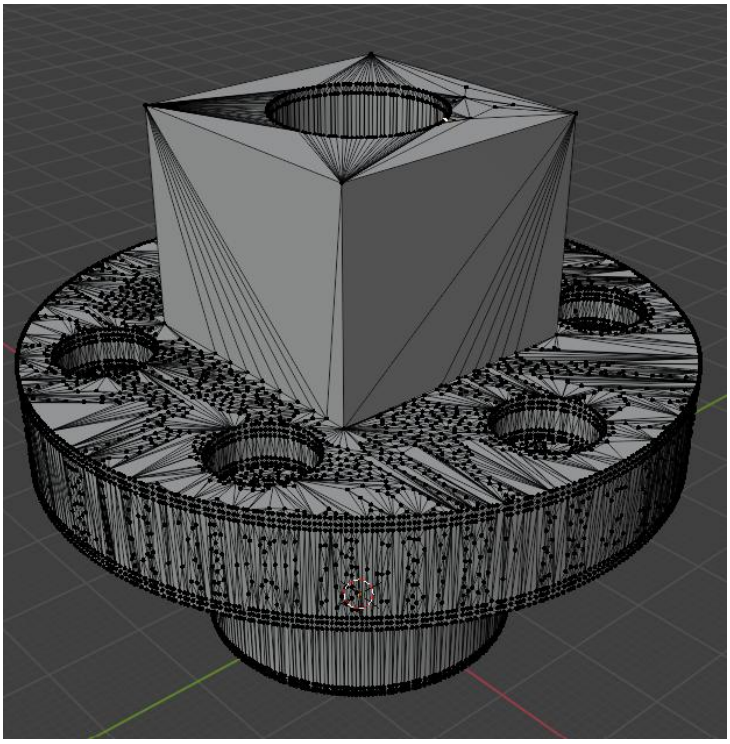


Figura 58. Groud truth.

Caratteristiche della mesh poligonale:

- Numero vertici: 12129;
- Numero spigoli: 36423;
- Numero facce: 24282.

Questo disegno CAD rappresenta il modello ideale a cui la rete neurale deve tendere. Per la sua modellazione ed esportazione si è utilizzato il software Creo 11. Proprio per il modo in cui è stato generato il componente, tutte le parti che lo compongono sono ben definite senza presenza di difetti.

È importante tenere conto, in questo e nei seguenti casi studio, come le caratteristiche della mesh poligonale varino da componente a componente, pur mantenendo un buon grado di precisione. I valori impostati su Creo Parametric sono i medesimi per ogni

pezzo, ma il numero di vertici, spigoli e facce cambiano a seconda della geometria del pezzo.

7.1.2 Flangia forata original

I modelli CAD dei componenti nominati secondo questa designazione, si generano a partire da due processi di elaborazione:

1. Rottura manuale della mesh poligonale:

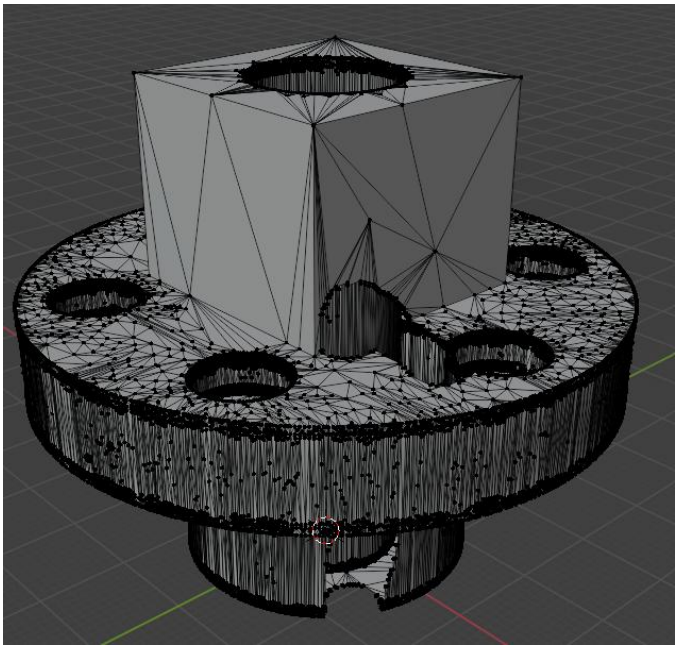


Figura 59. Original mesh poligonale.

Caratteristiche della mesh poligonale:

- Numero vertici: 28494;
- Numero spigoli: 85031;
- Numero facce: 56523.

Questo file CAD è stato generato a partire dal ground truth per effettuare delle piccole spaccature nel modello. La mesh iniziale risulta troppo grossolana, per questo motivo è stato effettuato un remeshing per infittire il campionamento, sono stati aumentati i vertici per creare dei buchi nel reticolo e successivamente è stata applicata una decimazione.

2. Mesh ricostruita tramite scansione 3D:

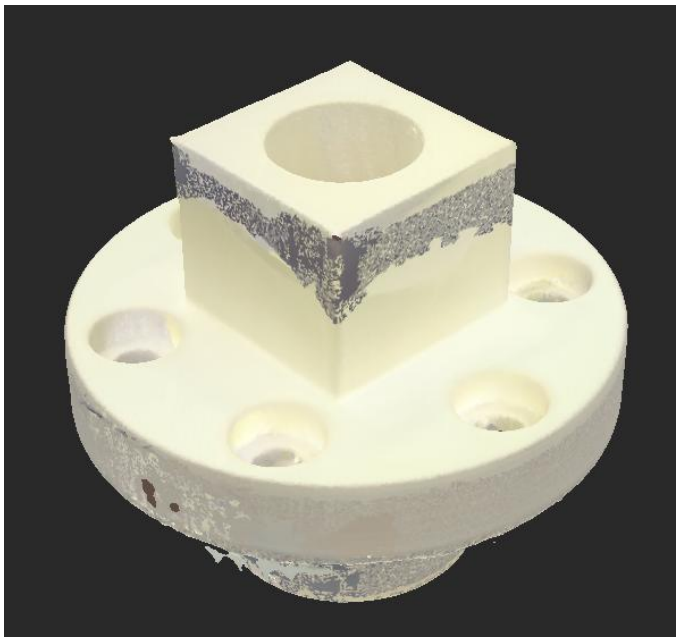


Figura 60. Original scanner 3D.

Il modello mostrato in figura è il risultato di una scansione della flangia forata stampata in 3D. Sulla superficie si nota la presenza dei difetti nelle aree in cui il reticolo risulta più fitto. Per questo, vengono eseguite due scansioni parziali per migliorare la qualità del risultato, come mostrano nella figura seguente.

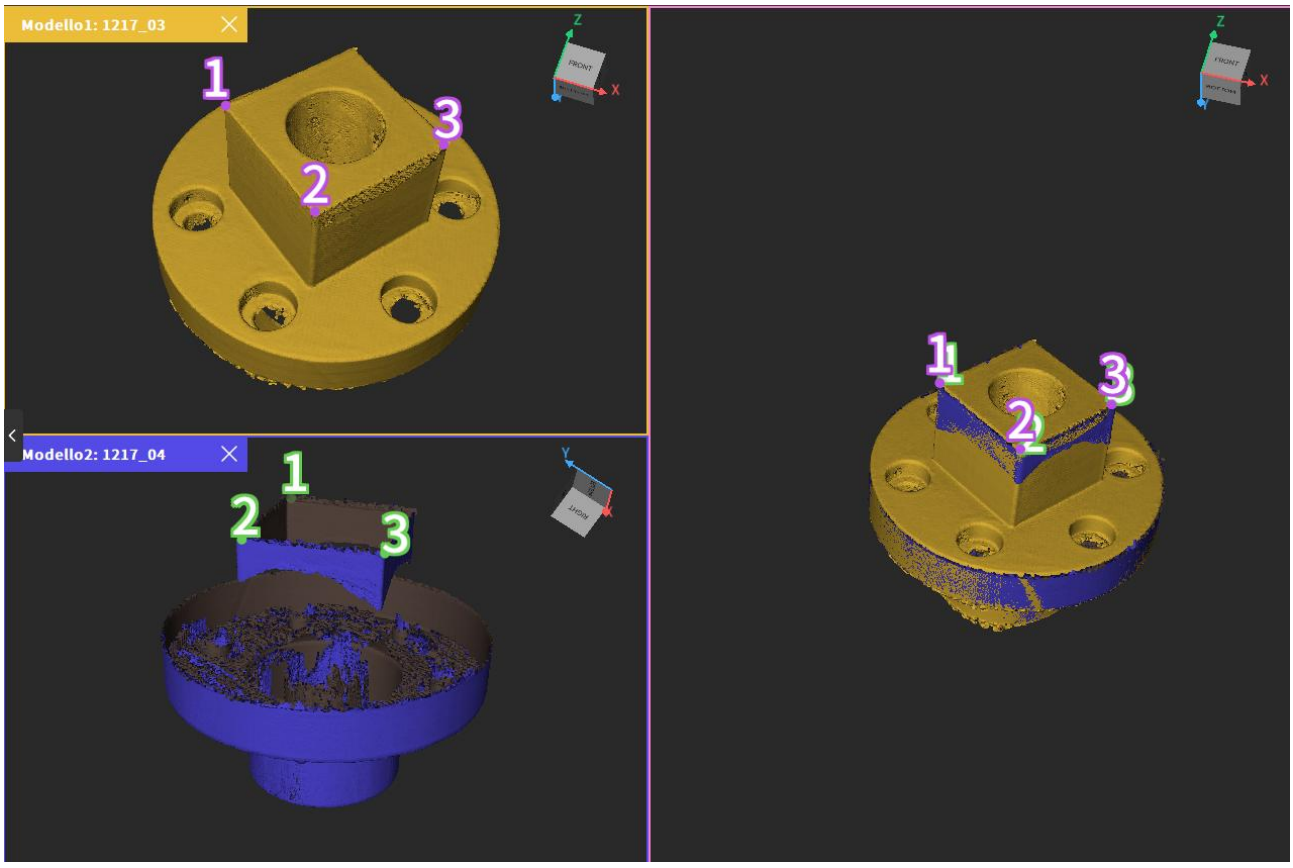


Figura 61. Unione delle scannerizzazioni parziali.

Successivamente, il pezzo è stato importato su Blender dove è stata eseguita un'operazione di remeshing e decimazione per cercare di ottenere una migliore qualità dell'input alla rete. Di seguito viene mostrato il risultato finale della manipolazione su Blender:

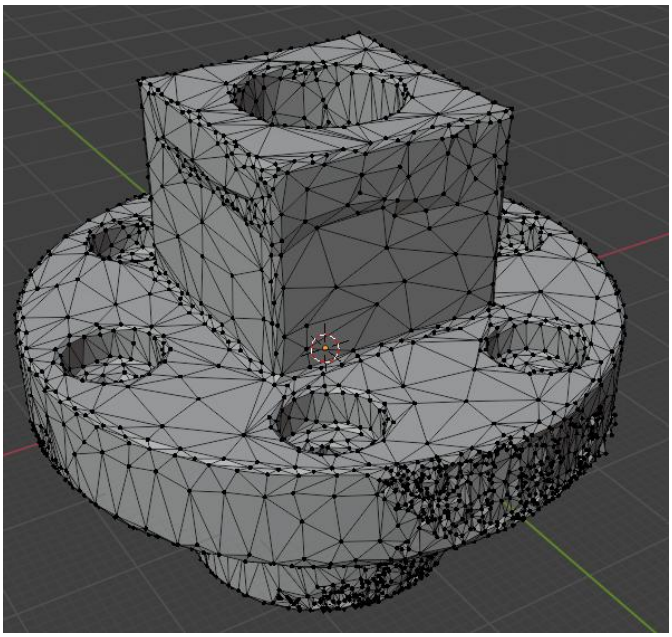


Figura 62. Remesh finale.

Caratteristiche della mesh poligonale:

- Numero vertici: 11365;
- Numero spigoli: 38522;
- Numero facce: 26660.

7.1.3 Risultati

Come affermato in precedenza, la fase di training per entrambi i file original è stata eseguita con la rete neurale sgcn.py. I valori della loss sono stati salvati ogni 10 epoche e immagazzinati in un file .txt per poi essere graficati in funzione delle epoche.

Di seguito si riportano i risultati finali a fine elaborazione, presentando il modello CAD corrispondente all'epoca 1000:

1. Rottura manuale della mesh poligonale:

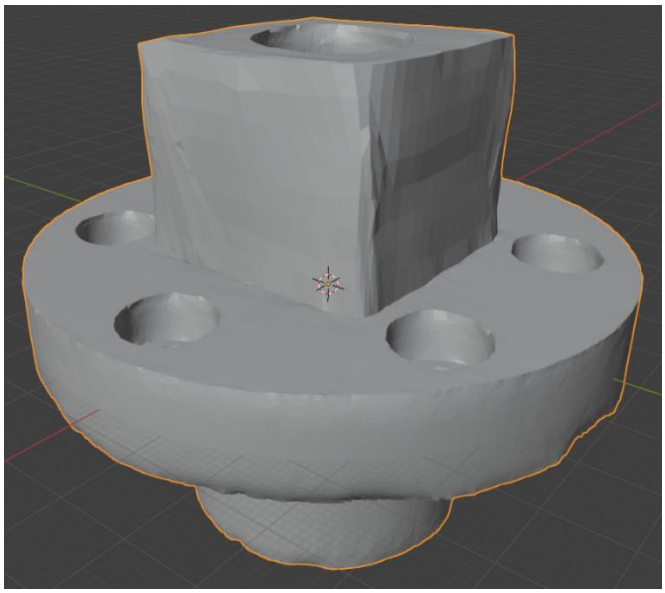


Figura 63. Output rottura manuale.

Caratteristiche della mesh poligonale:

- Numero vertici: 61607;
- Numero spigoli: 184857;
- Numero facce: 123236.

Si nota in questo caso una deformazione della sezione rettangolare, dovuta alla mancanza di un'ampia porzione di reticolo. Tuttavia la sezione circolare in basso non ha presentato questo difetto, nonostante la sezione mancante fosse similare. Da qui si può intuire la difficoltà della rete a ricostruire spigoli vivi.

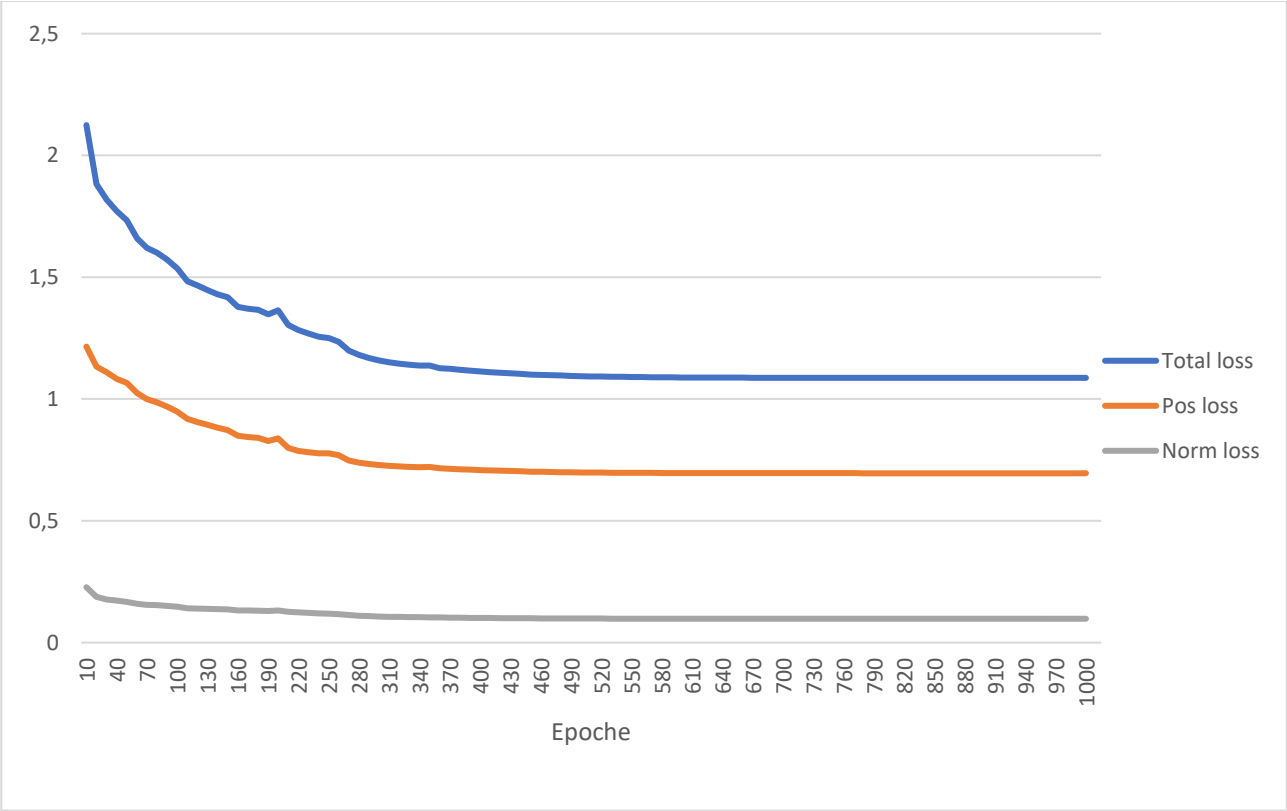


Figura 64. Andamento loss.

2. Mesh ricostruita tramite scansione 3D:

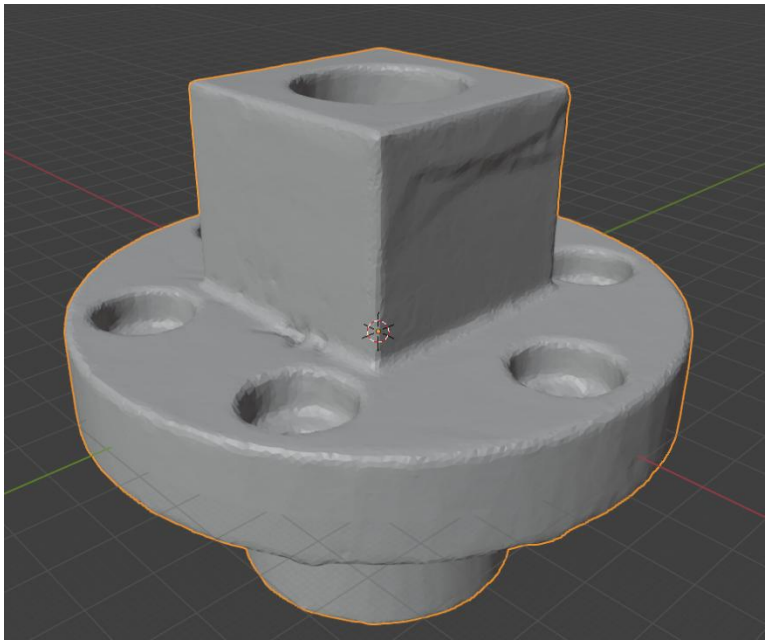


Figura 65. Output scannerizzazione.

Caratteristiche della mesh poligonale:

- Numero vertici: 65270;
- Numero spigoli: 195846;
- Numero facce: 130564.

Si è ottenuto un buon risultato, nonostante i difetti della scansione.

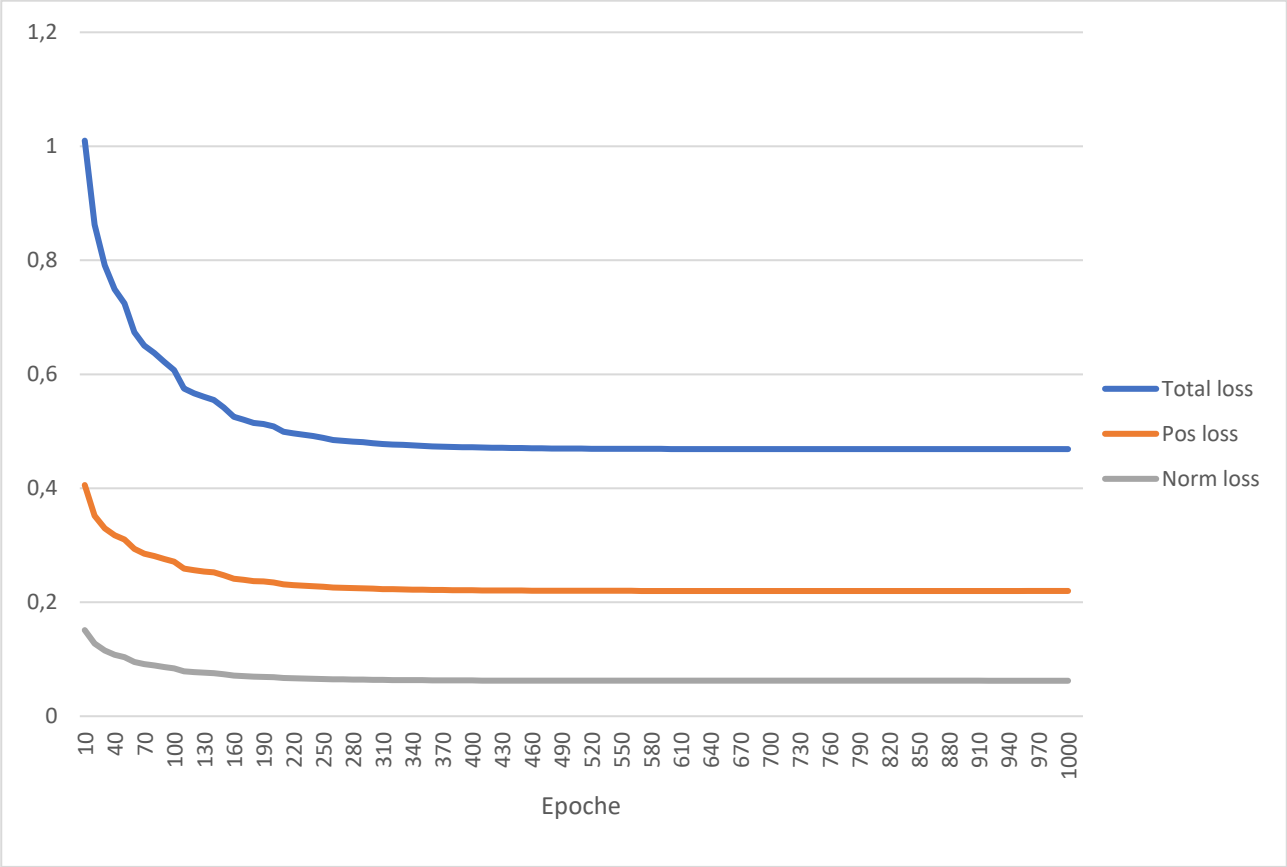


Figura 66. Andamento loss.

7.1.4 Problematiche riscontrate

Si nota dalla deformazione del pezzo e dai valori riportati nei grafici, che la rete neurale presenta un valore di loss più elevato nel caso di mesh incompleta quando questa viene realizzata manualmente eliminando i vertici dal reticolo poligonale.

Per verificare se questa problematica risulta una limitazione effettiva della rete neurale SeMIGCN, si rende necessario eseguire un nuovo test nel quale le aree mancanti risultano più contenute rispetto al caso precedente.

Di seguito si riporta la nuova rottura della mesh creata sempre utilizzando il metodo usato precedentemente:

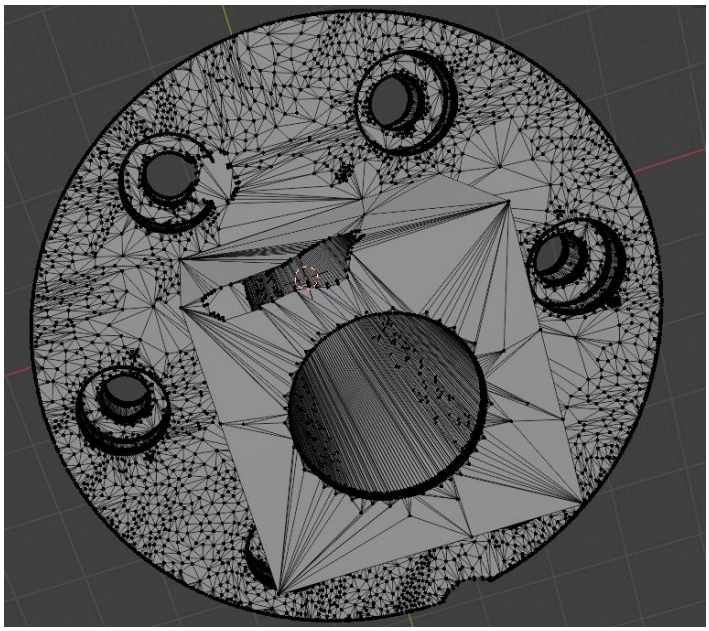


Figura 67. Rottura remesh.

Caratteristiche della mesh poligonale:

- Numero vertici: 35594;
- Numero spigoli: 106279;
- Numero facce: 70669.

Al termine del trainig si ottiene il seguente risultato:

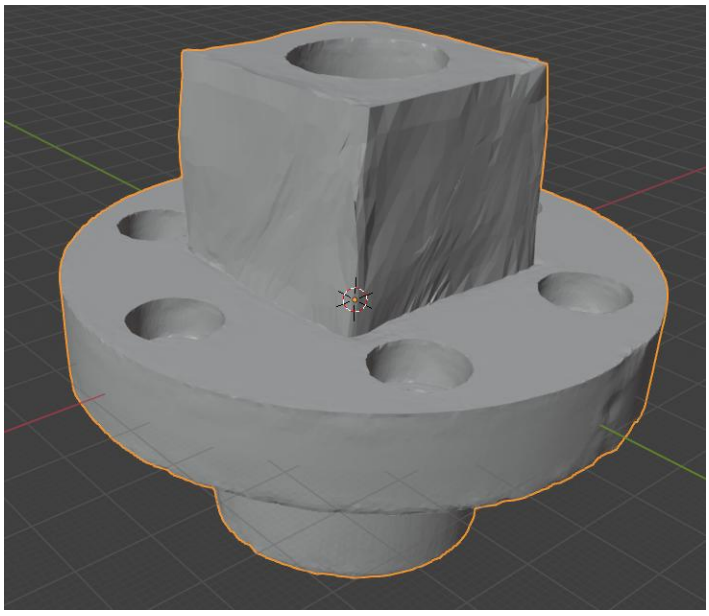


Figura 68. Output rottura manuale remesh.

Caratteristiche della mesh poligonale:

- Numero vertici: 62295;
- Numero spigoli: 186921;

- Numero facce: 124614.

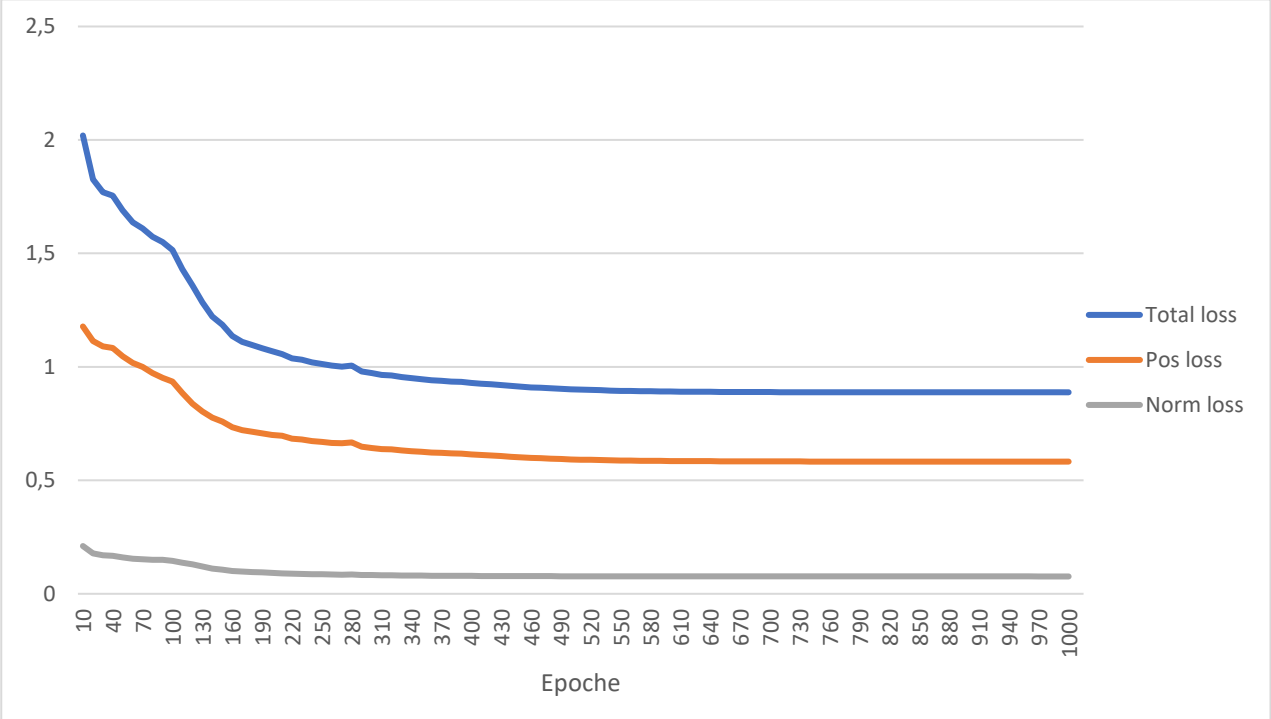


Figura 69. Andamento loss remesh.

Con la modifica apportata al reticolo, si riscontra un notevole miglioramento visivo, in quanto la forma del pezzo risulta meno distorta, in particolare per quanto riguarda la parte a sezione quadrata. Inoltre l'effetto della modifica si può osservare anche a nel grafico, dove il valore finale di loss si porta sotto l'unità.

7.1.5 Valutazioni

La problematica riscontrata nella rottura manuale, mette in evidenza quali possono essere le limitazioni nell'utilizzo di SeMIGCN.

Per quanto riguarda la performance della rete neurale nel caso in cui la mesh di input proviene dalla scansione 3D, si può constatare un'efficienza maggiore, in quanto il reticolo poligonale risulta molto meno danneggiato, presentando difetti superficiali ma non mancanze totali della mesh. Tuttavia nel caso in cui si creino manualmente dei buchi, la discontinuità del reticolo deve essere di entità ridotta e se presente negli spigoli vivi, si ha un incremento nella difficoltà di riparazione.

Inoltre, dall'andamento riportato dai grafici si può notare il plateau della loss dopo un certo numero di epoche corrispondente al basso valore di learning rate.

Ora, al fine di poter effettuare un confronto tra i risultati ottenuti, è necessario applicare un allineamento delle geometrie, così che si possano sovrapporre e una scalatura, per

poter confrontare componenti delle medesime dimensioni. Si può effettuare il tutto tramite Blender, rinominando per semplicità di confronto la seguente nomeclatura:

- {mesh-name}_A.obj → ground truth, file di progetto;
- {mesh-name}_B.obj → original scan, output dello scanner 3D;
- {mesh-name}_C.obj → output rottura tramite Blender, in uscita dalla rete neurale;
- {mesh-name}_D.obj → output scansione, in uscita dalla rete neurale.

Questa procedura è stata applicata in tutti i casi studio.

Il confronto viene eseguito tramite il software MeshLab, al fine di determinare la differenza tra il groud truth e i risultati ottenuti dalla scansione e dall'elaborazione fatta dalla rete neurale.

Tale analisi tiene conto dei parametri forniti da MeshLab, quali:

- Distanza minima: si intende la distanza minore tra le due mesh a confronto, nel caso in cui sia nulla, significa che le due elaborazioni coincidono;
- Distanza massima: indica la maggiore distanza tra le mesh a confronto;
- Distanza media: rappresenta la media di tutte le distanze rilevate tra il modello target e quello a confronto;
- RMS: è la distanza media quadratica tra le due mesh.

Nel caso della flangia forata, si riportano i seguenti valori rilevati dai confronti effettuati.

Confronto	Distanza minima [mm]	Distanza massima [mm]	Distanza media [mm]	RMS [mm]
A-C	0,719752	0,719752	-0,050124	0,217703
A-D	1,041928	1,041928	-0,129564	0,267796
B-D	0,565954	0,565954	0,026703	0,181888
A-B	1,102779	1,102779	-0,388068	0,599976

È importante notare come i valori di stanza minima e massima coincidano. Questo è dovuto al fatto che i modelli sono stati allineati perfettamente e il campionamento del reticolo risulta uniforme sulle mesh a confronto.

1) Caso AC: si ha una buona sovrapposizione tra le mesh in esame. La rete ha ricostruito correttamente i fori ma sono presenti problemi relativi alla parte prismatica. In figura è presente il componente C.

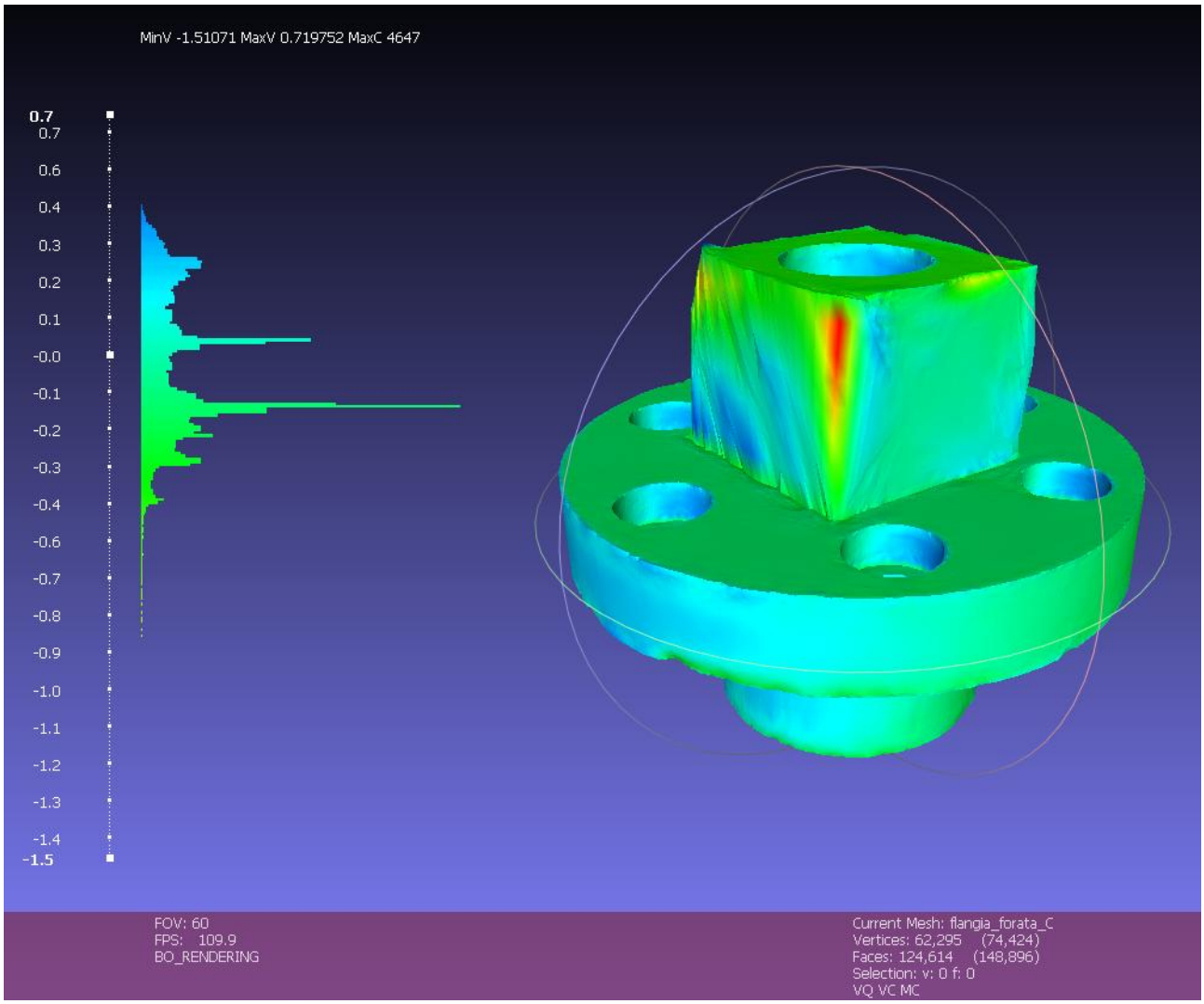


Figura 70. Confronto AC.

2) Caso AD: la rete ha ricostruito con precisione il modello proveniente dalla scannerizzazione. Le discrepante sono minime. L'alta qualità della scansione effettuata ha influito positivamente sul risultato finale ottenuto dall'elaborazione della rete. La superficie della corona circolata che risultava frastagliata è stata ricostruita in maniera soddisfacente. In figura è presente il componente D.

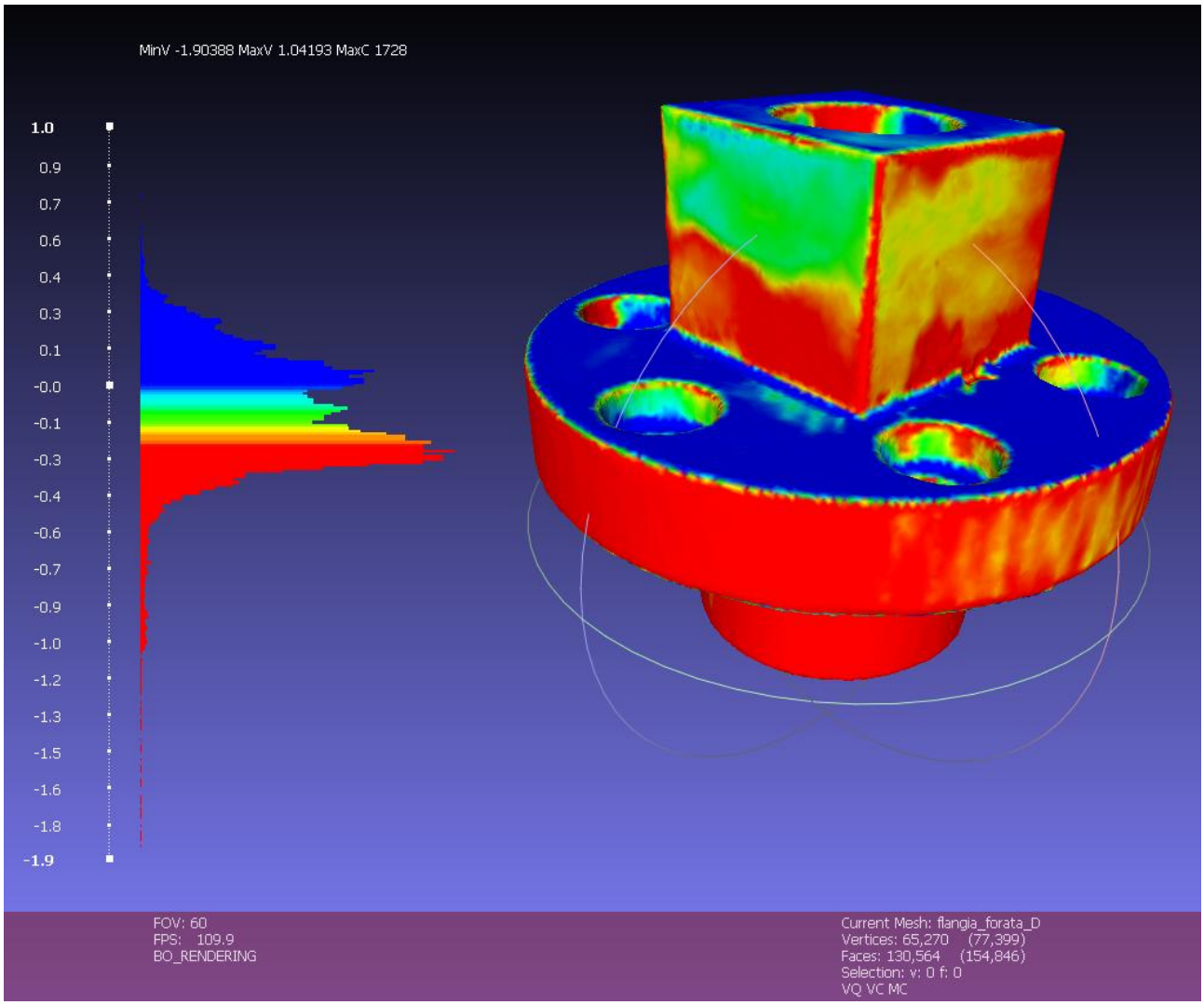


Figura 71. Confronto AD.

3) Caso BD: la rete durante l'elaborazione è in grado di mantenere quasi inalterate le dimensioni, e sono presenti differenze minime. Questo significa che SeMIGCN non altera le dimensioni dei componenti in ingresso, ma si limita a ricostruirli lì dove necessario. In figura è presente il componente D.

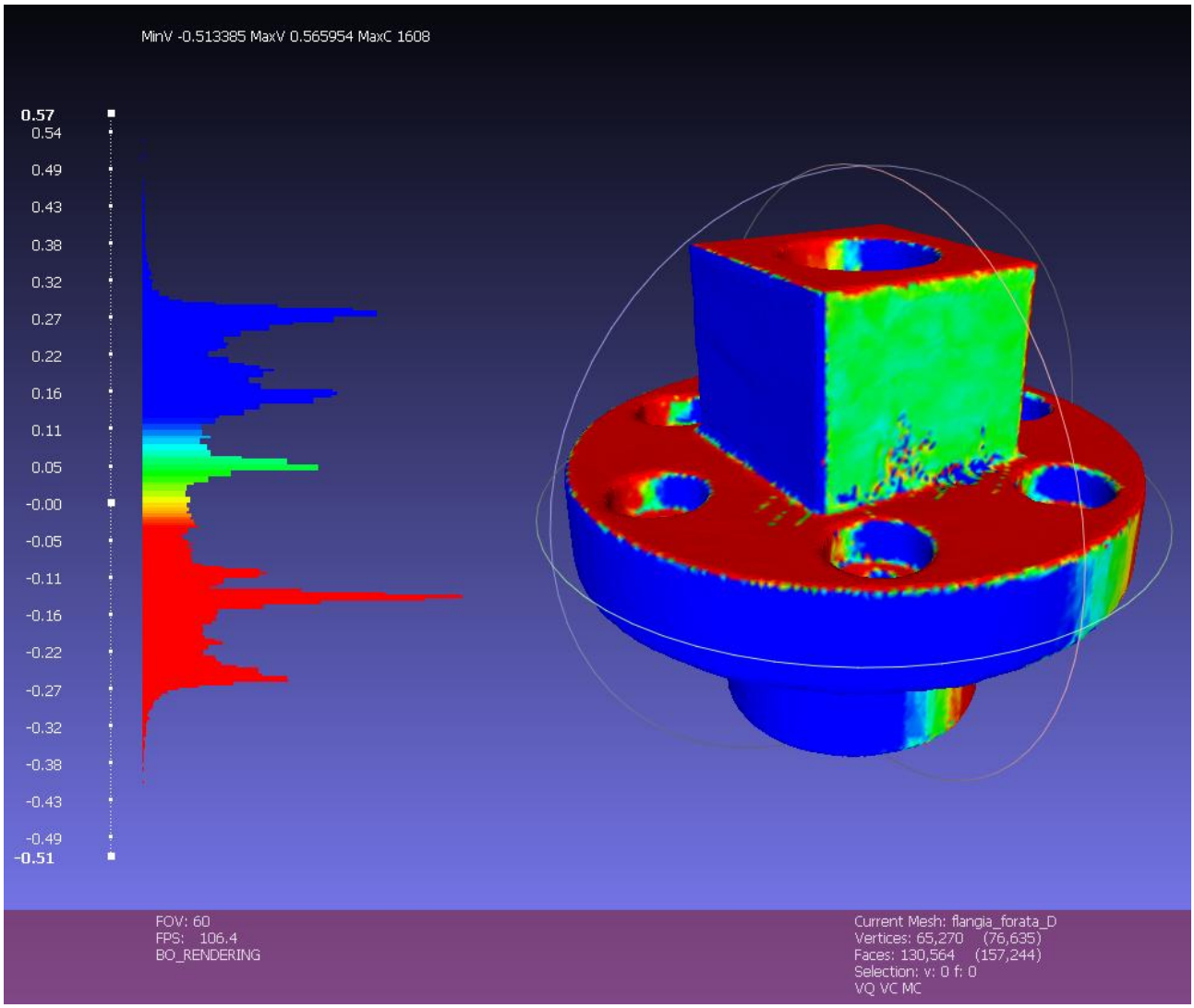


Figura 72. Confronto BD.

4) Caso AB: è evidente la presenza di una parte più rovinata nella superficie cilindrica, dovuta alla scannerizzazione. Si tratta di differenze minime. In figura è presente il componente B.

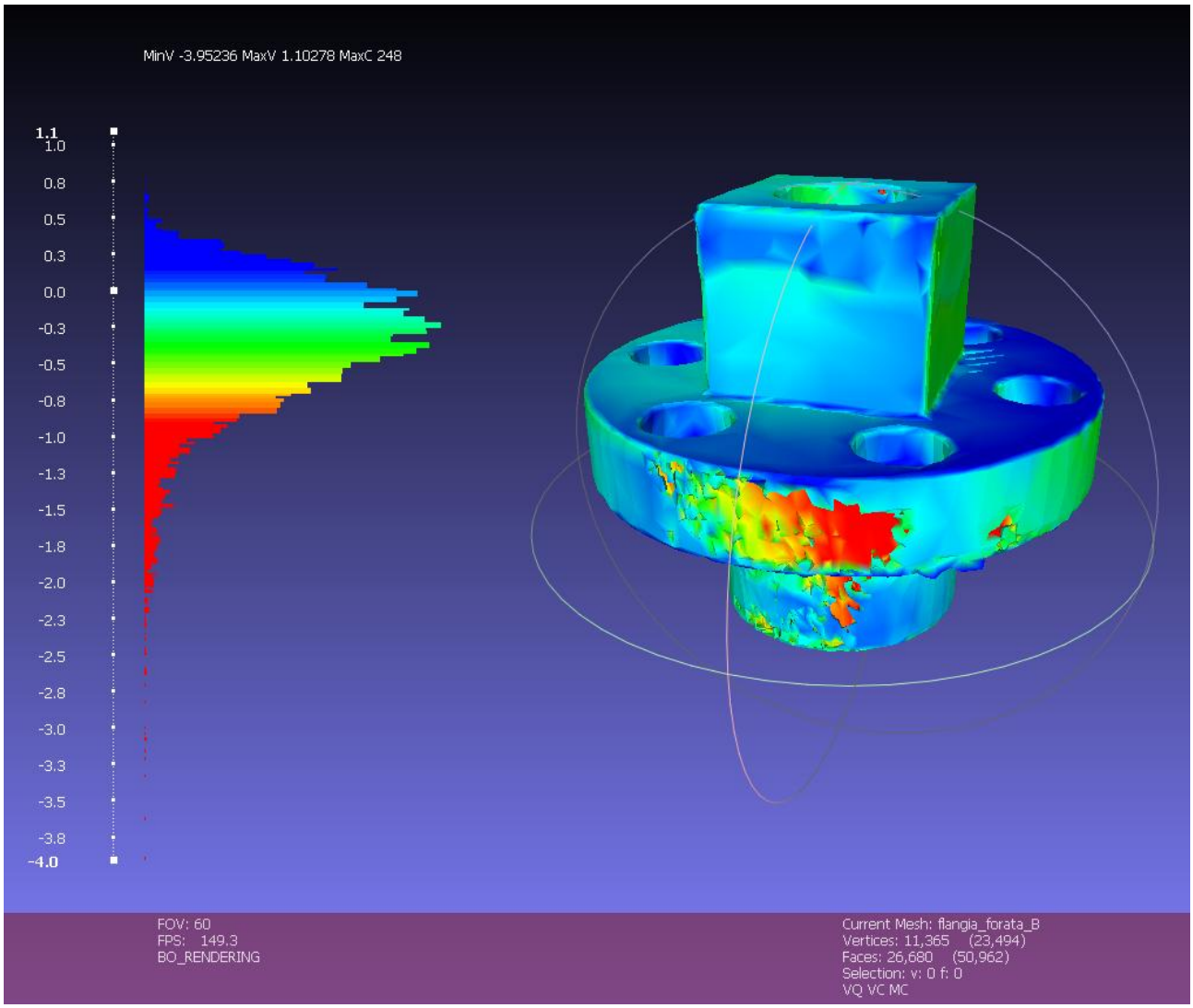


Figura 73. Confronto AB.

7.2 Supporto saldatura

Per questo modello CAD, come per il caso precedente, si effettuano vari test dando in pasto alla rete file differenti però utilizzando sempre gli stessi parametri per permettere così un più facile confronto e valutazione della rete neurale.

Al fine di verificare se variando la geometria cambiano anche i parametri di ottimo della rete neurale, si effettua un ulteriore fase di testing. I parametri sono riportati nella tabella seguente.

Test	Mesh_gt	Mesh_original	Tipologia	Learning rate	N°epoche	Batch size	k1	k2	kn	Dm_size	Drop rate	loss
Test1	56486	56991	real	0,005	100	5	4	4	4	40	0,6	0,444

Test2	56486	56991	real	0,005	300	5	4	4	4	40	0,6	0,356
Test3	56486	56991	real	0,005	1000	5	4	4	4	40	0,6	0,346
Test4	56486	56991	real	0,005	1000	5	4	4	4	40	0,6	0,35
Test5	56486	56991	real	0,005	1000	5	4	4	4	40	0,6	0,3528

I test per trovare i parametri della rete neurale che minimizzano la funzione di perdita, sono stati effettuati sulla base delle conoscenze acquisite dal modello CAD precedente. Si fornisce in input alla rete la stessa mesh poligonale, ma con un campionamento più fitto, ma rimangono inalterati invece la tipologia e il learning rate.

Il test è stato ripetuto aumentando progressivamente il numero di epoche. Confrontando i valori di loss ottenuti, questi risultano inferiori rispetto a quelli ottenuti nei test del pezzo precedente; tuttavia, i parametri di ottimo trovati in precedenza risultano attendibili. Da questo risultato si può affermare che i parametri utilizzati nella rete sono indipendenti dalla geometria del componente in esame. Si ha una variazione rispetto al valore minimo dato da questi valori solo variando il campionamento della mesh di input.

7.2.1 Supporto saldatura ground truth

In figura viene mostrato il modello CAD nominato `supporto_saldatura_gt.obj` come prevede la designazione vista precedentemente.

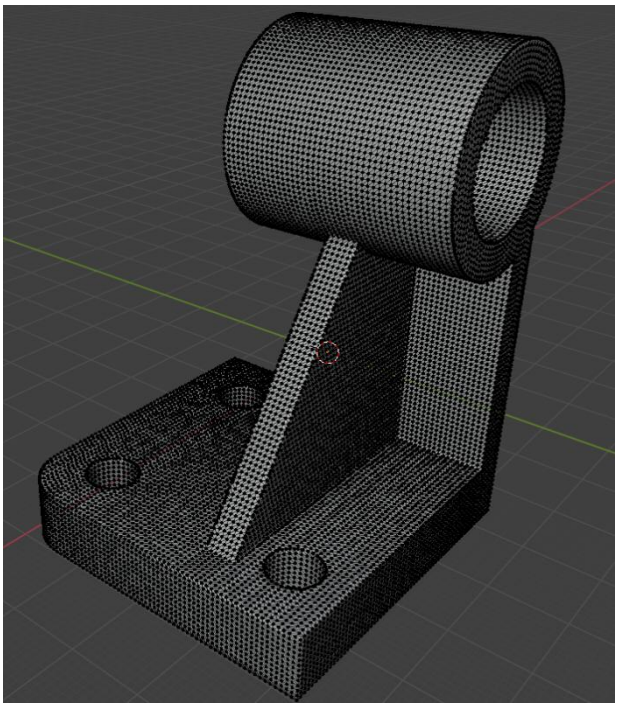


Figura 74. Ground truth.

Caratteristiche della mesh poligonale:

- Numero vertici: 28237;
- Numero spigoli: 84729;
- Numero facce: 56486.

Questo modello CAD rappresenta il target dell'addestramento della rete neurale, presentando una mesh generata tramite il software Creo 11 per la successiva esportazione in .stl. Sulla base dei risultati precedenti, per avere un buon compromesso tra consumo di memoria e precisione del modello, si è scelto come campionamento della mesh introno ai 50000 triangoli.

7.2.2 Supporto saldatura original

I modelli CAD presentati in questa sezione, rappresentano le mesh di input per la rete neurale e vengono denominati secondo la designazione `{mesh-name}_original.obj`.

1. Rottura manuale della mesh poligonale:

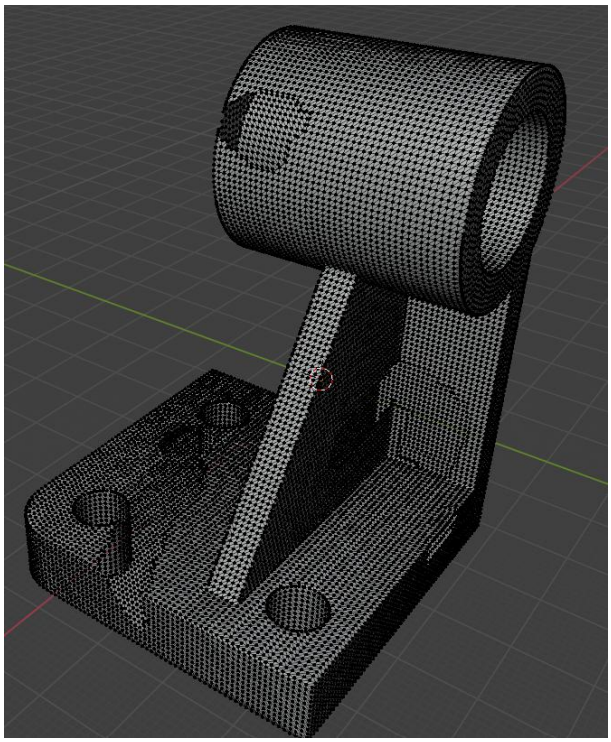


Figura 75. Original mesh poligonale.

Caratteristiche della mesh poligonale:

- Numero vertici: 26682;
- Numero spigoli: 79541;
- Numero facce: 52843.

Come per la `flangia_forata`, anche il `supporto_saldatura` si ottiene a partire dalla rottura casuale di alcune parti del reticolo. In figura si osservano le rotture apportate, le quali interessano sia superfici curve, sia piane intersecanti fori. In questo caso il più alto campionamento, permette di evitare un'ulteriore fase di remeshing per eliminare il reticolo in maniera più precisa.

2. Scansione 3D:

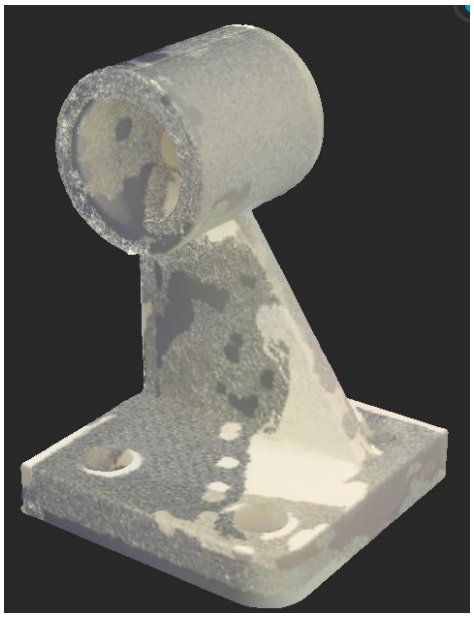


Figura 76. Original scanner 3D.

La figura mostra il risultato della scansione 3D tramite lo scanner Revopoint Mini. L'acquisizione della nuvola dei punti, date le dimensioni e la forma del modello, ha richiesto più scansioni, specialmente per evidenziare meglio gli spigoli che rimangono nascosti. Una volta ultimate le scansioni parziali, il software Revoscan 5 permette l'unione di più componenti tramite l'allineamento di punti in comune tra i modelli da unire, come mostra la seguente figura:

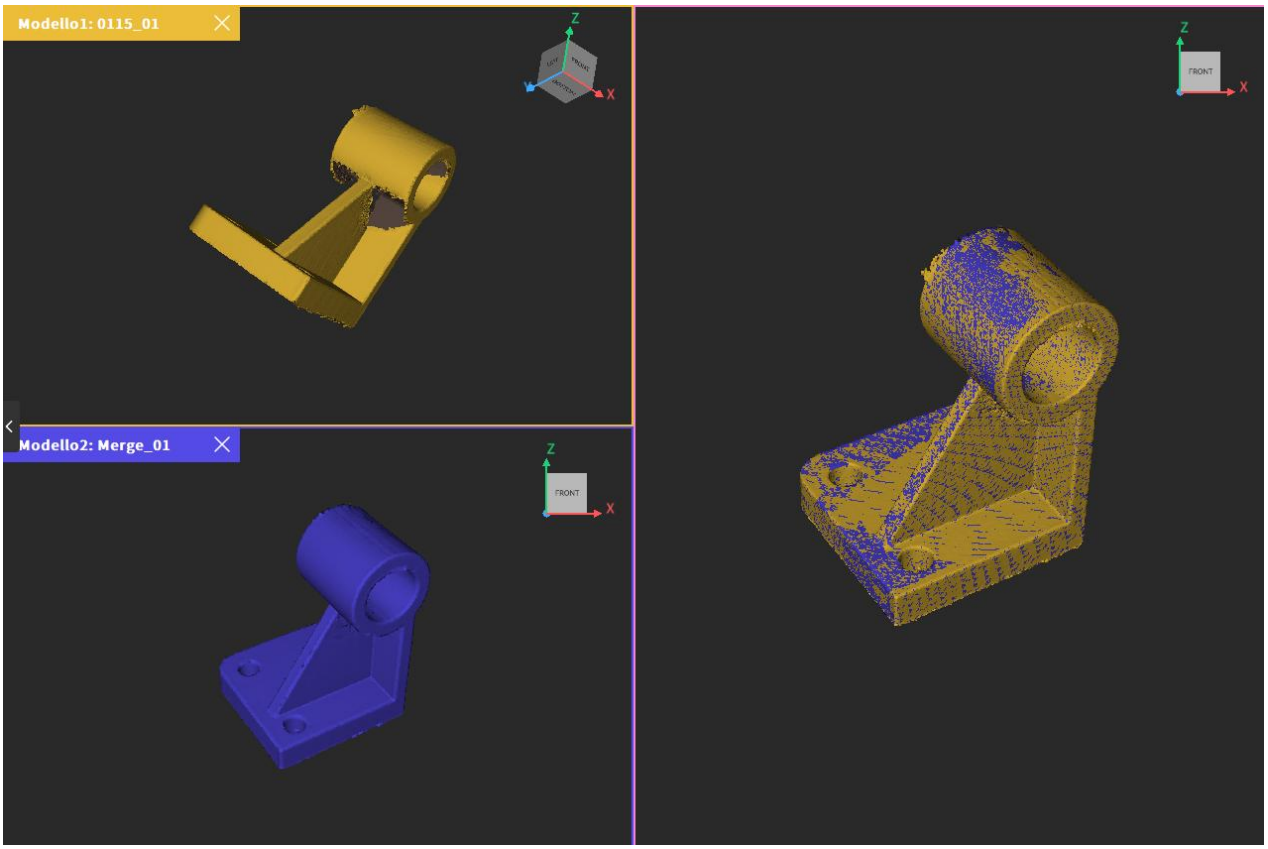


Figura 77. Unione delle scannerizzazioni parziali.

Il risultato finale viene importato su Blender dove vengono applicate le successive operazioni di remeshing e decimazione, per ottenere il risultato finale:

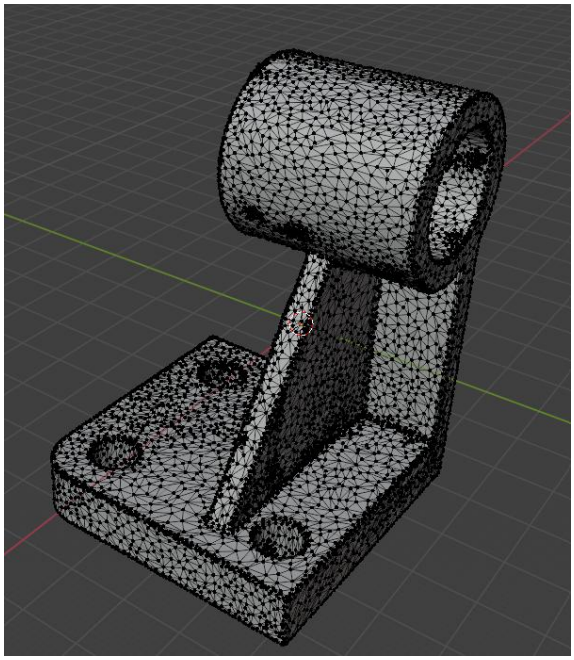


Figura 78. Remesh finale.

Caratteristiche della mesh poligonale:

- Numero vertici: 27346;
- Numero spigoli: 84713;
- Numero facce: 56991.

Si può osservare anche in questo caso alcune aree in cui il reticolo mesh risulta più fitto a causa della superficie che si presenta frastagliata.

7.2.3 Risultati

Di seguito si riportano i risultati finali presentando il modello CAD corrispondente all'epoca 1000:

1. Rottura manuale della mesh:

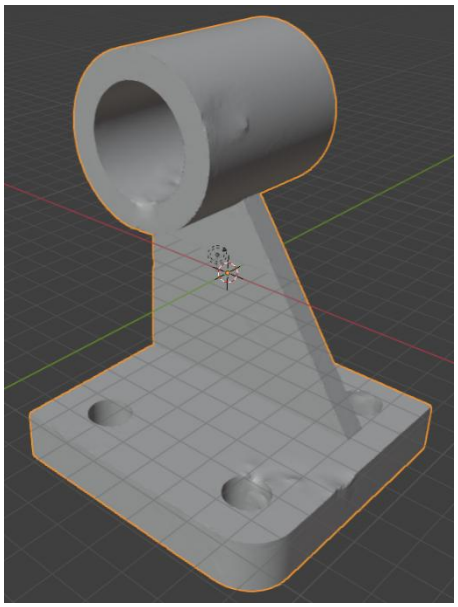


Figura 79. Output rottura manuale.

Caratteristiche della mesh poligonale:

- Numero vertici: 89689;
- Numero spigoli: 269085;
- Numero facce: 179390.

Nonostante l'entità della rottura, la rete ha ricostruito la forma senza eccesive deformazioni degli spigoli, riuscendo a ricostruire persino i bordi dei fori interni.

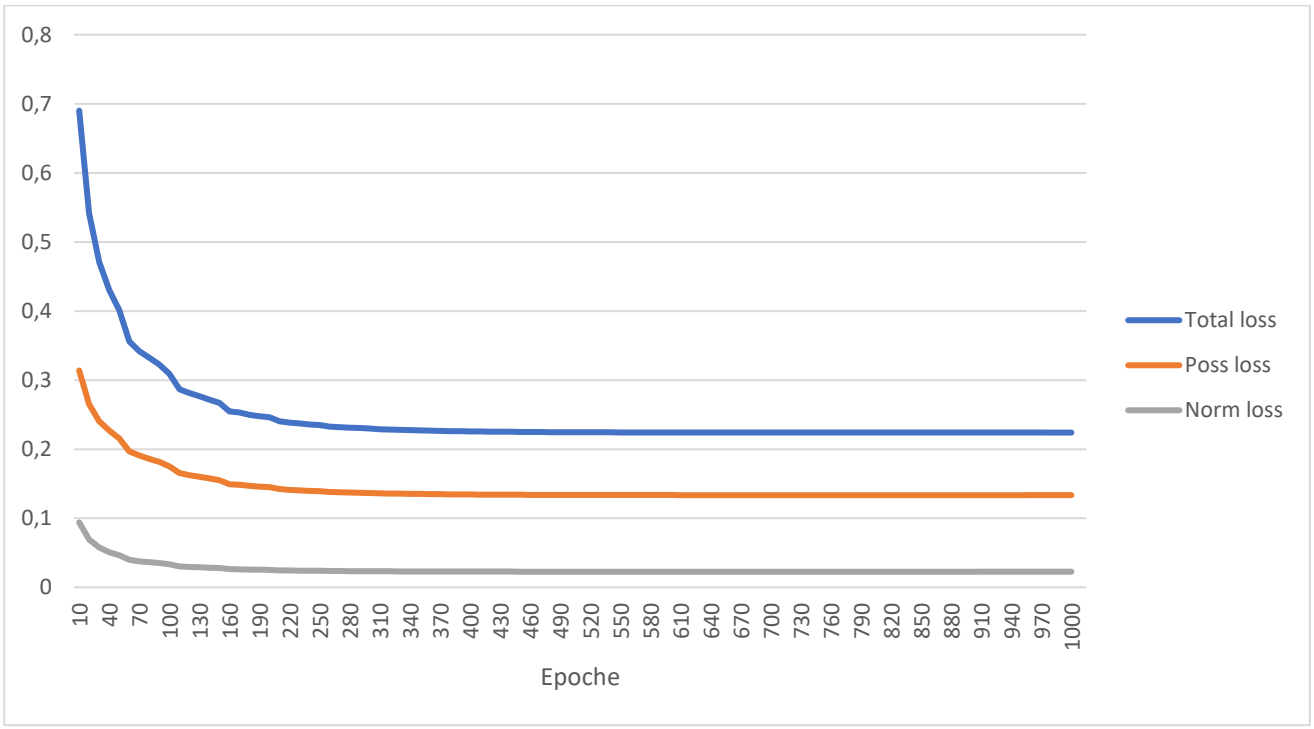


Figura 80. Andamento loss.

2. Mesh ricostruita tramite scansione 3D:

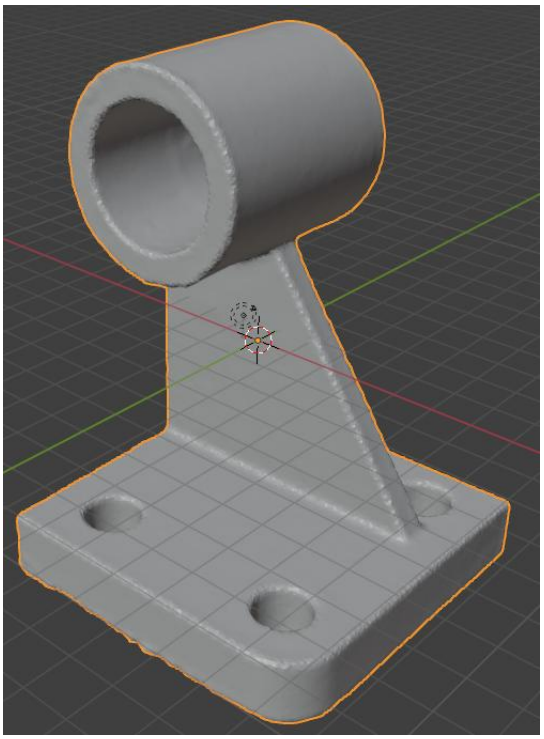


Figura 81. Output scannerizzazione.

Caratteristiche della mesh poligonale:

- Numero vertici: 66869;
- Numero spigoli: 200625;
- Numero facce: 133750.

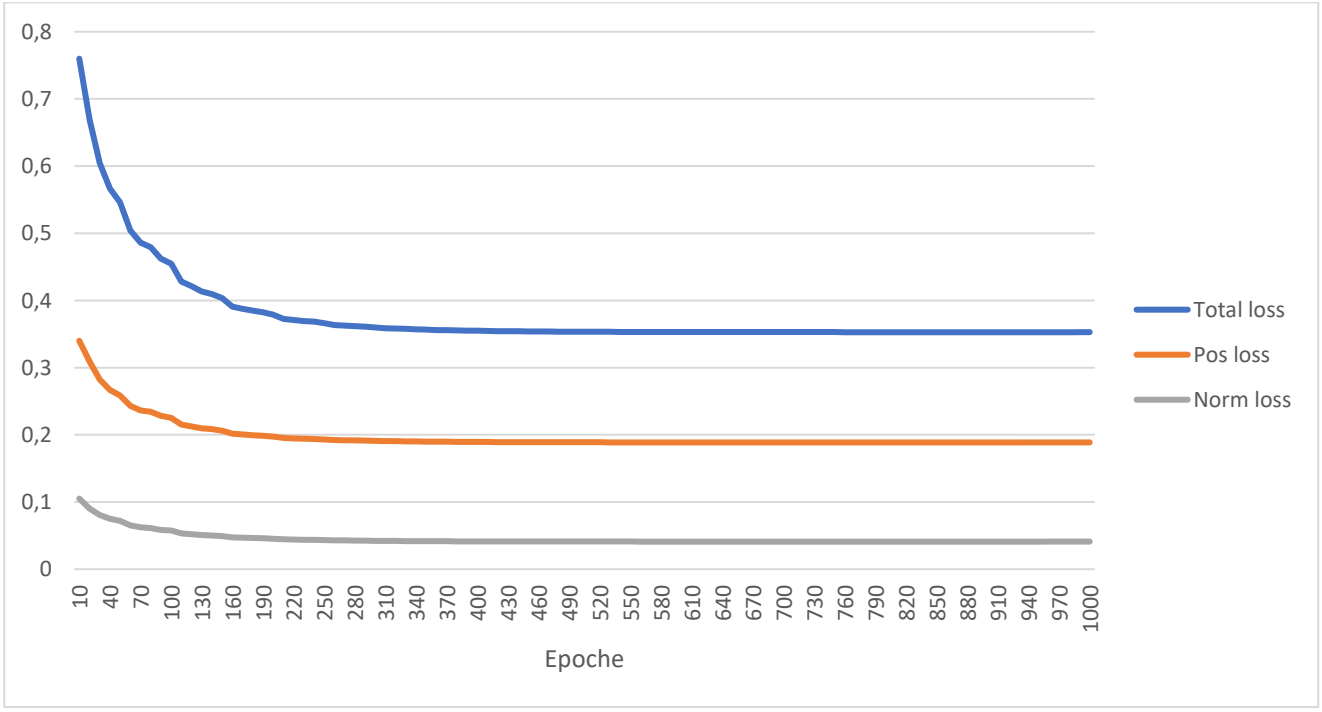


Figura 82. Andamento loss.

7.1.4 Valutazioni

Dal risultato dei test effettuati nella prima parte emerge che i parametri di ottimo della rete neurale rimango sempre gli stessi al variare della geometria del pezzo, tuttavia risulta possibile variare questo valore attraverso un diverso campionamento della superficie del pezzo.

Nel caso del pezzo in esame, si riscontra una buona ricostruzione nella sezione cilindrica e una lieve deformazione negli spigoli di contorno del pezzo e in corrispondenza dei bordi interni. Questa maggiore discontinuità di ricostruzione è dovuta in primo luogo a un'ampia area mancante e in secondo luogo all'aver tolto una parte del reticolo nei bordi e nella superficie interna dei fori.

In aggiunta, come riscontrato nel caso precedente l'andamento del valore della funzione di perdita presenta un plateau, sempre per il valore basso settato nel learning rate.

Di seguito si riportano i dati relativi al confronto effettuato tramite MeshLab.

Confronto	Distanza minima [mm]	Distanza massima [mm]	Distanza media [mm]	RMS [mm]
A-C	0,838667	0,838667	-0,033695	0,231148
A-D	2,397143	2,397143	0,015967	0,569188
B-D	4,648316	4,648316	0,158696	0,371793
A-B	2,490636	2,490636	-0,652111	1,099378

1) Caso AC: nelle sezioni ricostruite dalla rete si hanno discostamenti maggiori. La ricostruzione però è soddisfacente e le distanze minime.

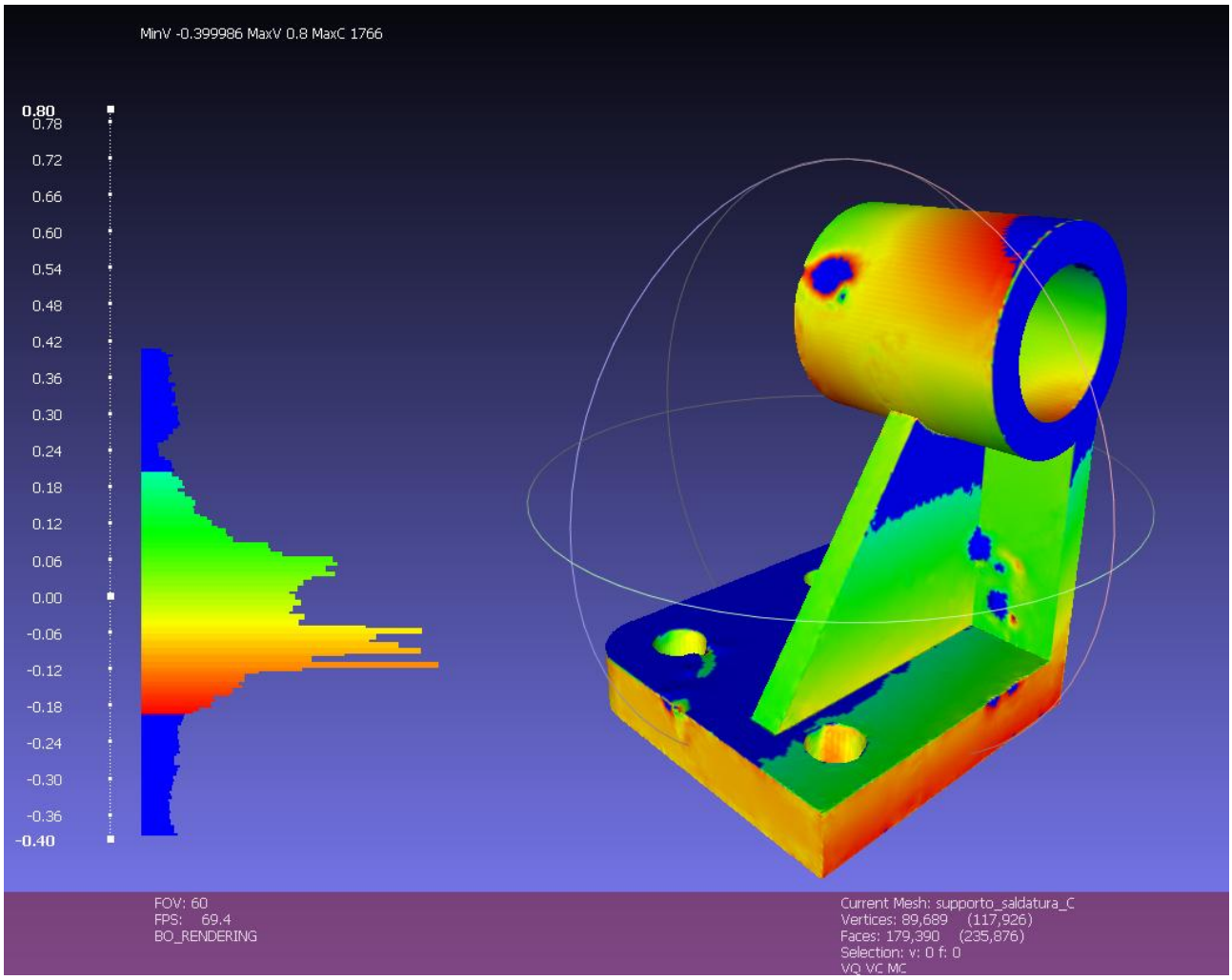


Figura 83. Confronto AC.

2) Caso AD: nel confronto tra ground truth e l'output della rete neurale si ha una buona ricostruzione. Gli scostamenti trai due modelli sono compatibili con la precisione fornita dallo scanner. Si ha quindi una perfetta ricostruzione.

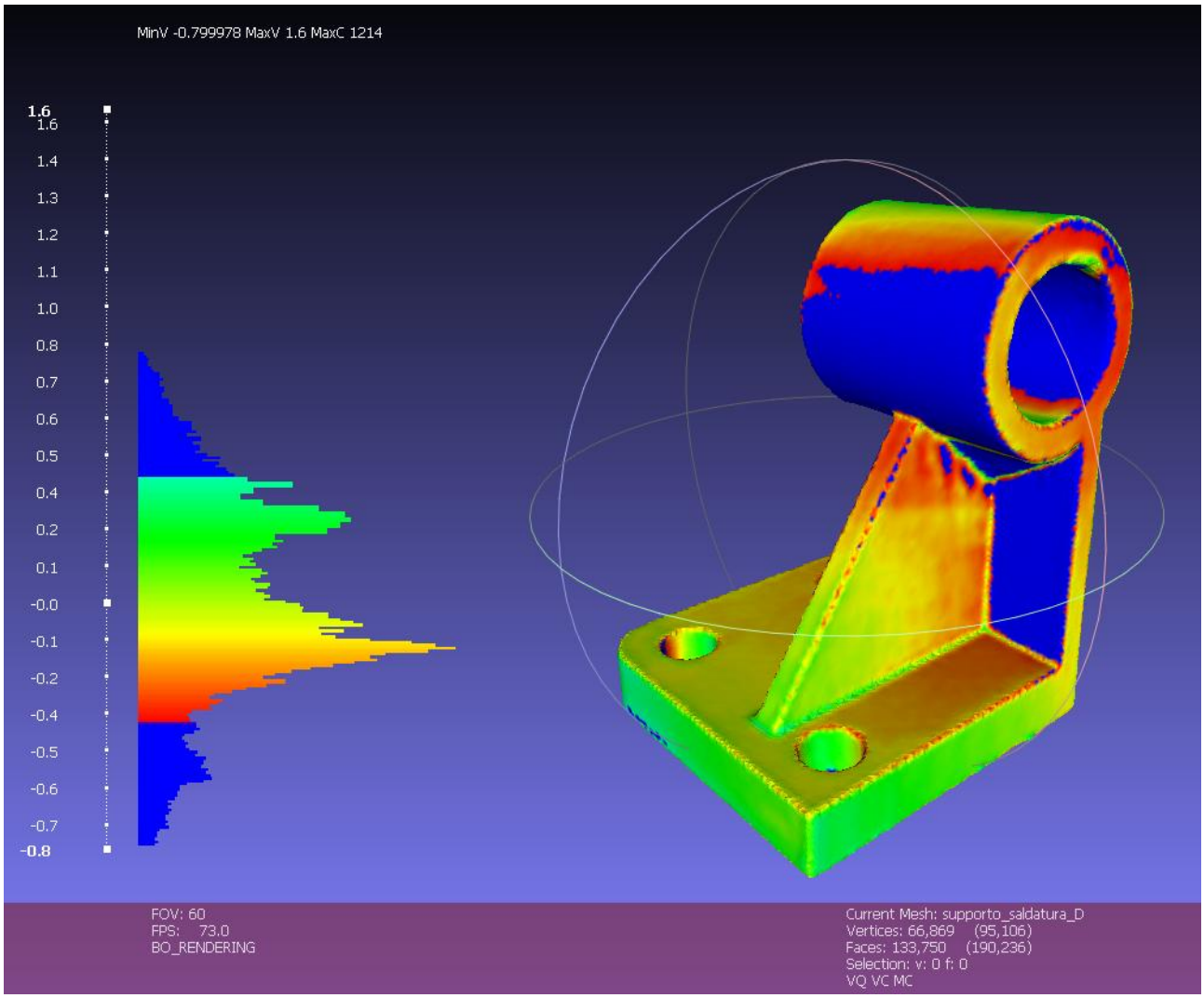


Figura 84. Confronto AD.

3) Caso BD: come nei casi precedenti, la rete non modifica le dimensioni del componente preso in input. Si hanno distorsioni e scostamenti in corrispondenza della superficie esterna in blu.

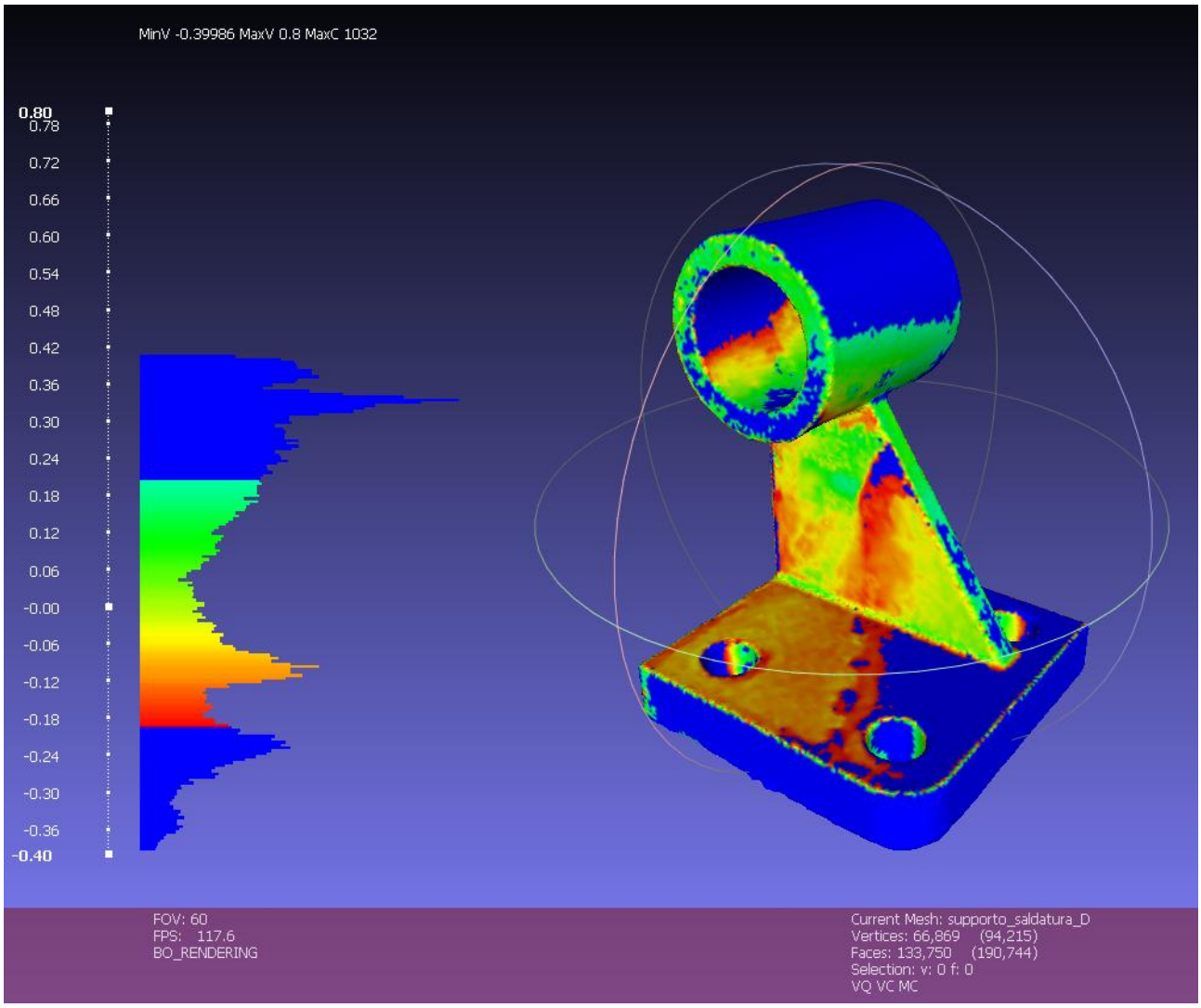


Figura 85. Confronto BD.

4) Caso AB: lo scanner è stato in grado di acquisire tutte le caratteristiche della geometria del ground truth, fornendo un buon risultato. Si hanno scostamenti maggiori nelle aree in blu.

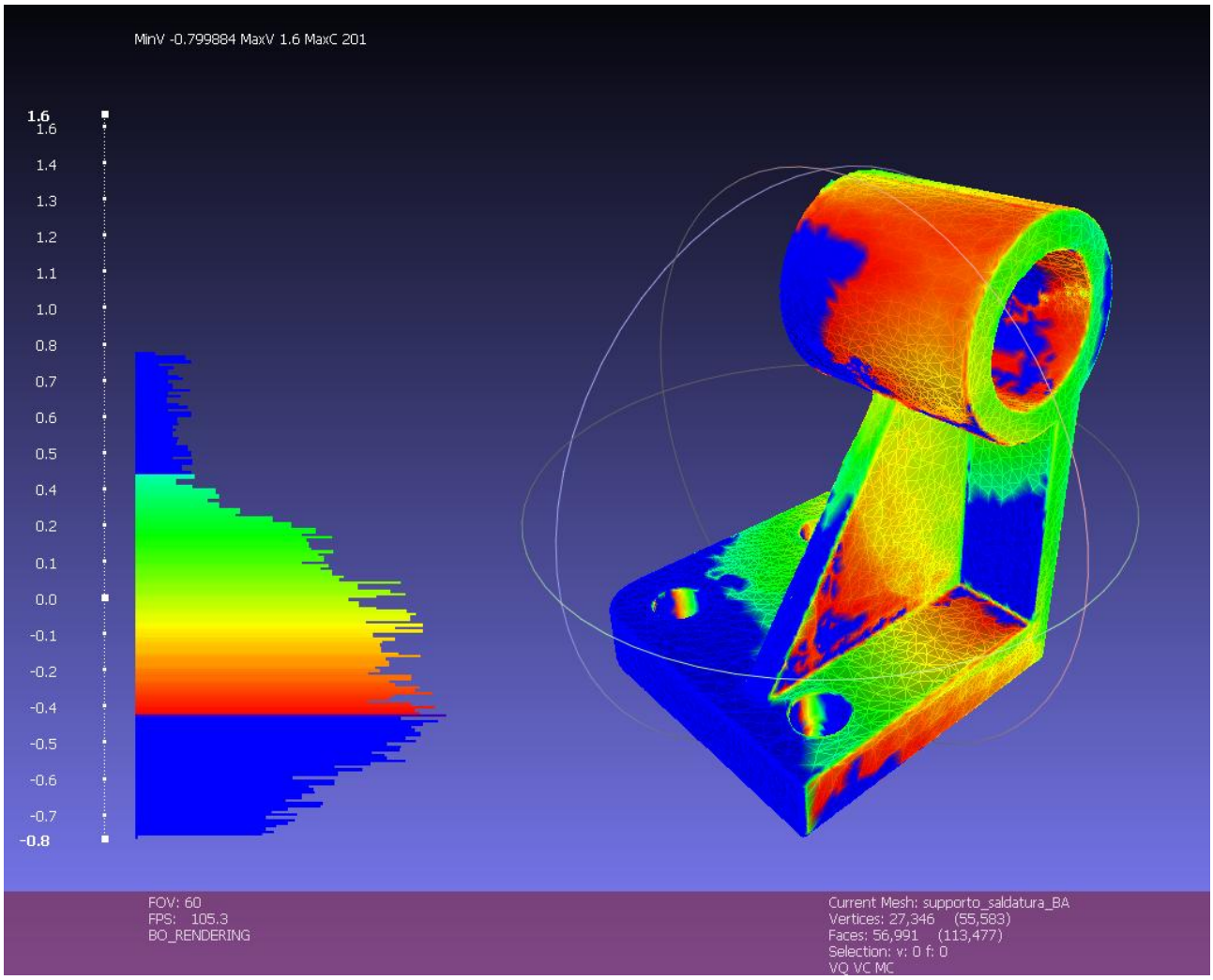


Figura 86. Confronto AB.

7.3 Boccola

In maniera analoga al test svolto per il `supporto_saldatura`, si mantiene costante il learning rate variando progressivamente il numero di epoche e il campionamento del file `original`. Come nei casi precedenti, avendo a disposizione una mesh con un numero superiore di triangoli, il valore di loss si riduce.

Test	Mesh_gt	Mesh_original	Tipologia	Learning rate	N°epoche	Batch size	k1	k2	kn	Dm_size	Drop rate	loss
Test1	37246	35154	real	0,005	100	5	4	4	4	40	0,6	0,401
Test2	37246	46058	real	0,005	100	5	4	4	4	40	0,6	0,571
Test3	37246	46058	real	0,005	1000	5	4	4	4	40	0,6	0,45416
Test4	37246	34300	real	0,005	1000	5	4	4	4	40	0,6	0,2527

Nella tabella i valori di loss sono relativi all'ultima epoca del test in questione, inoltre come si può vedere dal numero di triangoli, Test1 e Test4 sono stati eseguiti a partire dal ground truth mediante rottura manuale del reticolo. I valori di Test2 e Test3 invece si sono ottenuti a partire dalla scansione del modello stampato in 3D.

7.3.1 Boccola ground truth

Secondo la nomenclatura dei modelli CAD definita nei capitoli precedenti, nella figura si riporta il file nominato come `boccola_gt.obj`:

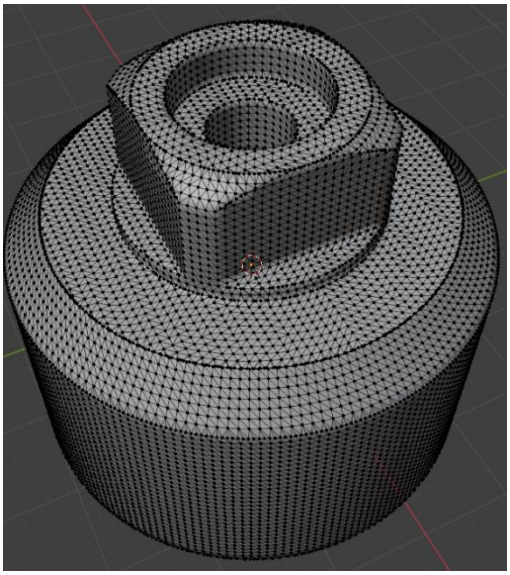


Figura 87. Ground truth.

Caratteristiche della mesh poligonale:

- Numero vertici: 18623;
- Numero spigoli: 55869;
- Numero facce: 37246.

In manera analoga ai modelli precedenti, il reticolo è stato generato da Creo 11, ponendo il target delle facce pari a circa 40000. La mesh così ricavata risulta già essere di tipo triangolare e di conseguenza non necessita di successiva elaborazione. Al software Blender compete solo l'esportazione del formato in .obj.

7.3.2 Boccola original

In questa sezione si propongono due modelli CAD che presentano una mesh incompleta. Questi file costituiscono l'input della rete neurale. in maniera analoga ai componenti meccanici precedenti, in accordo con la designazione vengono nominati entrambi `boccola_original.obj`.

1. Rottura manuale della mesh poligonale:

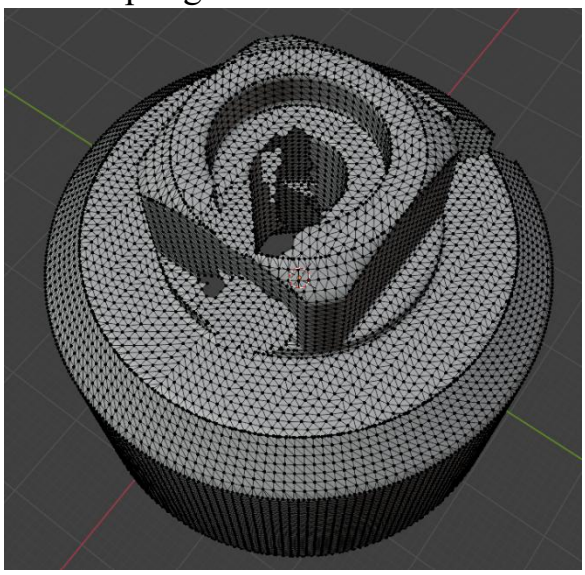


Figura 88. Original mesh poligonale.

Caratteristiche della mesh poligonale:

- Numero vertici: 17371;
- Numero spigoli: 51676;
- Numero facce: 34300.

Prendendo come input il file `boccola_gt.obj`, si procede con l'eliminazione del reticolo poligonale in maniera del tutto casuale, interessando aree piane, curve e bordi interni, per simulare la mancata lettura della superficie da parte dello scanner 3D. È importante notare che grazie al livello del campionamento iniziale, risulta agevole l'eliminazione mirata anche di piccole parti della superficie.

2. Mesh ricostruita tramite scansione 3D:

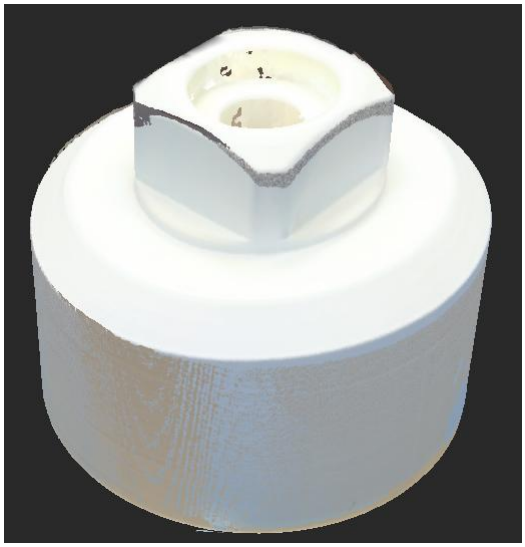


Figura 89. Original scanner 3D.

La figura sopra mostrata riporta il risultato finale della scansione 3D ottenuta tramite Revopoint Mini. La superficie apparentemente presenta solo qualche piccola porzione mancante, soprattutto nel foro interno, che risulta più difficile da raggiungere per via della configurazione geometrica esterna e dalla profondità del foro.

Al fine di ottenere la migliore qualità dalla scansione, si rendono necessarie due scansione differenti, le quali grazie all'elaborazione per mezzo di Revoscan 5, vengono unite in un unico modello come mostrato in figura.

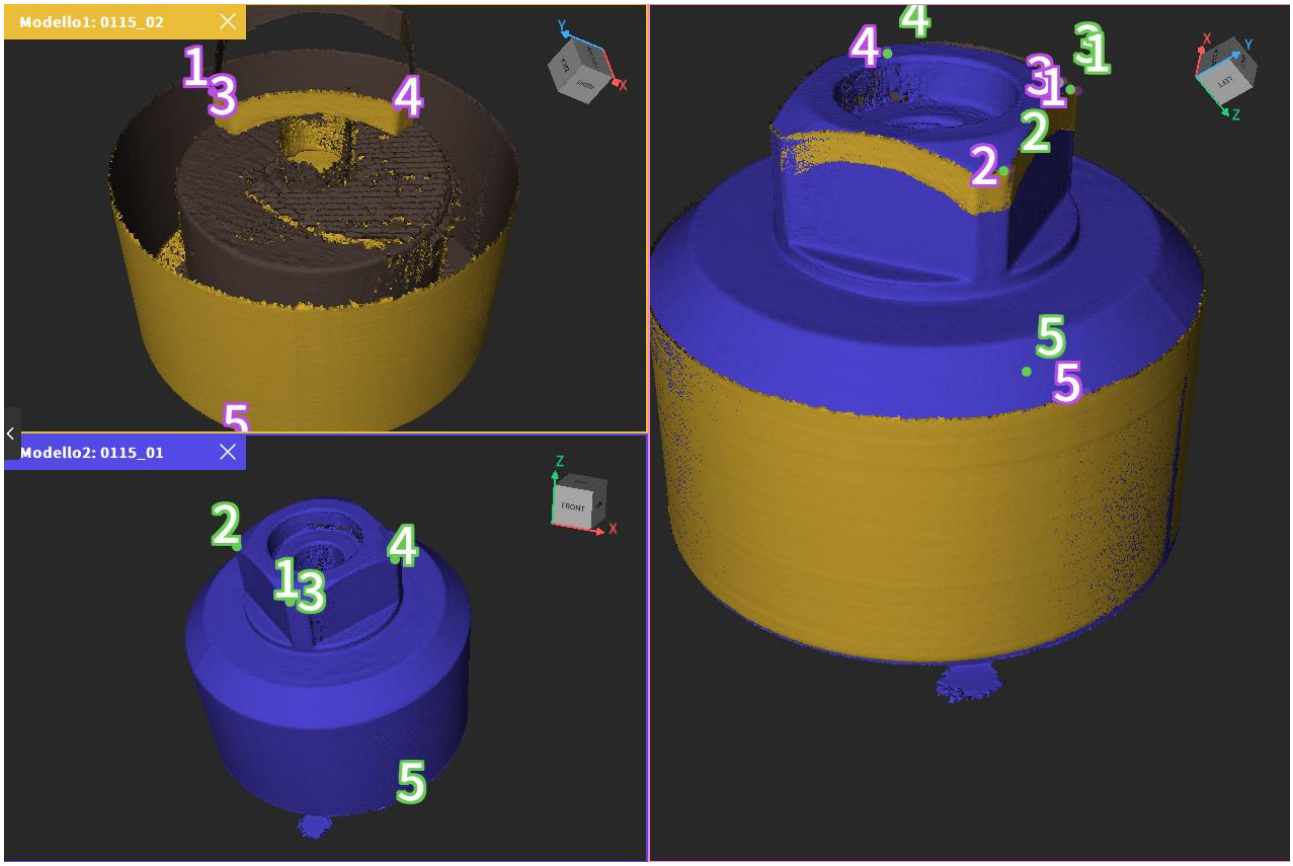


Figura 90. Unione delle scannerizzazioni parziali.

Si riporta di seguito il risultato finale del modello unito importato su Blender:

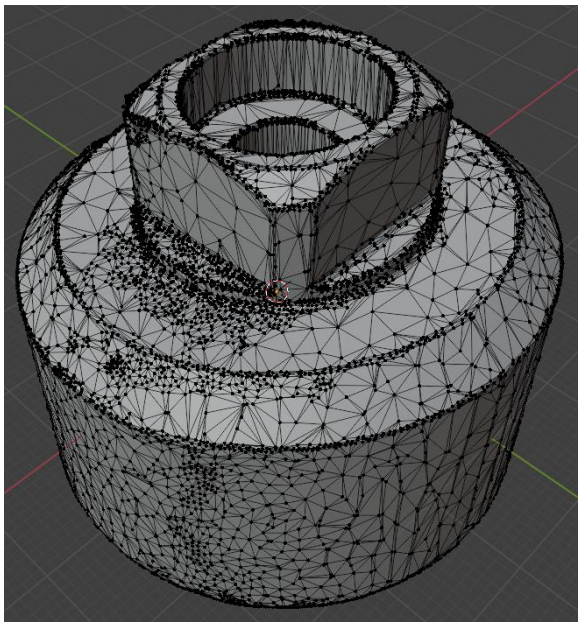


Figura 91. Remesh finale.

Caratteristiche della mesh poligonale:

- Numero vertici: 23038;
- Numero spigoli: 69090;
- Numero facce: 46056.

7.3.3 Risultati

Terminata la fase preliminare di acquisizione della mesh incompleta, si riportano di seguito i risultati relativi al training dei due file `original` di input, attraverso la visualizzazione dei modelli CAD relativi all'epoca 1000.

1. Rottura manuale della mesh poligonale:

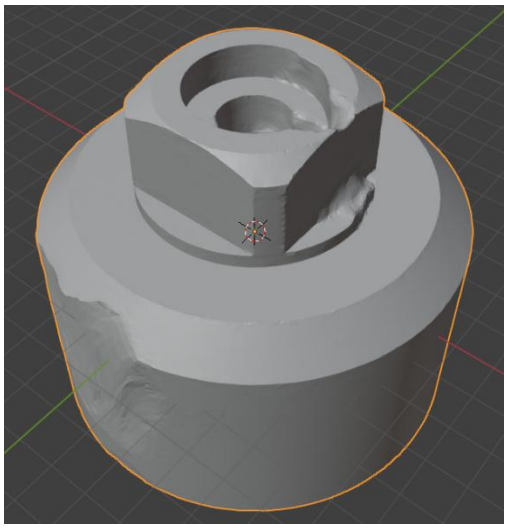


Figura 92. Output rottura manuale.

Caratteristiche della mesh poligonale:

- Numero vertici: 69567;
- Numero spigoli: 208701;
- Numero facce: 139136.

Dalla visualizzazione del risultato finale, emerge nuovamente la difficoltà della rete a ricostruire grandi aree mancanti della mesh poligonale, presentando una distorsione della forma nelle regioni danneggiate. Tuttavia, inserendo il comando `real`, si riscontra un miglioramento nella definizione di bordi e spigoli.

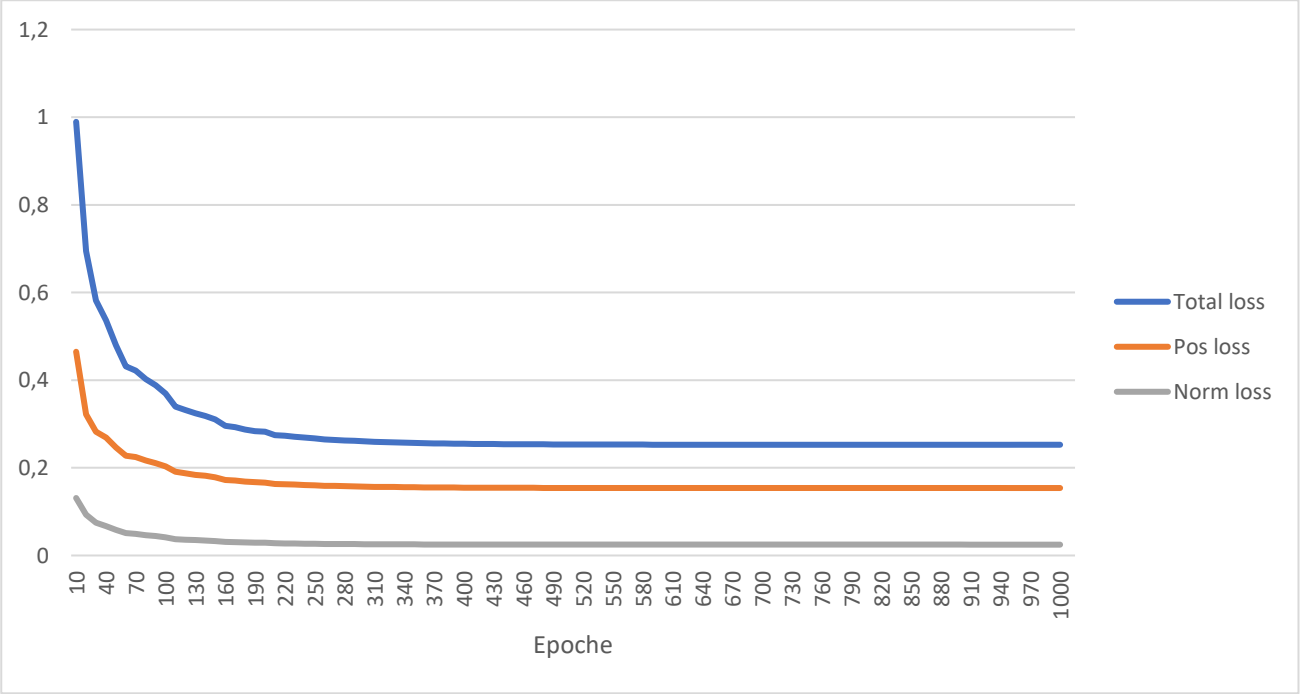


Figura 93. Andamento loss.

2. Mesh ricostruita tramite scansione 3D:

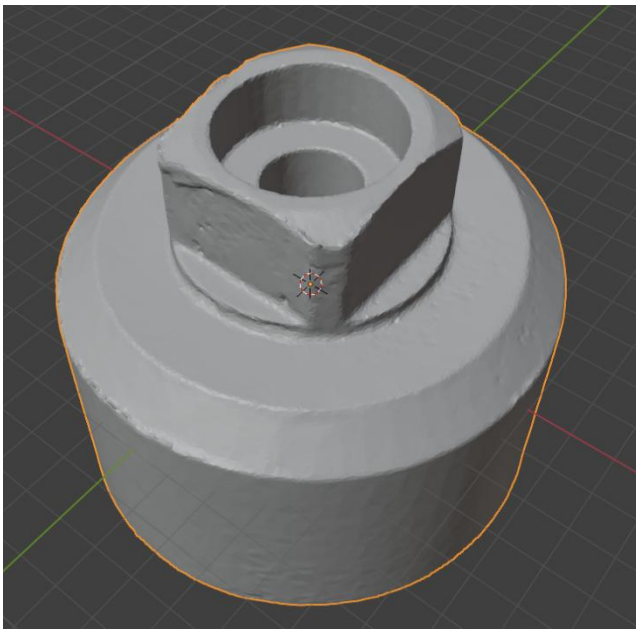


Figura 94. Output scannerizzazione.

Caratteristiche della mesh poligonale:

- Numero vertici: 82402;
- Numero spigoli: 247206;
- Numero facce: 164804.

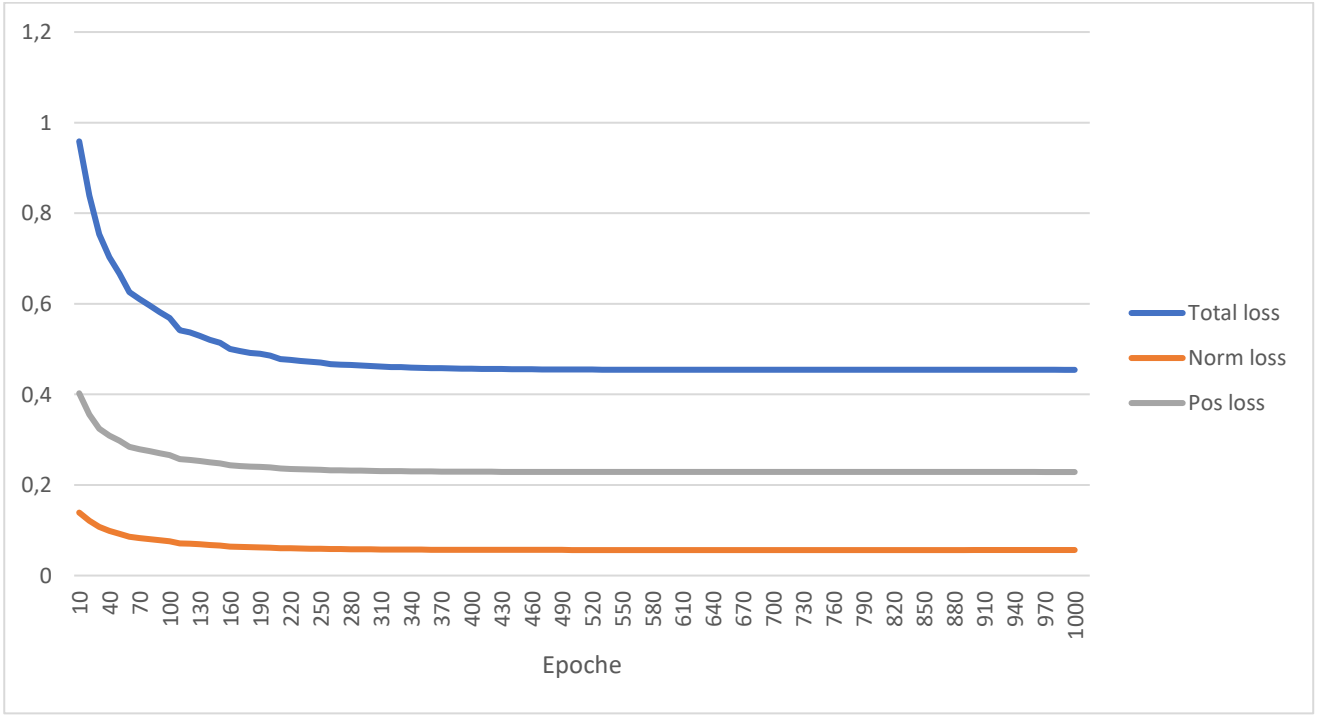


Figura 95. Andamento loss.

7.3.4 Problematiche riscontrate

La prima problematica riscontrata durante l'elaborazione di questo componente, riguarda la difficoltà della rete neurale nel leggere correttamente la geometria per quanto riguarda il modello ottenuto mediane scansione 3D. In particolare nella fase del preprocess della mesh, il file CAD in output ha presentato una notevole distorsione del reticolo, rendendo difficoltosa l'interpretazione del componete ed eliminando quasi completamente le caratteristiche geometriche della parte superiore.

Il problema è stato risolto ricostruendo manualmente su Blender la parte interna del foro centrale a diametro inferiore e per salvare le modifiche apportate si è resa necessaria un'ulteriore attività di remeshing. Il modello presentato nel paragrafo scansione 3D dei risultati, mostra la validità della soluzione trovata, in quanto il componente è stato ricostruito correttamente dalla rete neurale.

Inoltre, per quanto riguarda la rottura manuale della mesh del componente, per verificare se la distorsione è dovuta all'eccessiva area asportata, oppure a problematiche dovute alla conformazione della geometria, si esegue un nuovo test eliminando sempre in maniera casuale delle regioni più piccole della superficie.

Di seguito si offre una visualizzazione grafica delle nuove modifiche:

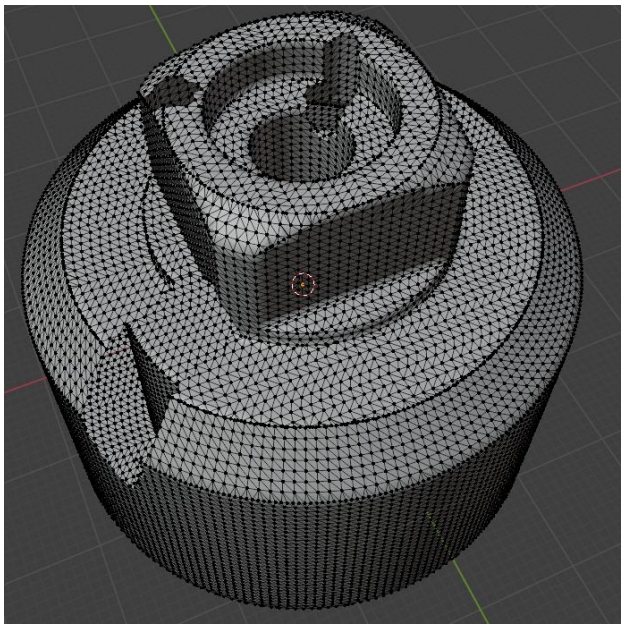


Figura 96. Rottura remesh.

Caratteristiche della mesh poligonale:

- Numero vertici: 18168;
- Numero spigoli: 54304;
- Numero facce: 36132.

Al termine della fase di training si ottiene il seguente risultato:

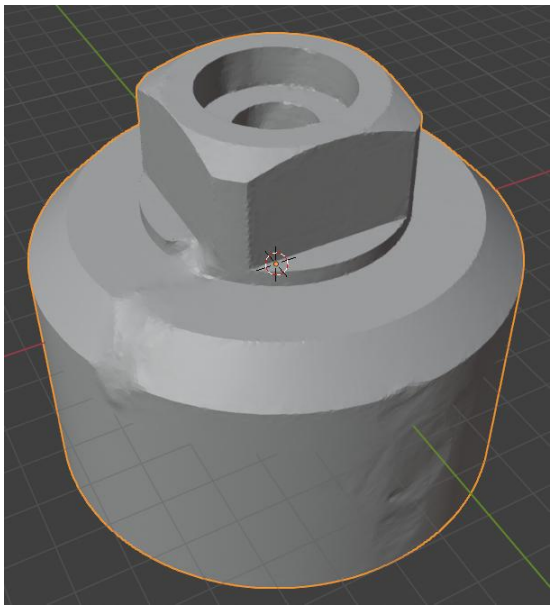


Figura 97. Output rottura manuale remesh.

Caratteristiche della mesh poligonale:

- Numero vertici: 70103;
- Numero spigoli: 210309;
- Numero facce: 140206.

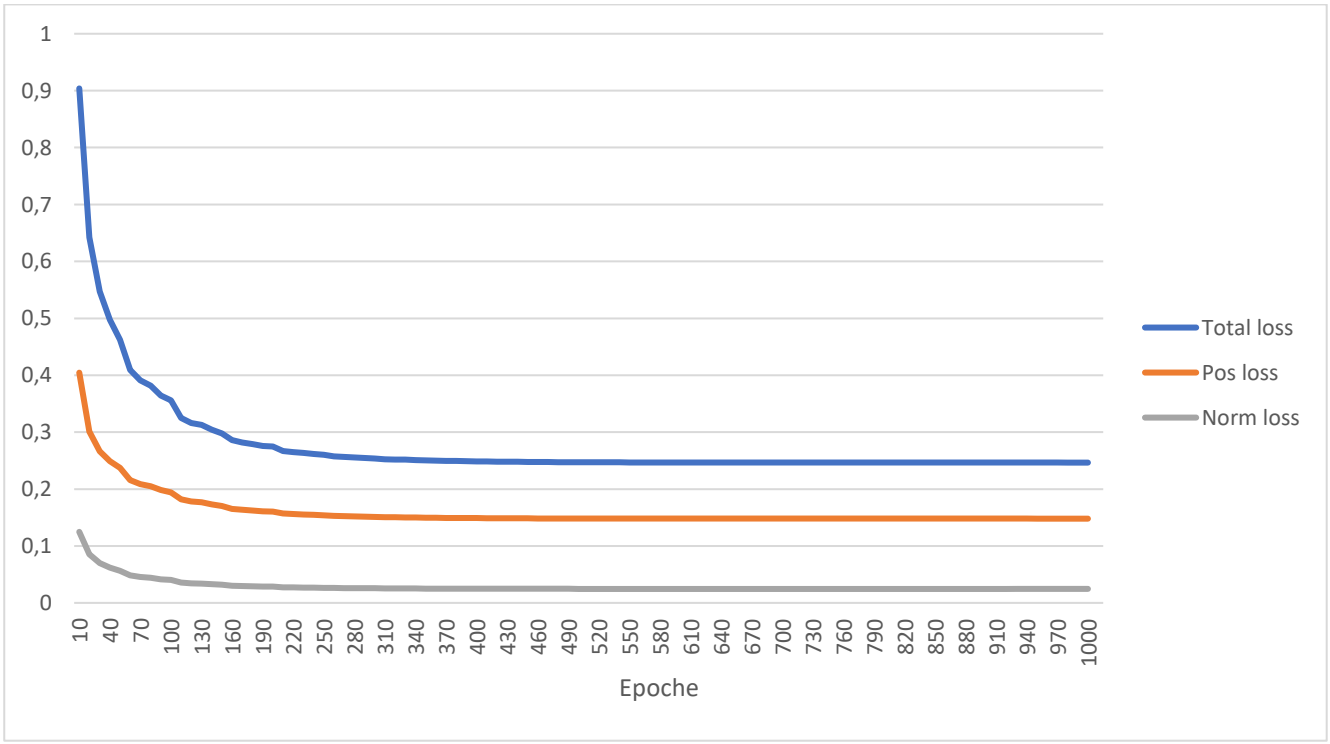


Figura 98.Aandamento loss remesh.

Come si può osservare dal modello CAD relativo all'output dell'epoca 1000, la mesh poligonale risulta essere meno distorta ricostruendo con più precisione gli spigoli e i

bordi del pezzo meccanico. Da ciò si può dedurre che la rete neurale può riparare mancanze di entità ridotta nel reticolo.

7.1.4 Valutazioni

In maniera analoga ai componenti meccanici precedenti, per permettere un confronto tra i vari test, nella fase di training si sono settati gli stessi parametri, variando solamente le discontinuità della mesh.

Dai risultati finali ottenuti, è emerso nuovamente la difficoltà della rete nel riparare importanti aree mancanti della superficie. Tuttavia, questa problematica sembra riferita solo all'entità della discontinuità e non è legata alla conformazione della geometria, come dimostrato dall'ultimo test. Partendo infatti sempre dal ground truth e effettuando piccole rotture nel reticolo, la qualità dell'output finale risulta notevolmente migliorata sia in termini di visualizzazione grafica, sia per quanto riguarda l'andamento della funzione di perdita.

Inoltre, con riferimento all'andamento del valore di loss in funzione del numero di epoche, si riscontra un plateau della funzione di perdita dovuto nuovamente al basso valore di learning rate.

Di seguito sono presenti i risultati ottenuti dal confronto tramite MeshLab.

Confronto	Distanza minima [mm]	Distanza massima [mm]	Distanza media [mm]	RMS [mm]
A-C	0,757094	0,757094	-0,010041	0,215403
A-D	3,851098	3,851098	-0,347866	0,720627
B-D	0,986762	0,986762	0,142966	0,312097
A-B	1,762467	1,762467	-0,459443	0,768494

1) Caso AC: il confronto evidenzia il difetto di ricostruzione della rete.

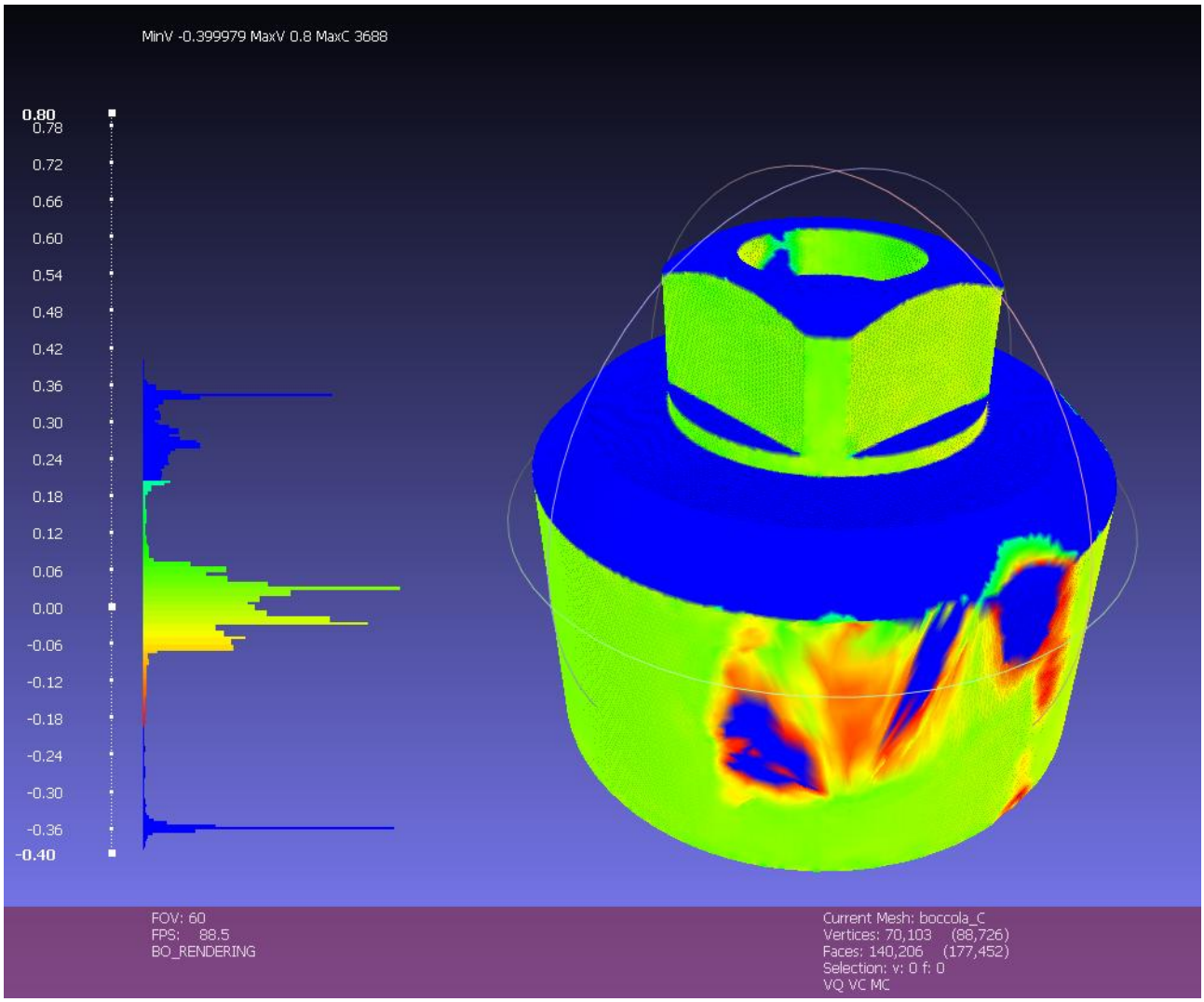


Figura 99. Confronto AC.

2) Caso AD: risulta evidente un difetto dovuto alla scansione effettuata nella parte superiore del componente.

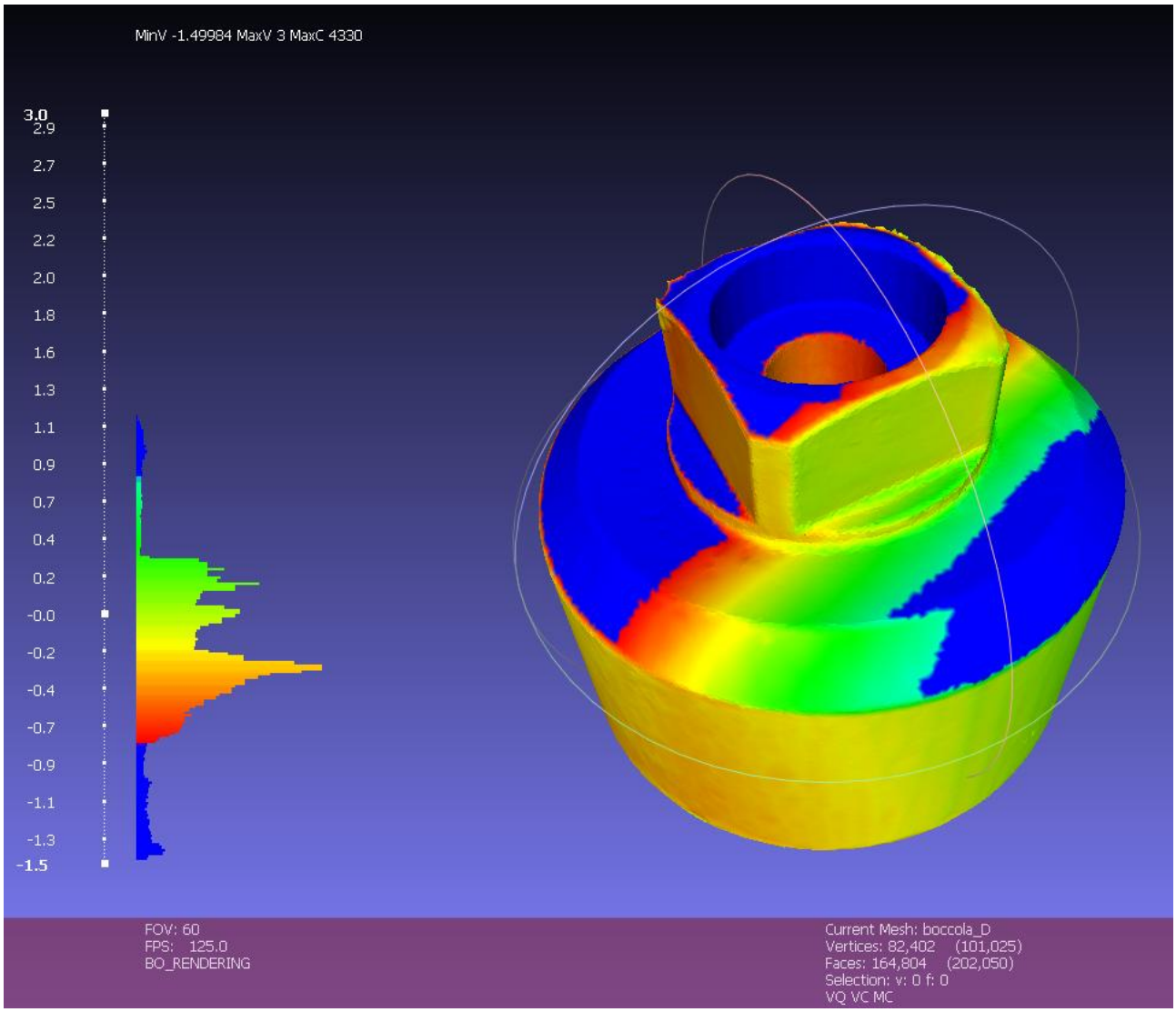


Figura 100. Confronto AD.

3) Caso BD: come nei casi precedente, la rete non modifica le dimensioni del modello datogli ingresso.

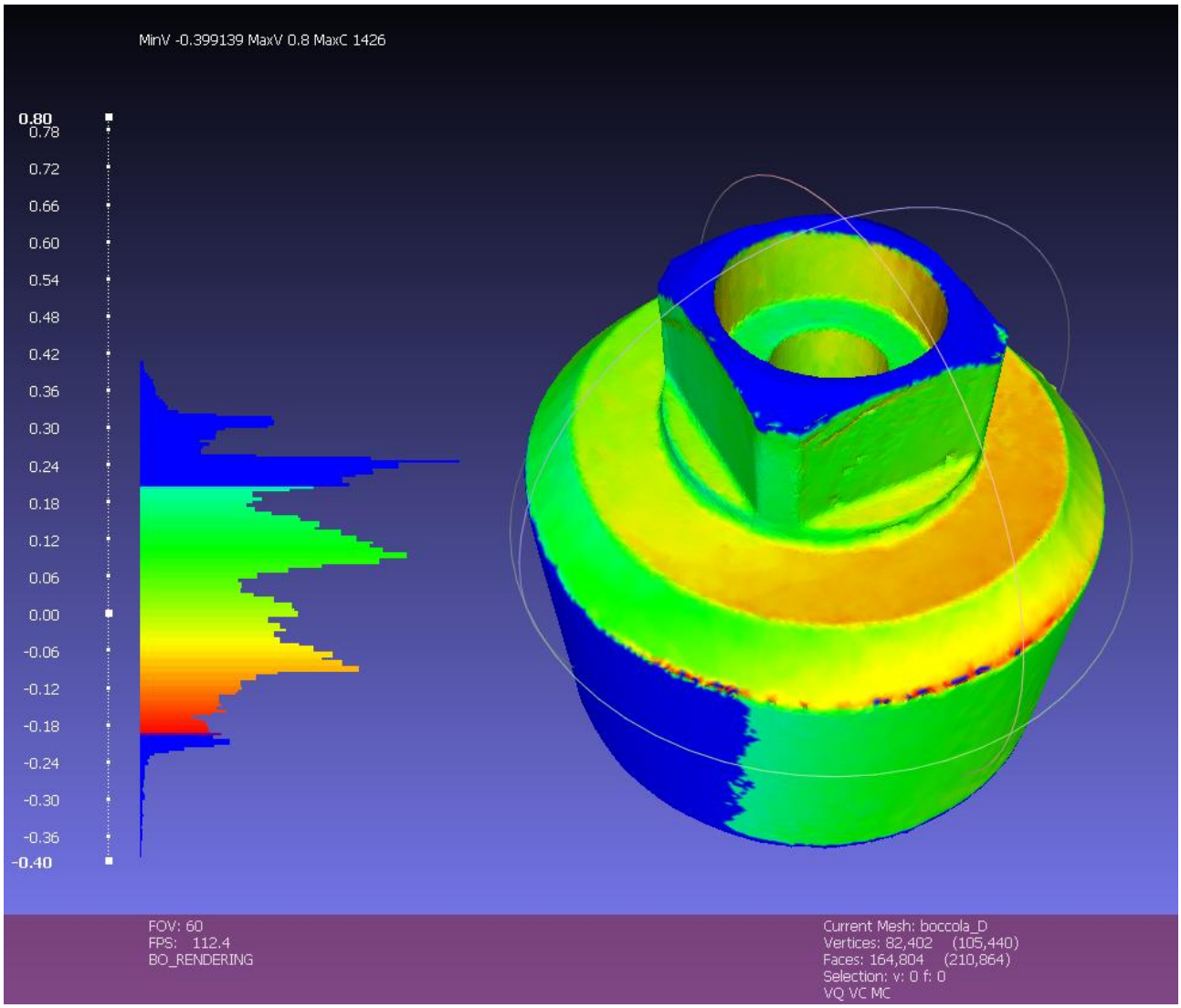


Figura 101. Confronto BD.

4) Caso AB: dai risultati ottenuti risulta evidente la presenza di differenza tra il ground truth e il modello ottenuto dalla scansione. Una seconda scansione più accurata porterebbe a migliorare il risultato ottenuto.

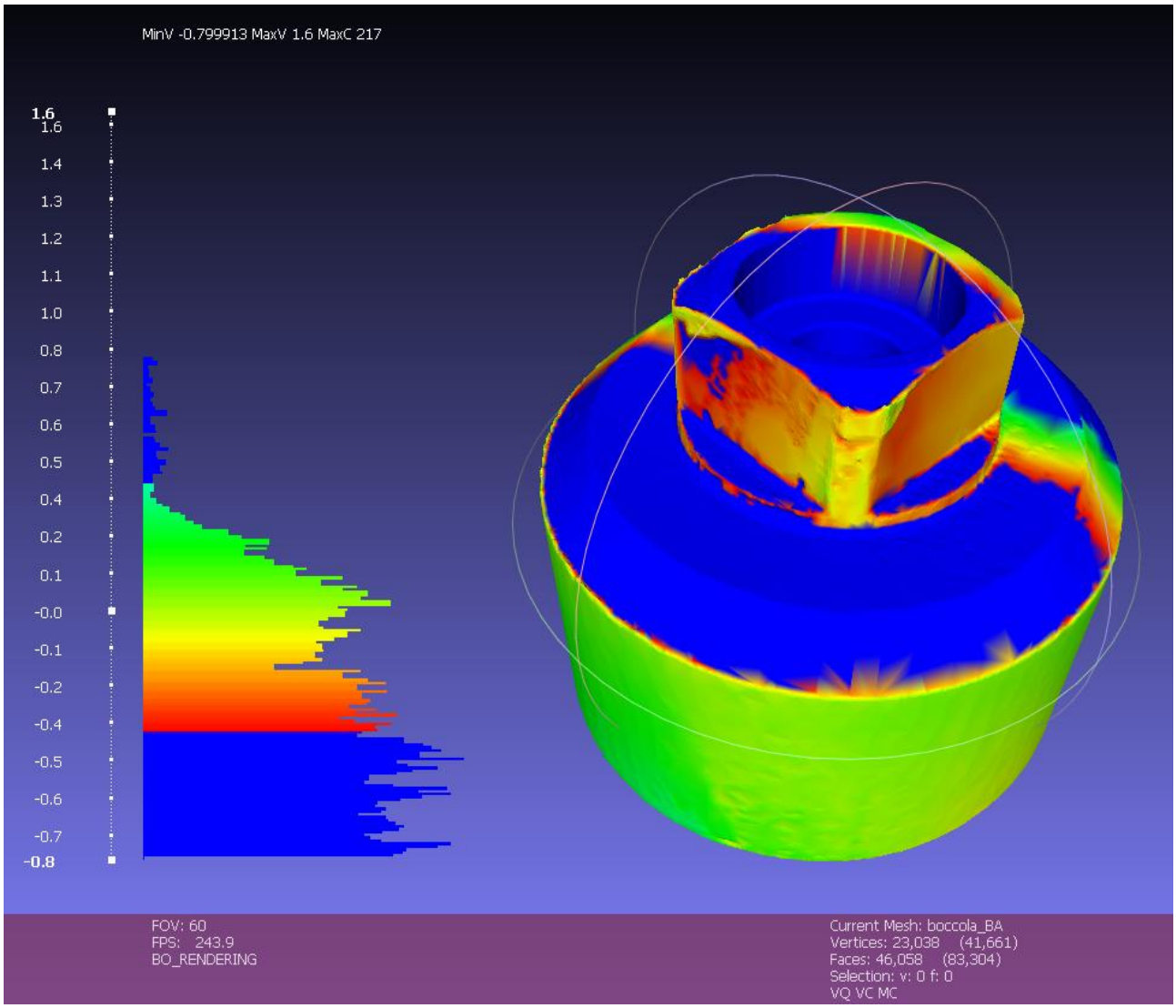


Figura 102. Confronto AB.

7.4 Giunto di Holdham

Si procede con la sperimentazione dei componenti meccanici analizzando ora il caso del `giunto_di_holdham`. In maniera analoga ai casi precedenti si ripete il test lasciando costante il valore del learning rate e la tipologia del file, variando progressivamente il numero di epoche e il campionamento della mesh incompleta di input.

Test	Mesh_gt	Mesh_original	Tipologia	Learning rate	N°epoche	Batch size	k1	k2	kn	Dm_size	Drop rate	loss
Test1	38168	30405	real	0,005	100	5	4	4	4	40	0,6	0,313
Test2	38168	55478	real	0,005	100	5	4	4	4	40	0,6	0,557
Test3	38168	30405	real	0,005	1000	5	4	4	4	40	0,6	0,38529
Test4	38168	35952	real	0,005	1000	5	4	4	4	40	0,6	0,5693

I valori della funzione di perdita risultano minimi quando il reticolo del file originale risulta più fitto, riducendo così l'approssimazione della superficie.

La tabella mostra il risultato di quattro test condotti su questo componente. Test1 e Test4 test sono stati condotti attraverso un modello in cui sono state eliminate delle facce dalla mesh manualmente, al fine di simulare una scansione reale del pezzo.

Discorso differente per quanto riguarda Test2 e Test3, che come si può intuire dalla differenza del campionamento tra la mesh ground truth e original, quest'ultima è stata ottenuta tramite una scansione reale del `giunto_di _holdham` stampato in 3D.

7.4.1 Giunto di Holdham ground truth

Si presenta in figura il modello che secondo la nomenclatura definita nei capitoli precedenti per poter essere correttamente letti dalla rete, viene nominato come `giunto_holdham_gt.obj`.

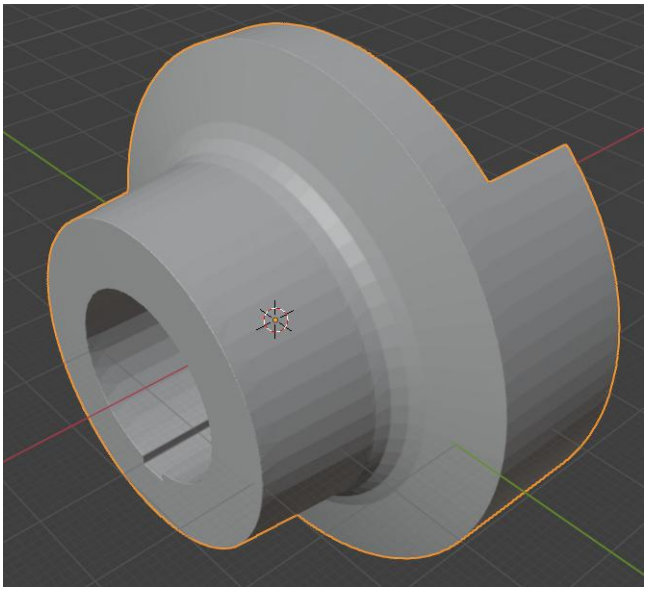


Figura 103. Ground truth.

Caratteristiche della mesh poligonale:

- Numero vertici: 19084;
- Numero spigoli: 57252;
- Numero facce: 38166.

In maniera analoga ai pezzi meccanici visti in precedenza, la mesh poligonale viene generata sul software Creo 11 prima della sua esportazione in formato .stl. Per quanto riguarda il livello di campionamento è stato scelto un intorno del valore target pari a 40000 triangoli.

7.4.2 Boccola original

Vengono presentati di seguito i modelli CAD in formato .obj che la rete neurale SeMIGCN riceve come input. Questi file presentano una mesh incompleta e in accordo con lo standard sulla nomenclatura vengono nominati entrambi come `boccola_original.obj`.

1. Rottura manuale della mesh:

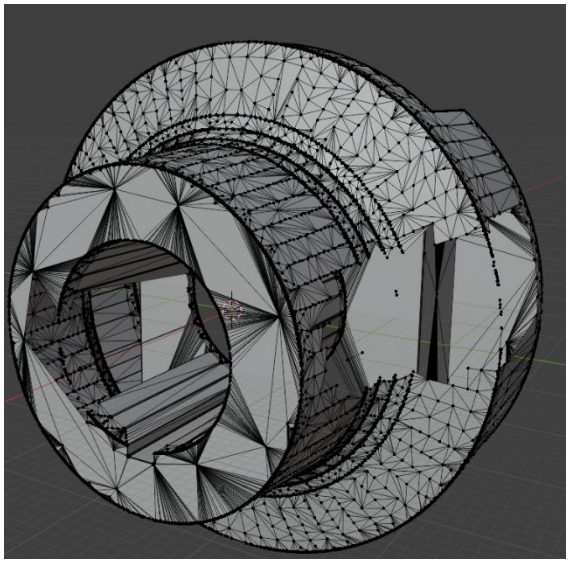


Figura 104. Original mesh poligonale.

Caratteristiche della mesh poligonale:

- Numero vertici: 18178;
- Numero spigoli: 54111;
- Numero facce: 35952.

Utilizzando la mesh poligonale del modello `boccola_gt.obj` come file di partenza, si procede con l'eliminazione dei triangoli in aree scelte casualmente. Rispetto ai casi precedenti, il componente presenta un lungo tratto conico nel collegamento tra le due parti a diverso diametro. Al fine di studiare il comportamento nella rete neurale in questo tratto si è provveduto ad eliminarne una buona superficie come mostrato in figura.

2. Mesh ricostruita tramite scansione 3D:

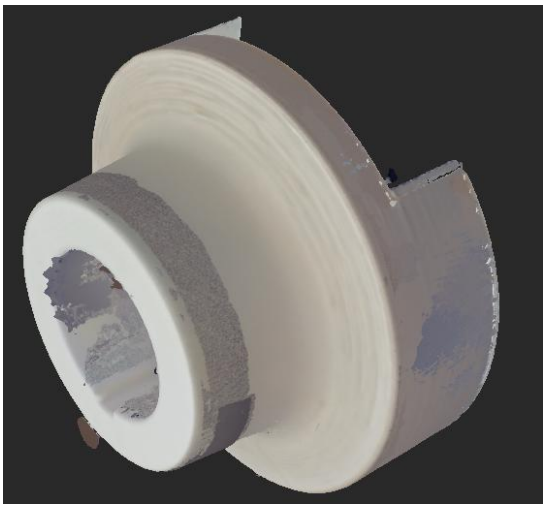


Figura 105. Original scanner 3D.

La figura sopra riportata mostra l'output finale della scansione 3D tramite Revopoint mini e la successiva elaborane attraverso il software Revosscan5. Data la complessità della geometria, per ottenere una più alta qualità del modello di output, si eseguono due scansioni parziali che in un secondo momento vengono unite tramite la procedura di allineamento fornita dal software Revoscan, come mostrato nella figura seguente:

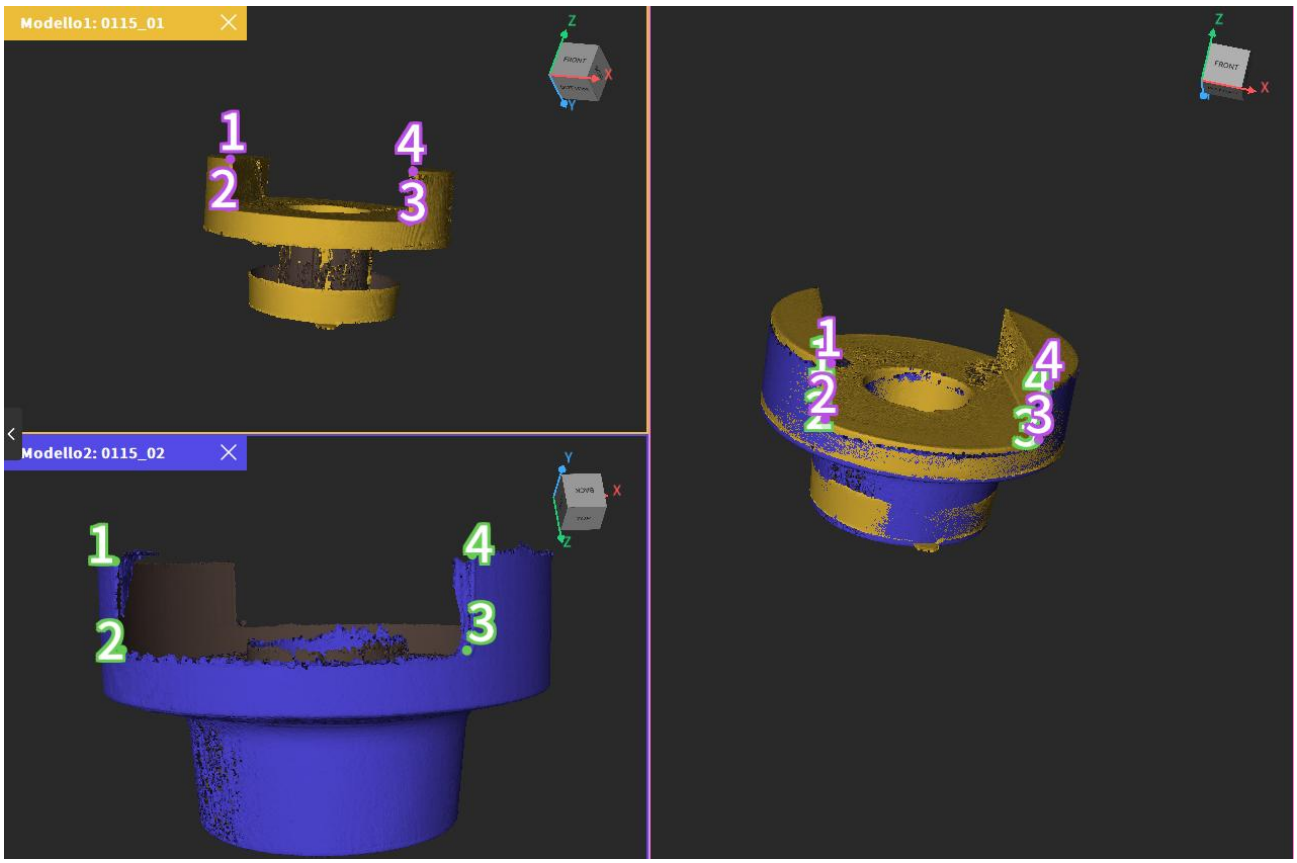


Figura 106. Unione delle scannerizzazioni parziali.

Dopo aver completato l'allineamento delle due parti, il file CAD così ottenuto viene importato su Blender per la successiva fase di remeshing e decimazione. Il risultato finale della mesh del file `original` si presenta nella figura seguente:

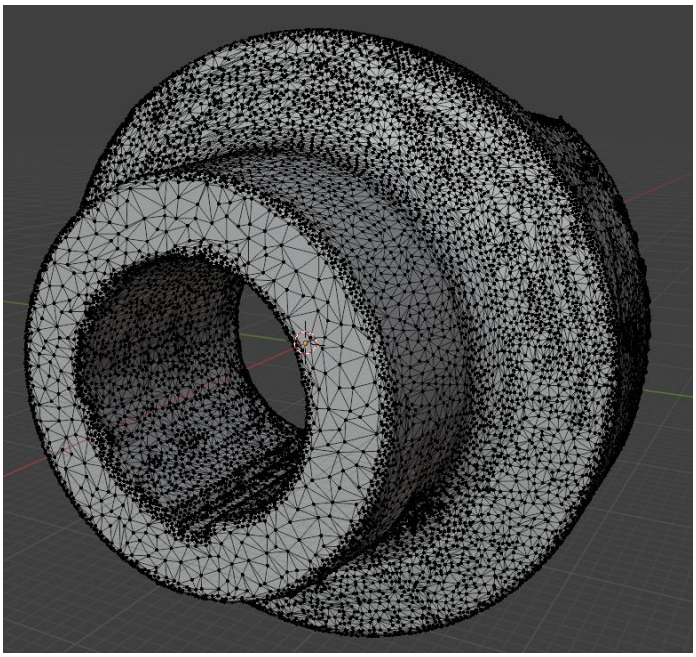


Figura 107. Remesh finale.

Caratteristiche della mesh poligonale:

- Numero vertici: 27755;
- Numero spigoli: 83217;
- Numero facce: 55476.

Dalla figura si riscontra un campionamento più fitto della mesh nel tratto conico.

7.4.3 Risultati

Terminata la fase di elaborazione sui file `original`, questi ultimi vengono dati in pasto alla rete neurale per eseguire la fase di training mediante il lancio dell'algoritmo contenuto nel file sgcn.py. Analogamente ai pezzi meccanici presi in esame nei test precedenti, di seguito si riportano i modelli corrispondenti all'epoca 1000.

1. Rottura manuale della mesh:

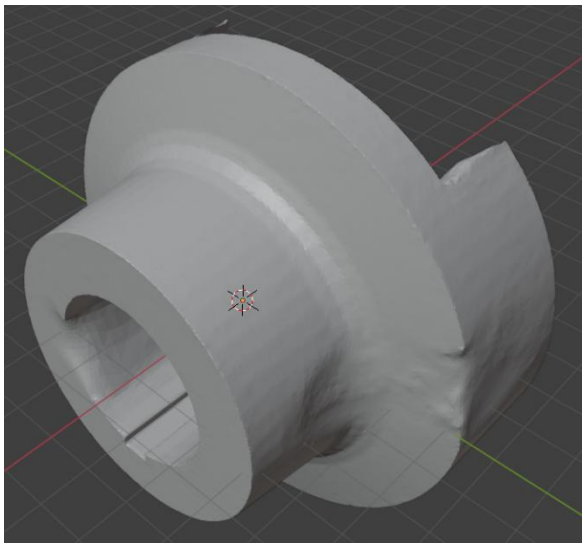


Figura 108. Output rottura manuale.

Caratteristiche della mesh poligonale:

- Numero vertici: 51806;
- Numero spigoli: 155418;
- Numero facce: 103612.

La rete neurale nella fase di training ha ricostruito, senza distorcere eccessivamente, la geometria nella parte conica, nonostante l'entità della mancanza creata. Il difetto iniziale risulta visibile per una leggera ammaccatura della superficie. La qualità del risultato ottenuto risiede nella maggiore facilità riscontrata nel poter riparare la mesh danneggiata se questa si trova in regioni a sezione circolare.

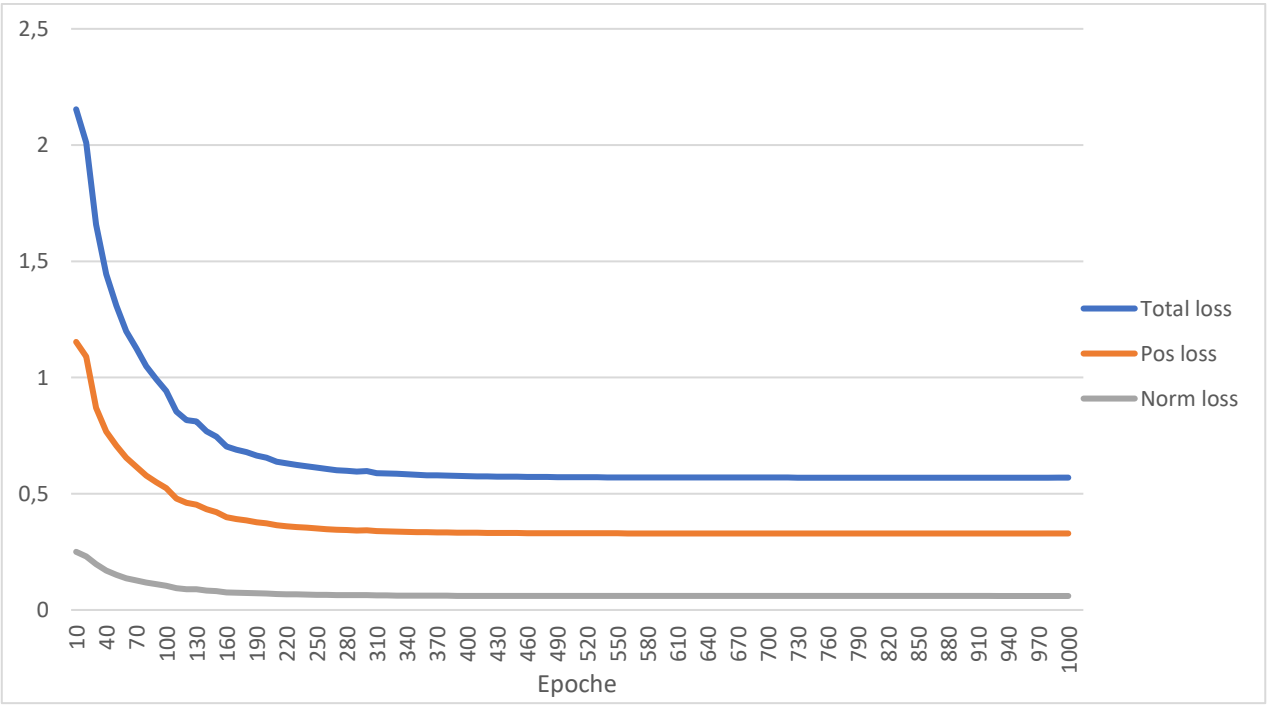


Figura 109. Andamento loss.

2. Mesh ricostruita tramite scansione 3D:

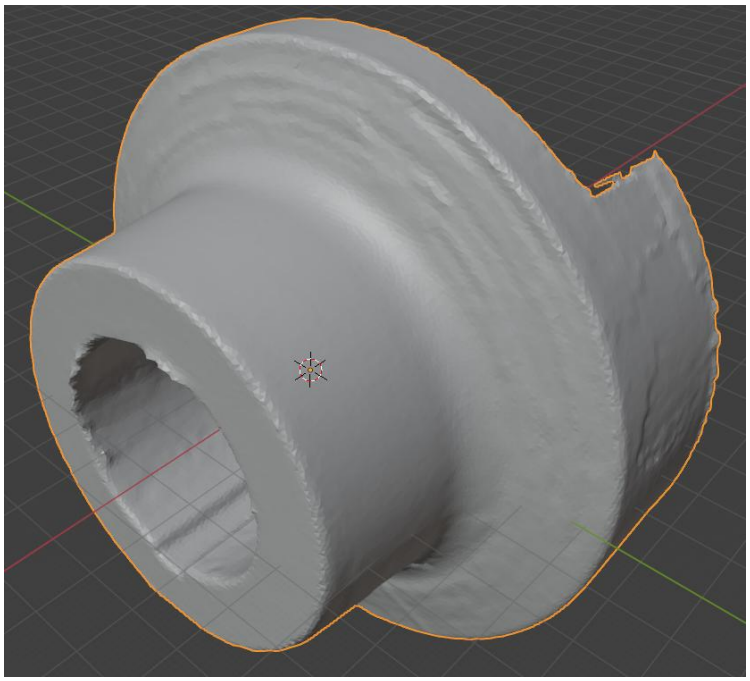


Figura 110. Output scannerizzazione.

Caratteristiche della mesh poligonale:

- Numero vertici: 61894;
- Numero spigoli: 185694;
- Numero facce: 123794.

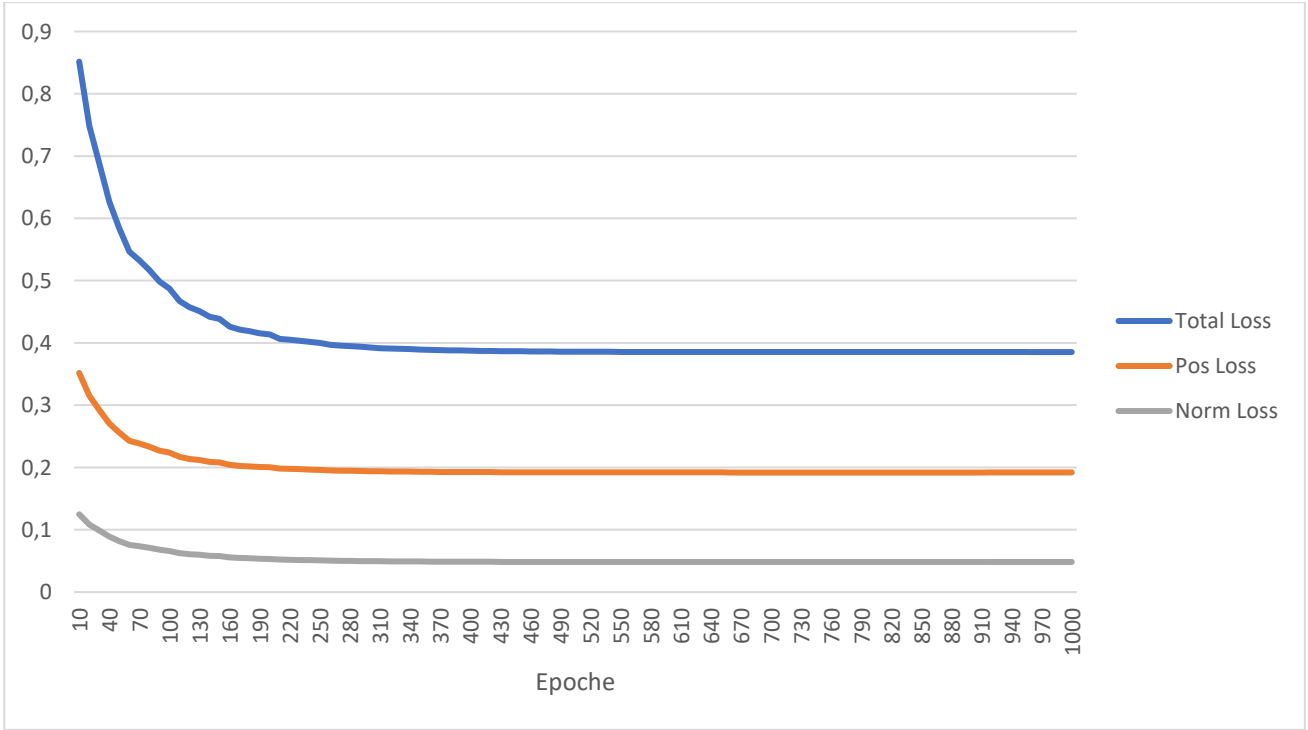


Figura 111. Andamento loss.

7.4.4 Valutazioni

In questo test la rete è stata sottoposta alla ricostruzione del pezzo meccanico avente differenti caratteristiche geometriche. Tuttavia, utilizzando sempre la medesima procedura di elaborazione, il risultato finale ottenuto risulta di buona qualità. Ci si scontra però con le medesime problematiche viste anche nei pezzi analizzati in precedenza, in particolare, per quanto riguarda l'entità della rottura della mesh poligonale e la differente qualità di riparazione, in base a dove si trovano le discontinuità, ottenendo una qualità migliore nel caso di superfici curve rispetto a bordi o spigoli vivi.

In aggiunta come si può osservare nel caso del training della scansione 3D, la fase di elaborazione tramite il software Revoscan 5 per creare una mesh completa e impermeabile attraverso l'unione di due scansioni parziali, ha prodotto un output finale di elevata qualità, raggiungendo così l'obiettivo preposto.

Di seguito vengono riportati i risultati ottenuti dal confronto MeshLab.

Confronto	Distanza minima [mm]	Distanza massima [mm]	Distanza media [mm]	RMS [mm]
A-C	6,044323	6,044323	-0,01615	0,28172
A-D	1,440935	1,440935	-0,349278	0,474796
B-D	0,552247	0,552247	-0,1239	0,279832
A-B	2,736481	2,736481	-0,234957	0,39828

1) Caso AC: il risultato finale ha mostrato il massimo valore di scostamento tra le due mesh nelle regioni ricostruite dalla rete neurale.

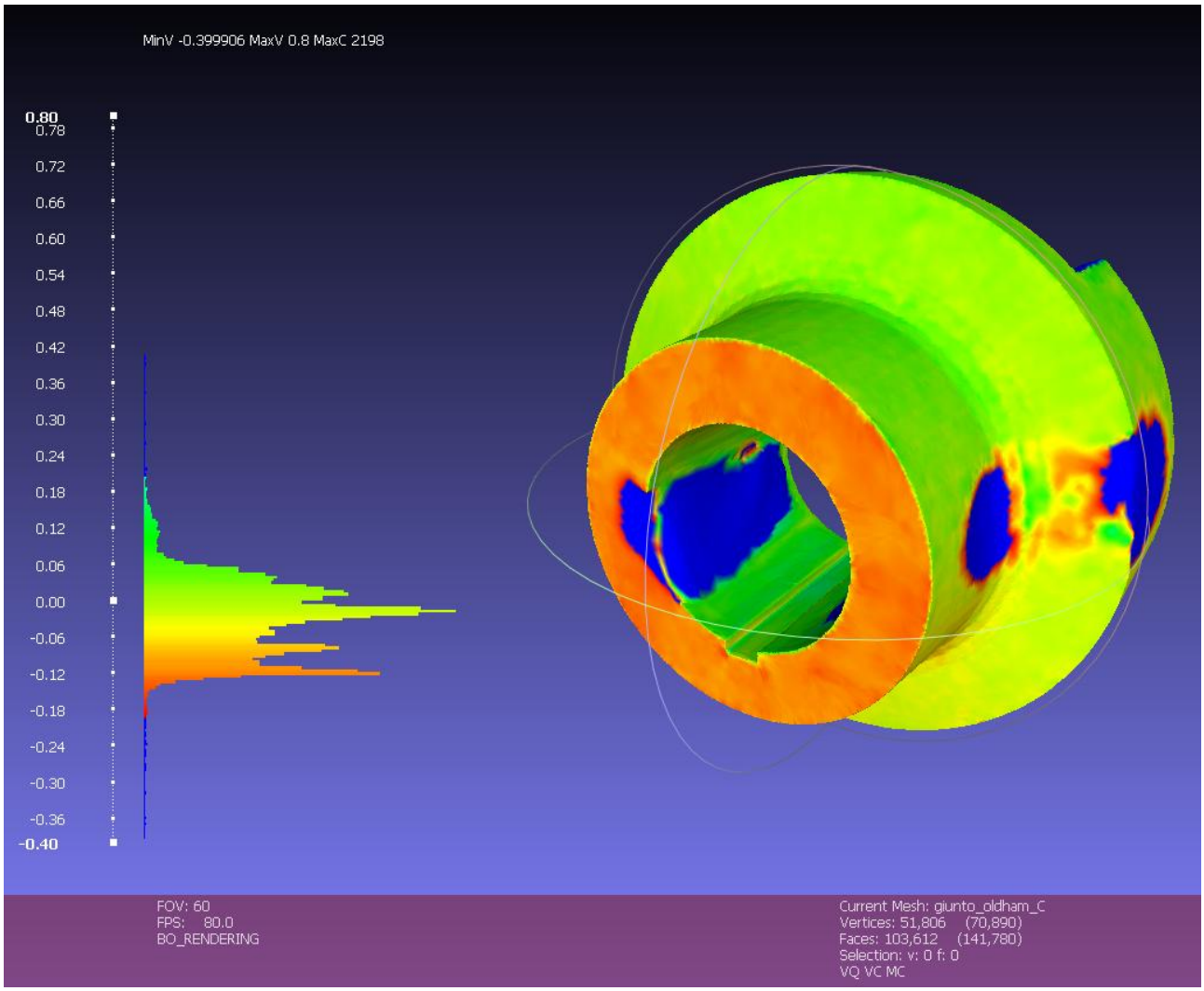


Figura 112. Confronto AC.

2) Caso AD: il modello di output della rete presenta una buona ricostruzione generale, con scostamenti contenuti. In blu si evidenzia il difetto dovuto alla fase di scansione.

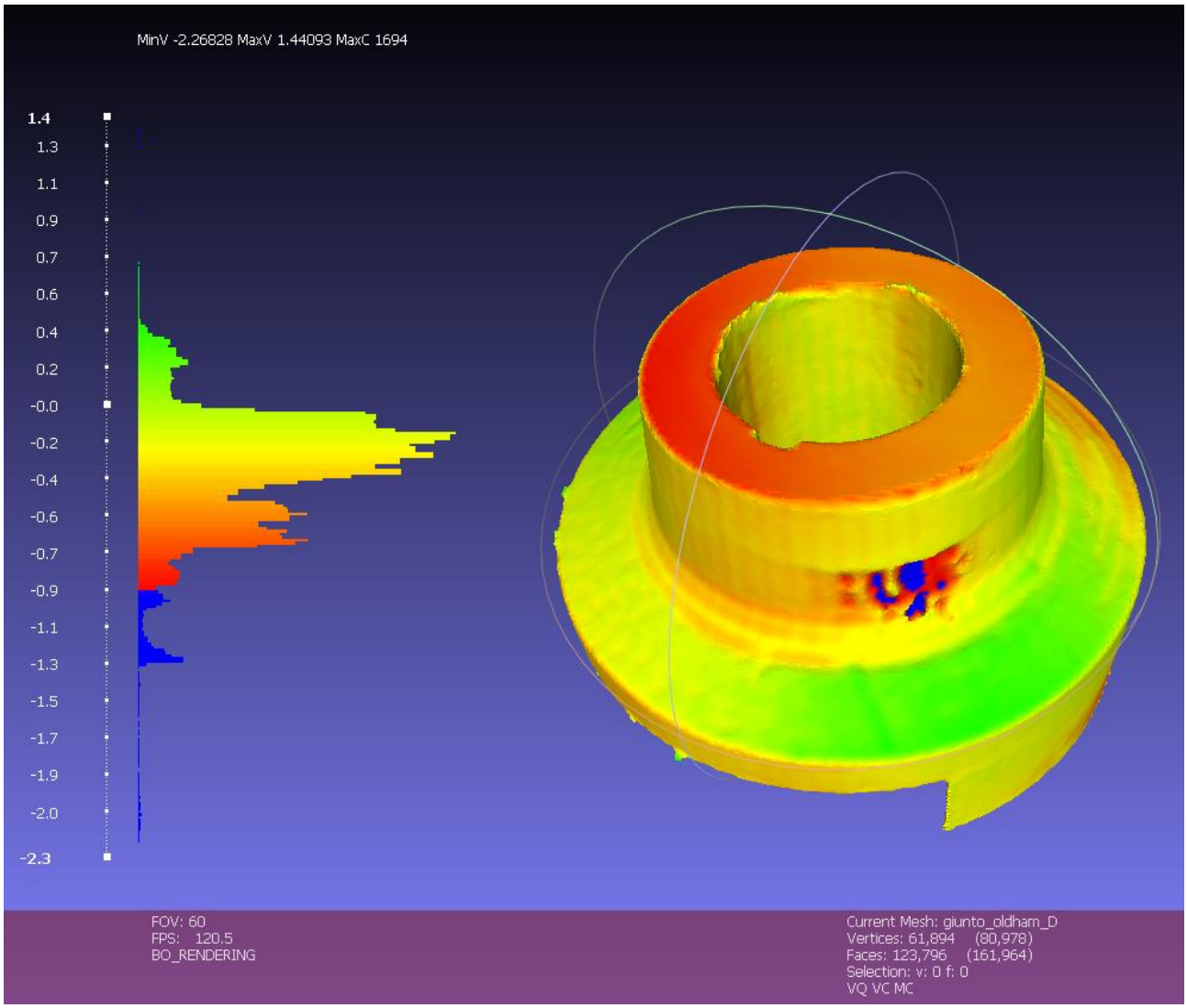


Figura 113. Confronto AD.

3) Caso BD: come riscontrato nei casi studio precedenti, la rete neurale mantiene pressoché inalterate le dimensioni del modello original di input senza presentare variazioni rilevanti.

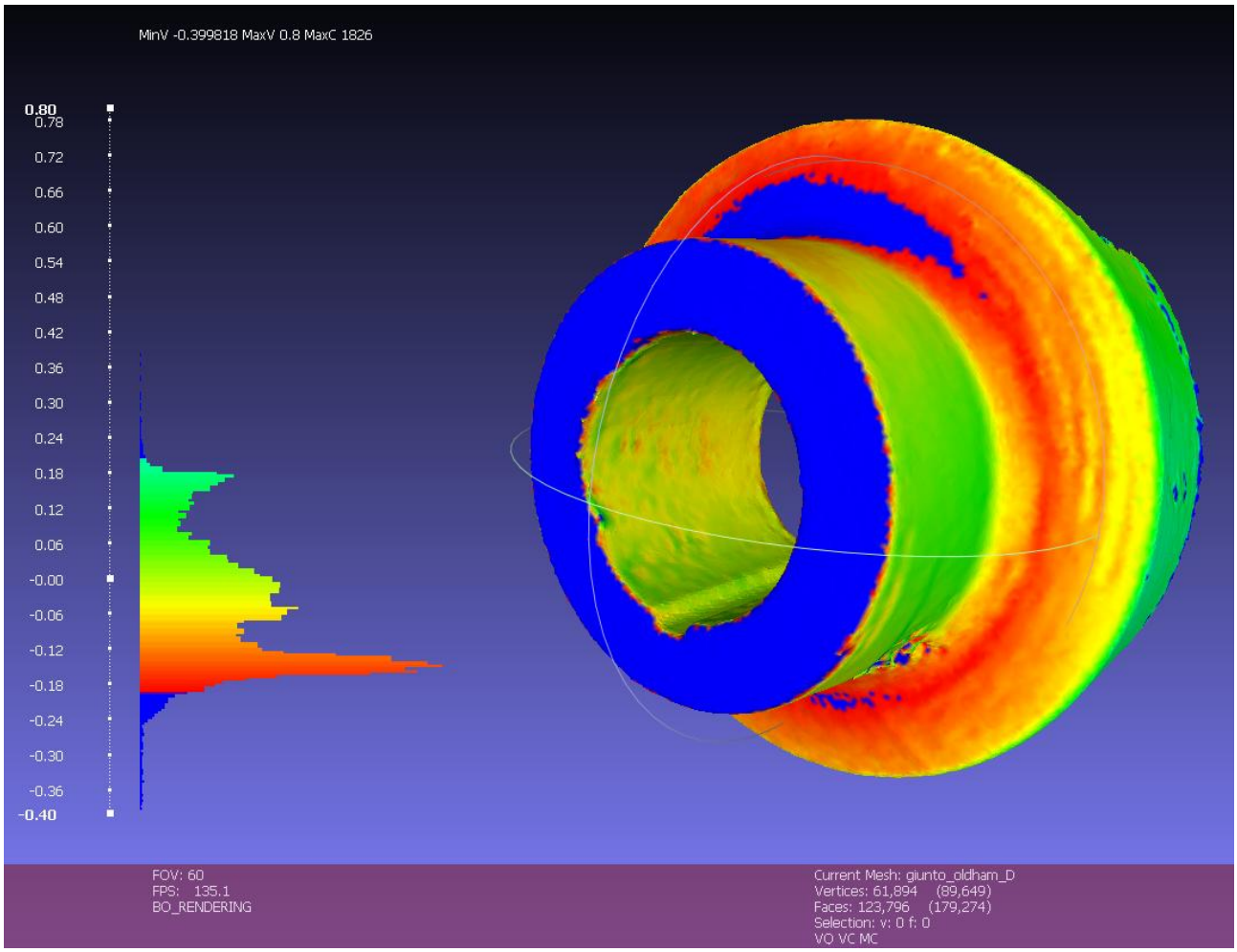


Figura 114. Confronto BD.

4) Caso AB: il modello ottenuto al termine della scansione presenta variazioni dimensionali distribuite in maniera più o meno uniforme, comunque presentando uno scostamento medio dell'ordine del decimo di millimetro.

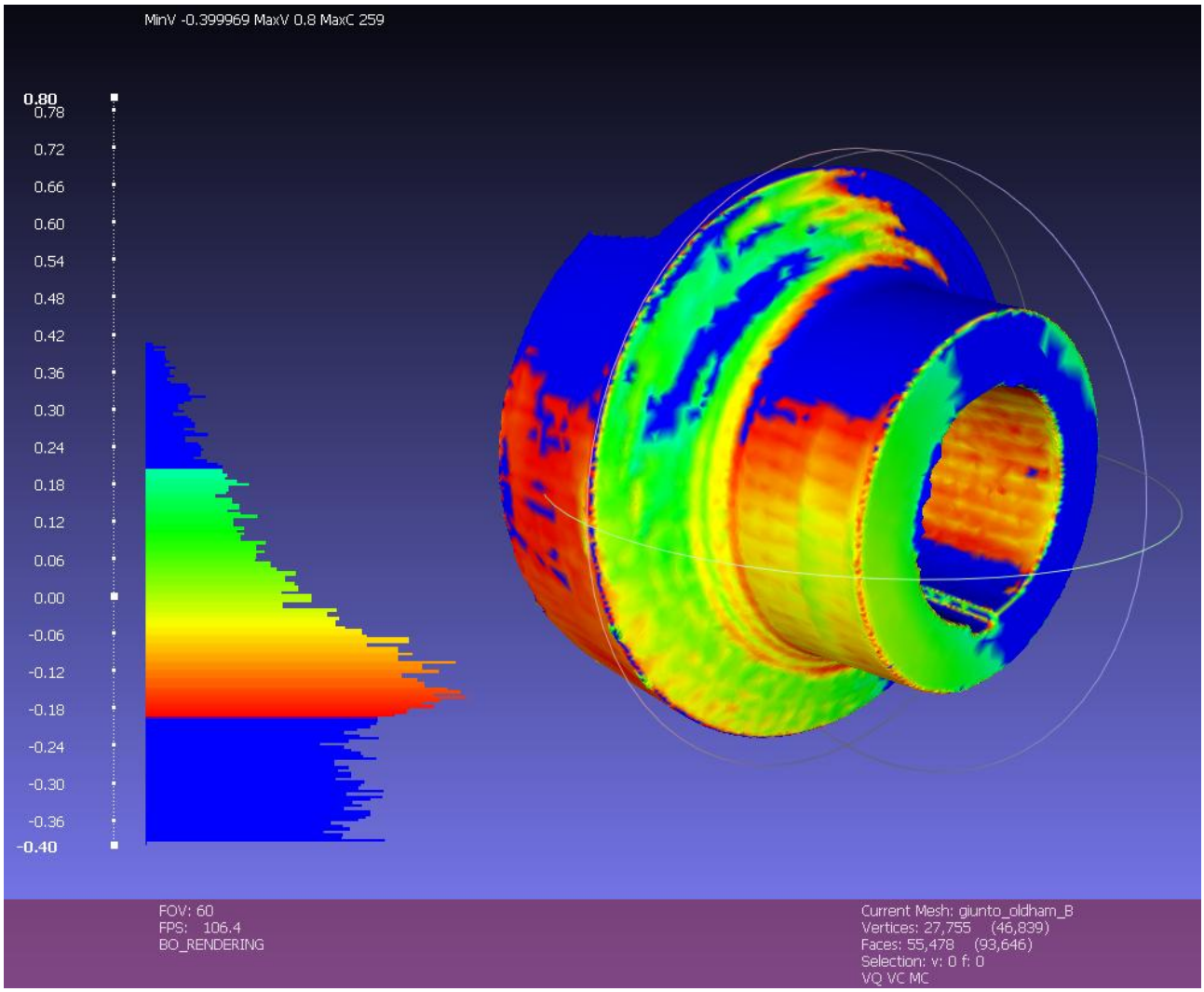


Figura 115. Confronto AB.

7.5 Flangia accoppiata

Si procede con l'analisi del componente meccanico definito con il nome di `flangia_accoppiata`. La particolarità di questo pezzo risiede nel fatto che la sua modellazione si basa sulla `flangia_forata` in modo da permettere il loro accoppiamento mediante la serie circolare di fori sulla sezione circolare.

Test	Mesh_gt	Mesh_original	Tipologia	Learning rate	N°epoche	Batch size	k1	k2	kn	Dm_size	Drop rate	loss
Test1	24184	50428	real	0,005	1000	5	4	4	4	40	0,6	0,34665
Test2	24184	22272	real	0,005	1000	5	4	4	4	40	0,6	0,1821

Come si può osservare dalla tabella riportata, rispetto ai pezzi presi in esame in precedenza, non sono stati effettuati test preliminari data la validità dei parametri di ottimo trovati.

7.5.1 Flangia accoppiata ground truth

Analogamente ai componenti presi in esame in precedenza, in figura si riporta il modello CAD di target, il quale secondo la nomenclatura definita nei capitoli precedenti viene definito con il nome di `flangia_accoppiata_gt.obj`.

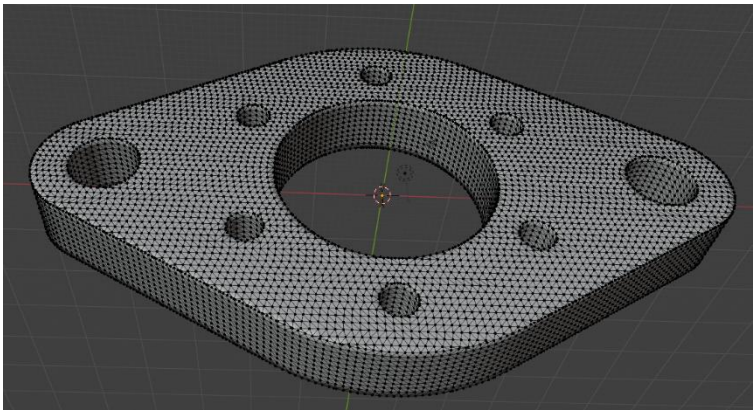


Figura 116. Ground truth.

Caratteristiche della mesh poligonale:

- Numero vertici: 12076;
- Numero spigoli: 36276;
- Numero facce: 24184.

Data la semplicità geometrica del componete e delle sue dimensioni ridotte, tramite il software Creo 11 si è generata la mesh triangolare prendendo come target un valore nell'intorno di 30000 triangoli. Un valore non molto elevato, ma come si vede dalla figura, fornisce un buon campionamento della superficie.

7.5.2 Flangia accoppiata original

Dopo aver definito il modello di target della rete neurale, si procede con la creazione dei due file CAD che costituiscono i modelli di input della rete SeMIGCN, denominati entrambi come `flangia_accoppiata_original.obj`.

1. Rottura manuale della mesh:

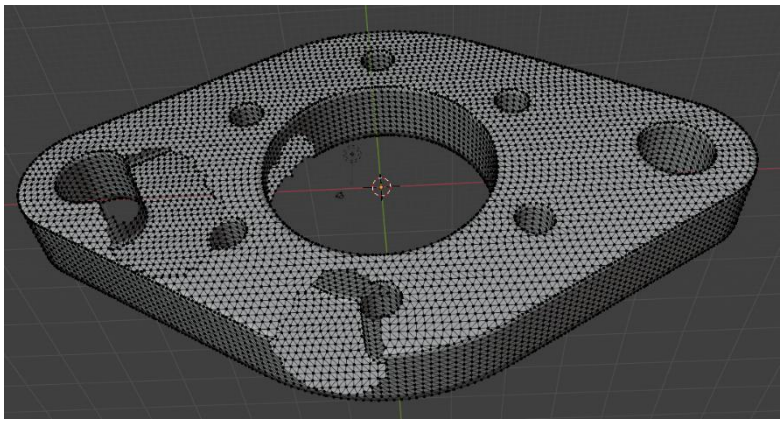


Figura 117. Original mesh poligonale.

Caratteristiche della mesh poligonale:

- Numero vertici: 11258;
- Numero spigoli: 33550;
- Numero facce: 22272.

Si procede all'eliminazione manuale e casuale dei triangoli che formano il reticolo. Per effettuare questa operazione, si prende come modello di input il file `flangia_accoppiata_gt.obj`. Dopo averlo importato su Blender si procede con l'eliminazione delle aree scelte. Osservando il modello di partenza, si nota un campionamento fitto dei vertici della mesh, cosa che rende superflua una fase preliminare di remeshing e decimazione del reticolo e permette di eseguire l'operazione di rimozione delle facce in maniera più controllata e precisa.

La figura sopra mostra la mesh original con le relative parti mancanti del reticolo.

2. Mesh ricostruita tramite scansione 3D:

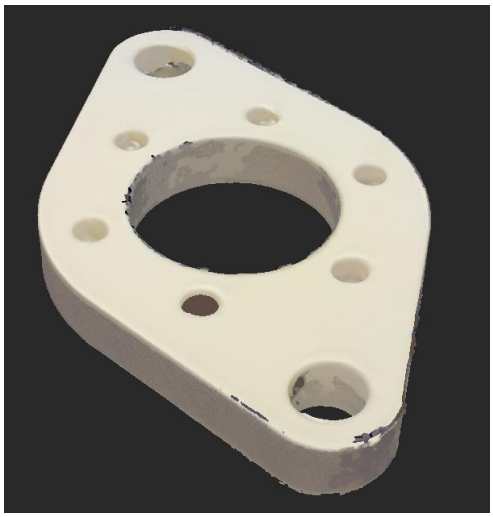


Figura 118. Original scanner 3D.

In figura è possibile osservare il risultato finale della scansione 3D tramite Revopoint mini e della elaborazione sulla nuvola di punti acquista tramite il software Revoscan 5.

Come accaduto per i pezzi meccanici analizzati nei test precedenti, anche in questo caso occorre effettuare due scansioni parziali del componente che in seguito vengono unite per formare la geometria totale definitiva. Questa unione viene mostrata nell'immagine seguente. I punti indicati servono per creare una corrispondenza nell'allineamento delle due parti.

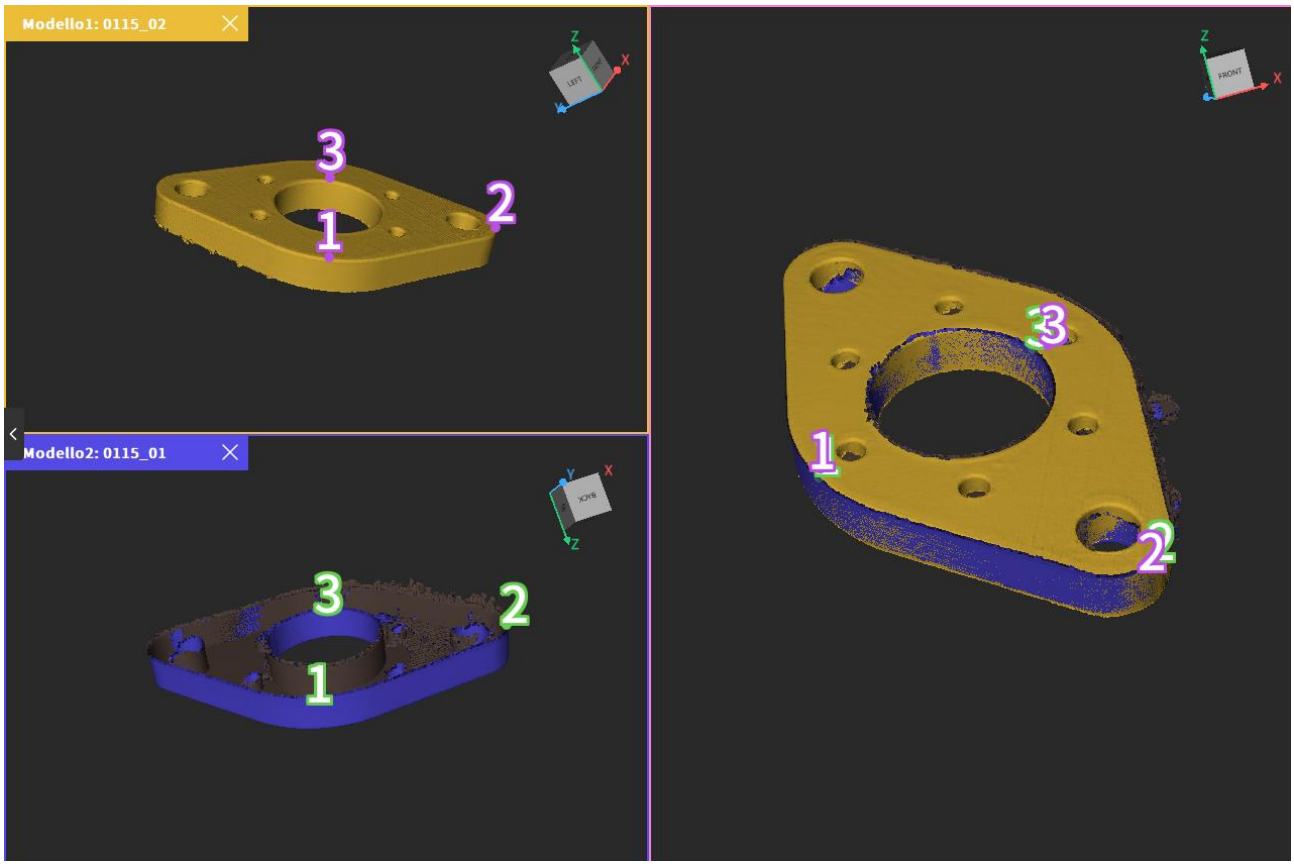


Figura 119. Unione delle scannerizzazioni parziali.

Dopo aver ricostruito la geometria complessiva, si effettua una successiva operazione di remeshing e decimazione per rendere idoneo il file ottenuto alla computazione della rete neurale. L'output finale è presentato nella figura seguente:

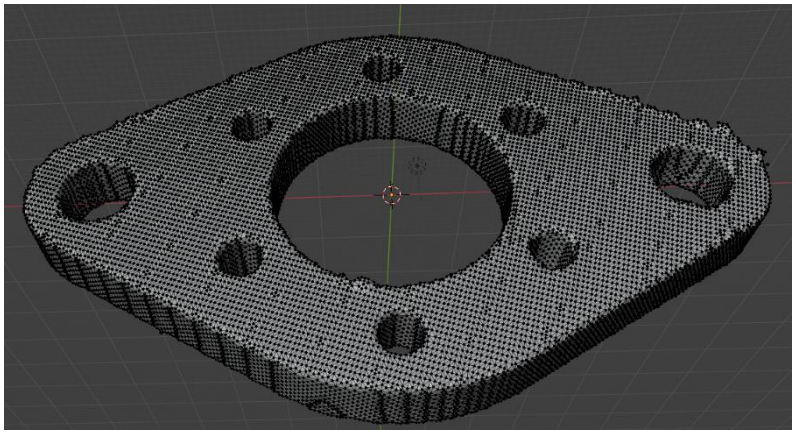


Figura 120. Remesh finale.

Caratteristiche della mesh poligonale:

- Numero vertici: 25202;
- Numero spigoli: 75642;
- Numero facce: 50426.

Rispetto alle scansioni passate, questo file presenta una geometria avente bordi meno definiti, infatti in alcuni punti si possono osservare persino delle distorsioni marcate (in alto a destra). Queste distorsioni vengono parzialmente livellate nella fase di smoothing della rete.

7.5.3 Risultati

I file CAD `original` generati nella fase precedente, sono dati ora in input per essere processati dalla rete neurale SeMIGCN. Al fine di poter confrontare le prestazioni della rete neurale sui due file di input, il training viene effettuato utilizzando gli stessi parametri definiti in precedenza. In questa sezione vengono visualizzati i modelli relativi al termine della procedura di training, corrispondenti quindi all'epoca 1000.

1. Rottura manuale della mesh:

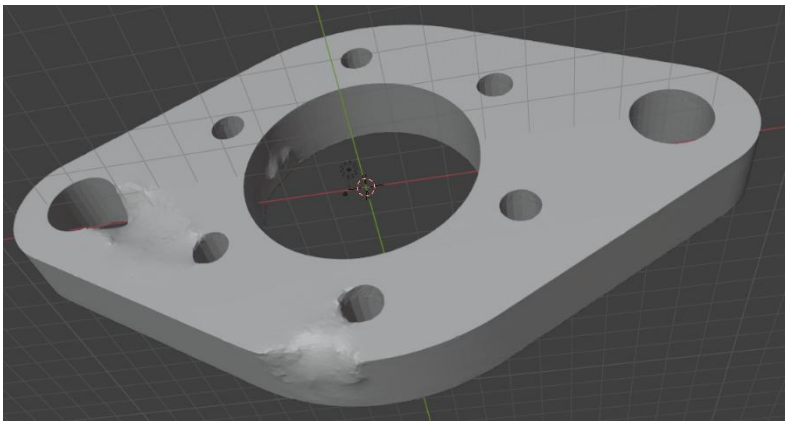


Figura 121. Output rottura manuale.

Caratteristiche della mesh poligonale:

- Numero vertici: 42114;
- Numero spigoli: 126390;
- Numero facce: 84260.

Data l'ampia regione asportata la rete neurale ha riparato la mancanza di reticolo producendo una distorsione marcata su bordo del foro a sinistra, mentre la forma del foro piccolo centrale è stata ricostruita quasi perfettamente dato che l'eliminazione delle facce lo ha toccato solo marginalmente, facendo rimanere intatta la superficie

interna del foro. Lo spigolo vivo esterno invece è stato riparato perdendo l'ortogonalità delle due superfici.

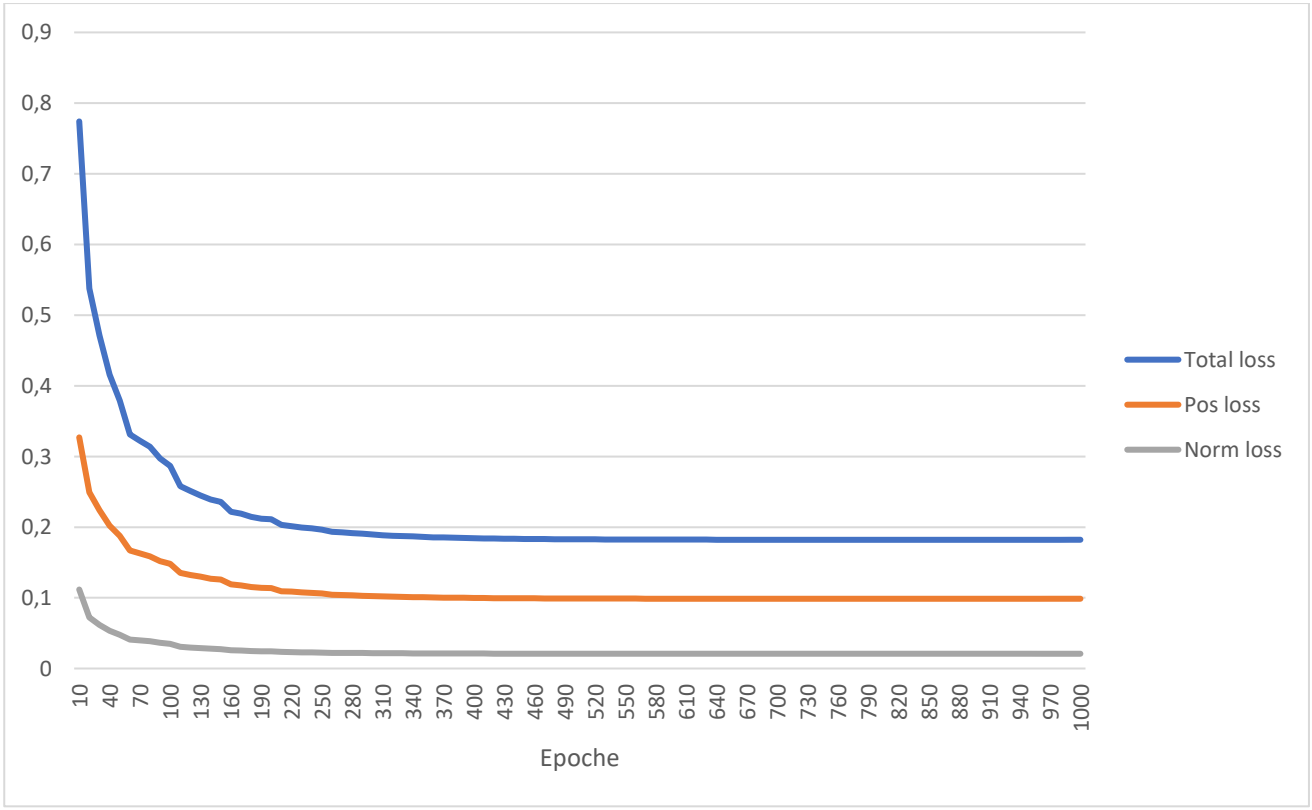


Figura 122. Andamento loss.

2. Mesh ricostruita tramite scansione 3D:

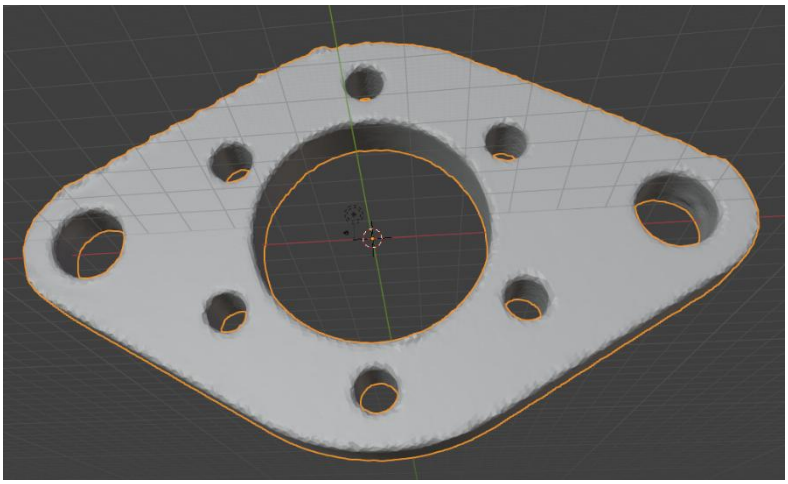


Figura 123. Output scannerizzazione 3D.

Caratteristiche della mesh poligonale:

- Numero vertici: 32201;
- Numero spigoli: 96651;
- Numero facce: 64434.

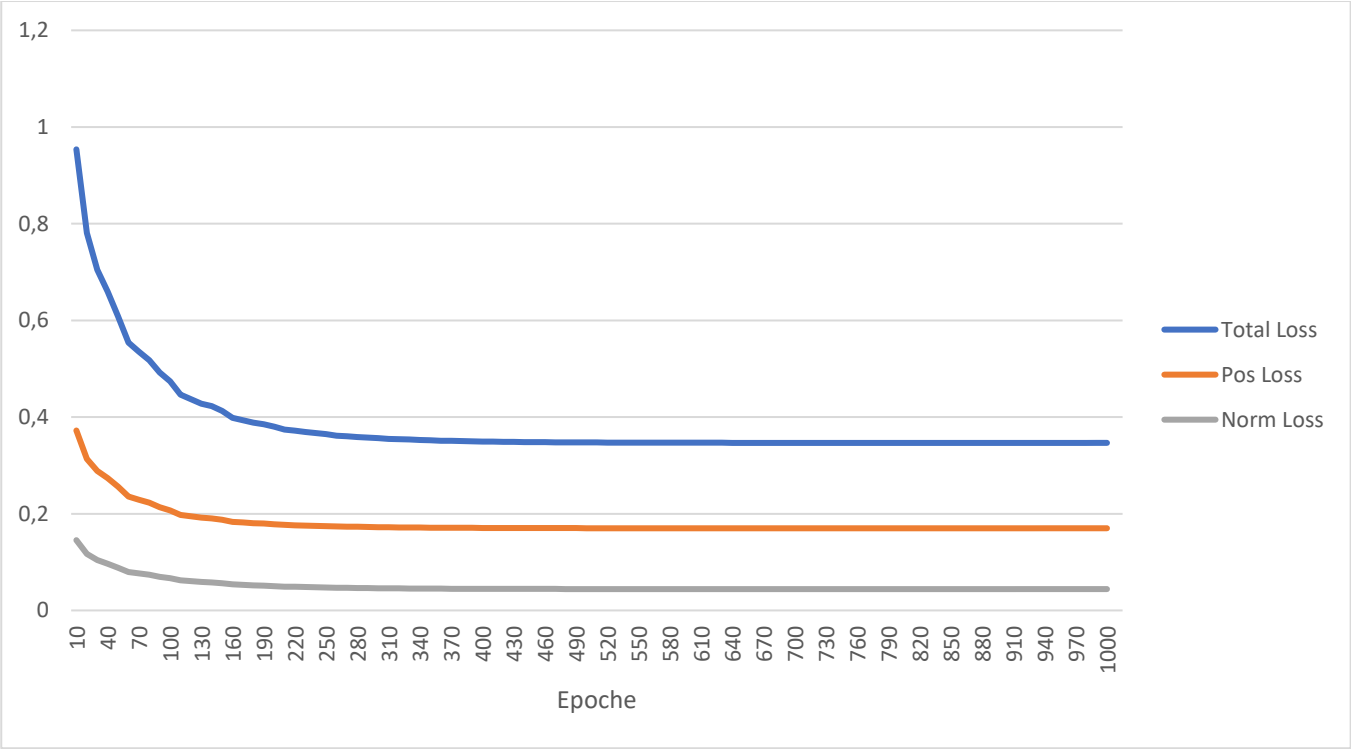


Figura 124. Andamento loss.

7.5.4 Valutazioni

Dai risultati riscontrati al termine dell'esecuzione di questo test, sono nuovamente emerse le problematiche analizzate nei casi studio trattati in precedenza, relativi alla quantità di vertici asportati. Infatti si può notare come – nel caso di rottura manuale del reticolo – i bordi dei fori e gli spigoli vivi hanno risentito maggiormente delle modifiche apportate. Solamente il foro a diametro inferiore in cui è stata asportata un'area più contenuta ha offerto risultati soddisfacenti.

Per quanto riguarda il modello relativo alla scansione 3D, la presenza di rumore a bassa frequenza, identificabile con le piccole distorsioni del reticolo in alcuni bordi esterni, ha impattato sulla qualità del risultato finale, mostrando tuttavia un buon livello di ricostruzione. La presenza di questo rumore, si può limitare solo in fase di scansione, ripetendola, oppure utilizzando uno strumento più performante.

Vengono di seguito riportati i dati relativi al confronto tra i modelli, tramite MeshLab.

Confronto	Distanza minima [mm]	Distanza massima [mm]	Distanza media [mm]	RMS [mm]
A-C	0,763262	0,763262	-0,1865	0,272241
A-D	1,792462	1,792462	0,628394	0,762012
B-D	1,30896	1,30896	-0,226908	0,755847
A-B	1,121039	1,121039	-0,214478	0,692256

1) Caso AC: il modello ottenuto in output dalla rete presenta una buona ricostruzione, tuttavia presenta maggiori distorsioni in corrispondenza dei difetti creati dalla rottura della mesh.

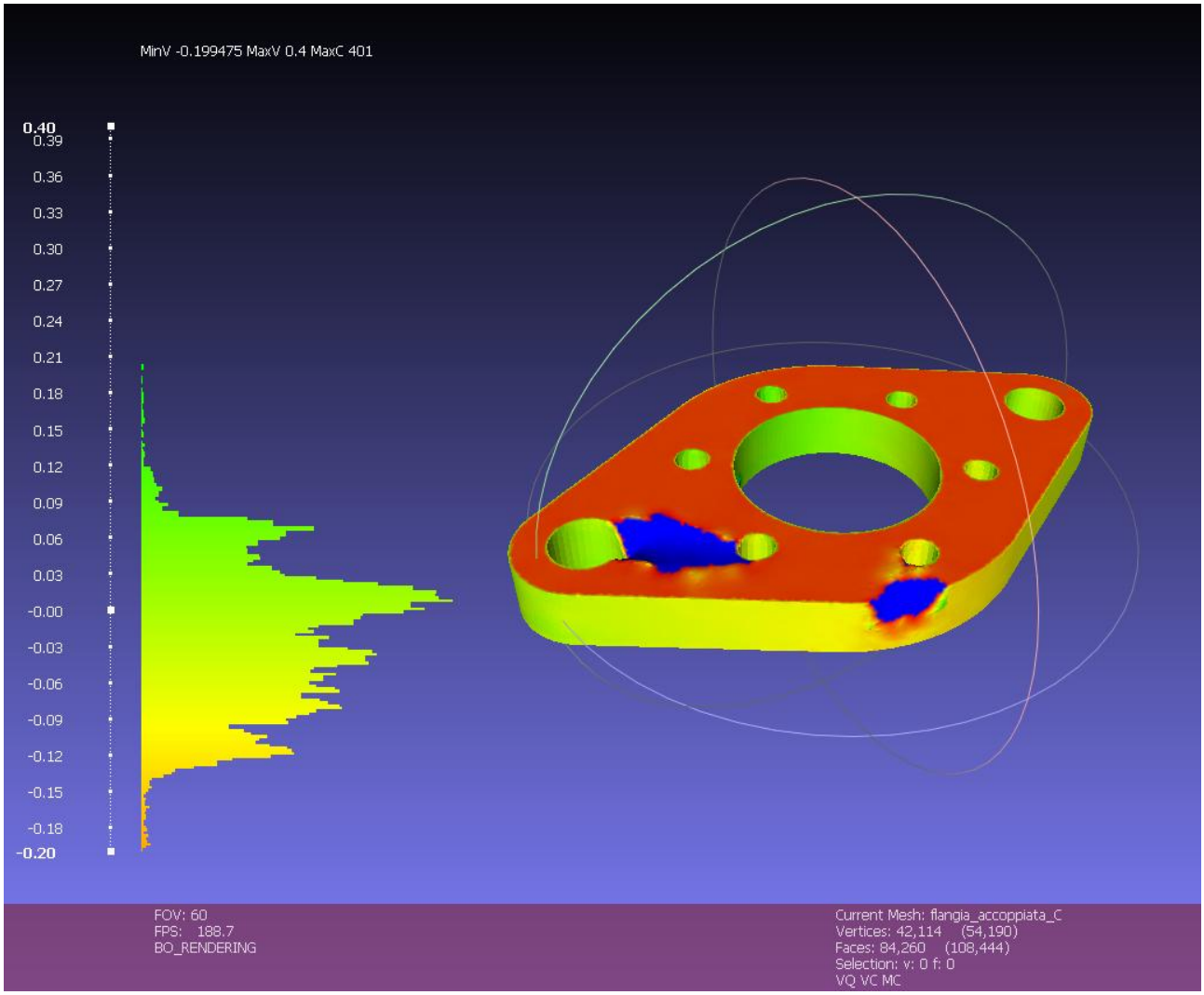


Figura 125. Confronto AC.

2) Caso AD: il modello ottenuto al temine della fase di training si discosta leggermente dal ground truth, a causa della bassa qualità ottenuta dalla scansione. Ciò viene evidenziato dalla presenza di rumore a bassa frequenza nei bordi.

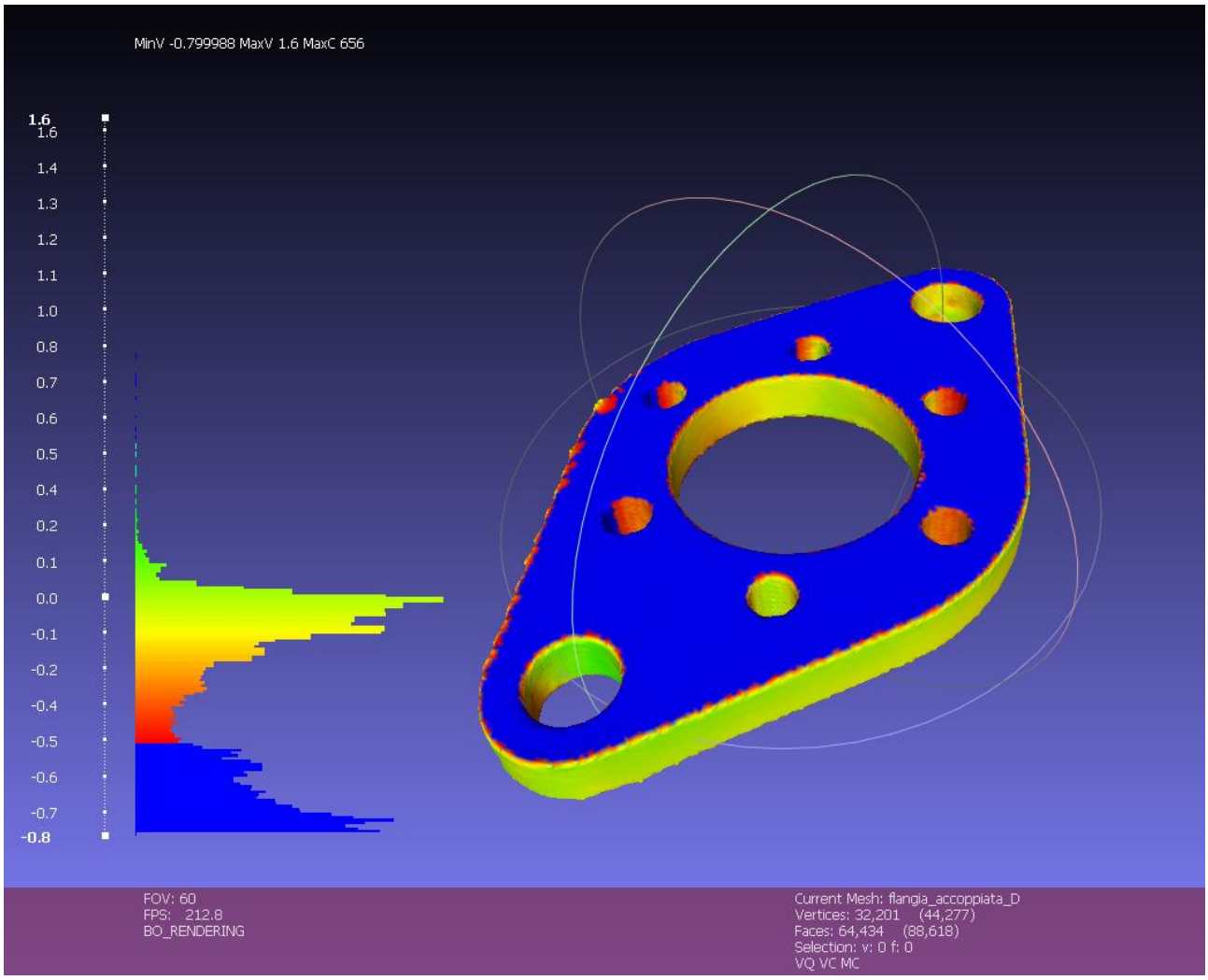


Figura 126. Confronto AD.

3) Caso BD: come riscontrato nei casi studio precedenti, la rete neurale rispetta le dimensioni del pezzo presentando però scostamenti più alti. Queste variazioni si presentano maggiormente nella superfice interna dei fori, come evidenziato dal colore blu.

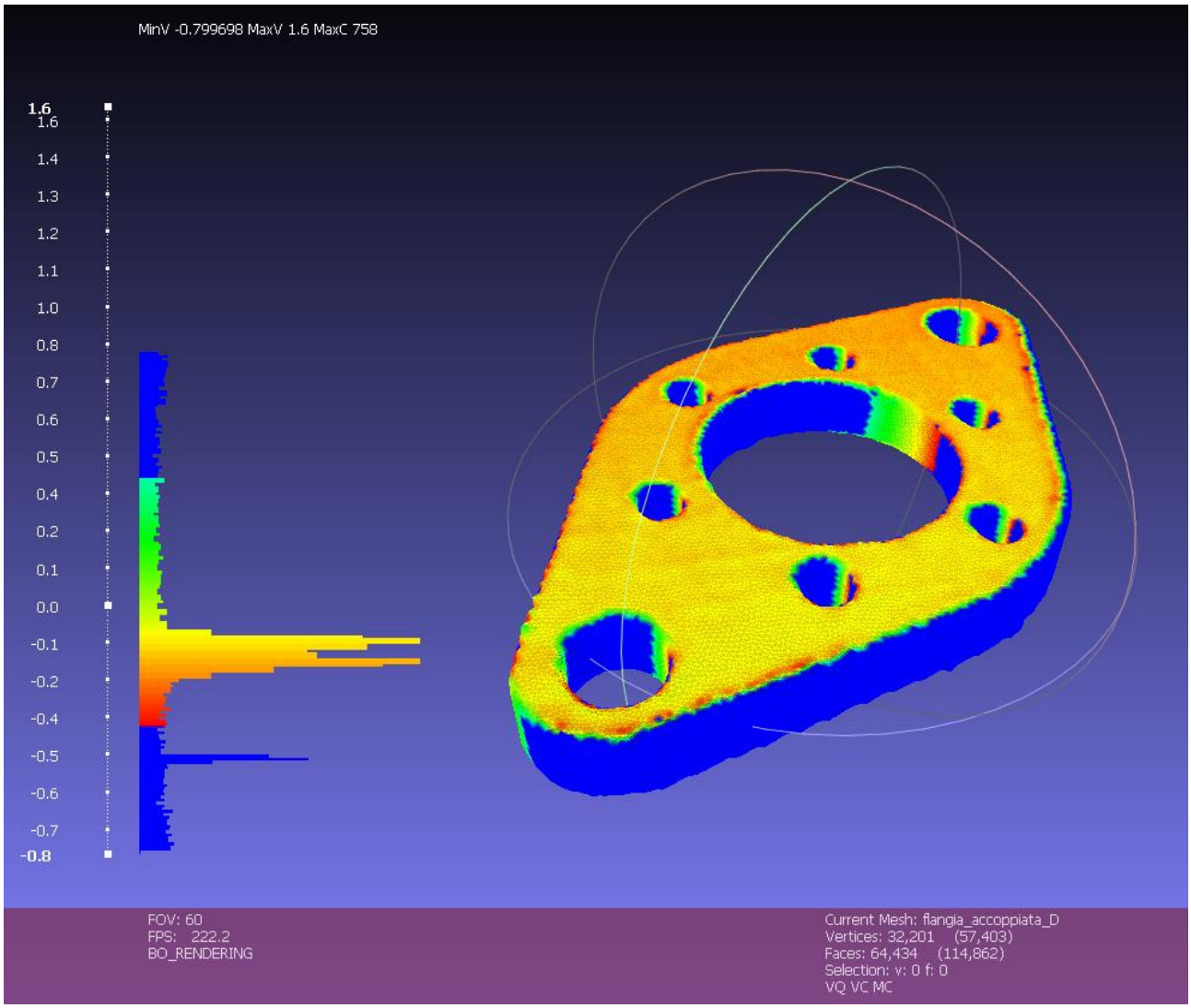


Figura 127. Confronto BD.

4) Caso AB: il file di output della scansione, presenta variazioni marcate rispetto al modello target, secondo una distribuzione non uniforme. La figura mostra la maggiore facilità di acquisizione dello scanner della superficie piana rispetto ai bordi interni.

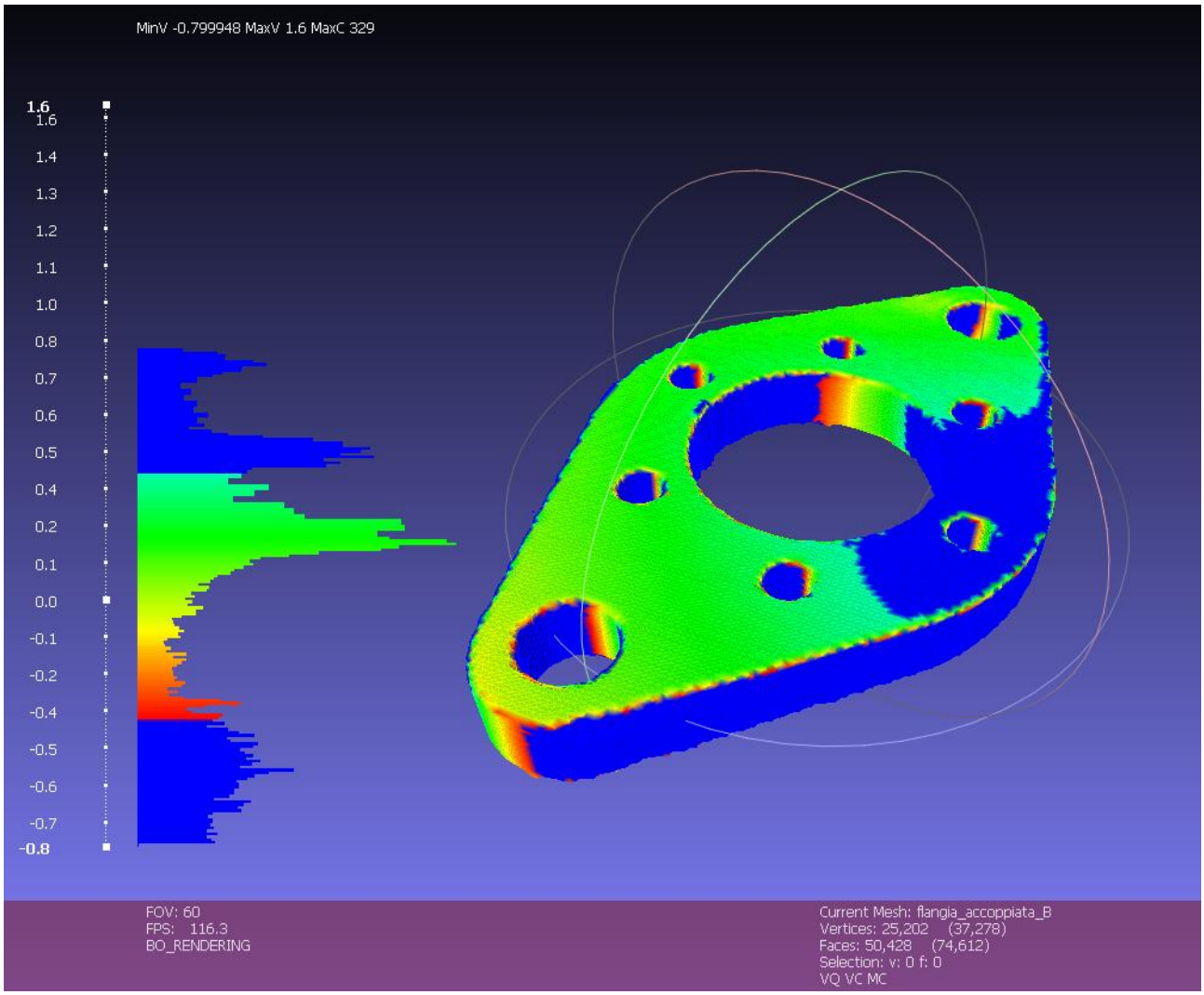


Figura 128. Confronto AB.

7.6 Morsetto da banco

In questa sezione si analizza in maniera approfondita il componente meccanico definito con il nome `morsetto_da_banco`. Per questo caso studio si effettuano vari test per verificare se i parametri trovati tramite la sperimentazione su componente `flangia_forata`, risultano attendibili.

Test	Mesh_gt	Mesh_original	Tipologia	Learning rate	N°epoche	Batch size	k1	k2	kn	Dm_size	Drop rate	loss
Test1	33716	50938	real	0,005	100	5	4	4	4	40	0,6	0,432
Test2	33716	50938	real	0,005	1000	5	4	4	4	40	0,6	0,344
Test3	33716	31912	real	0,005	1000	5	4	4	4	40	0,6	0,206
Test4	33716	33004	real	0,005	1000	5	4	4	4	40	0,6	0,211

Come si osserva dai dati riportati in tabella, i test sono stati condotti lasciando inalterato il settaggio della tipologia e il learning rate, variando invece il numero di epoche e il campionamento della mesh poligonale. Il Test1 e Test2 sono stati condotti utilizzando un file `original` proveniente dalla scansione 3D del pezzo. Al contrario il Test3 3 Test4 sono stati eseguiti utilizzando un file di input proveniente dalla rottura della mesh del ground truth.

7.6.1 Morsetto da banco ground truth

In maniera analoga ai casi studio analizzati in precedenza, si riporta il modello CAD di riferimento per il morsetto da banco. Il file in questione in formato .obj viene salvato come da nomenclatura con il nome `morsetto_da_banco_gt.obj`.

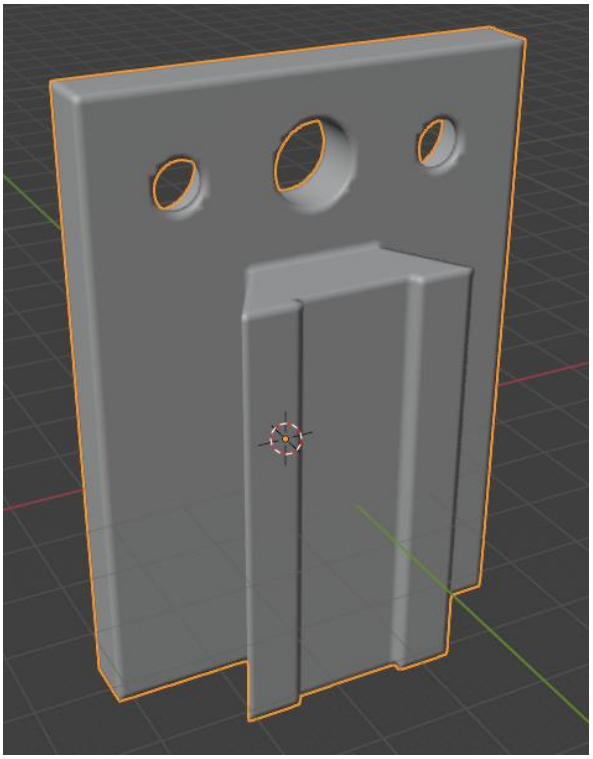


Figura 129. Ground truth.

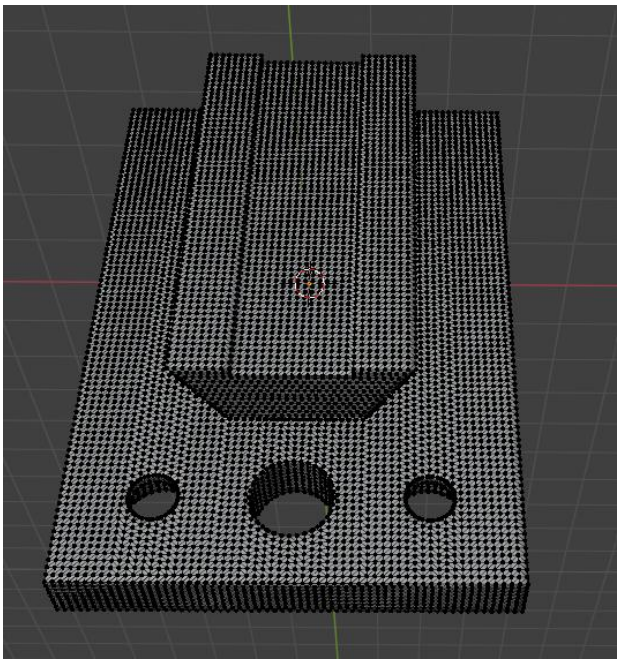


Figura 130. Ground truth, in formato mesh.

Caratteristiche della mesh poligonale:

- Numero vertici: 16854;
- Numero spigoli: 50574;
- Numero facce: 33716.

La mesh triangolare viene generata al termine della modellazione del componente sul software Creo 11, ponendo come target un valore intorno ai 30000 triangoli. Come si può vedere dall'immagine sopra, il campionamento risulta abbastanza fitto pur presentando un numero non elevatissimo di triangoli a beneficio del consumo di memoria.

7.6.2 Morsetto da banco original

Si presentano in questa sezione due modelli CAD in formato .obj e aventi mesh incompleta, i quali costituiscono l'input della rete neurale SeMIGCN. In maniera analoga ai casi studio precedenti, entrambi i file vengono denominati `morsetto_da_banco_original.obj`, in questo modo risulta possibile caricare i file all'interno degli algoritmi di elaborazione della rete.

1. Rottura manuale della mesh:

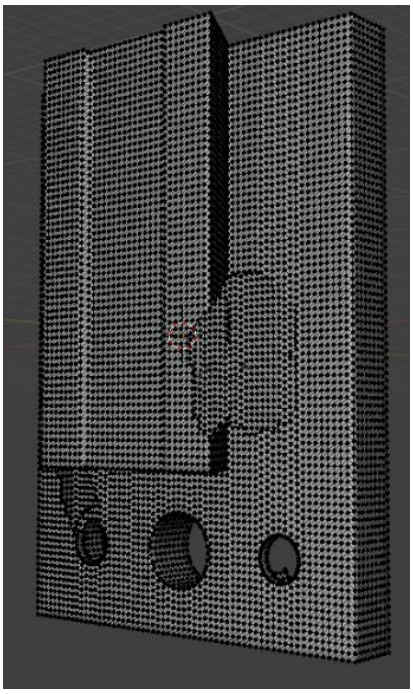


Figura 131. Original mesh poligonale.

Caratteristiche della mesh poligonale:

- Numero vertici: 16073;
- Numero spigoli: 47993;
- Numero facce: 31912.

A partire dal modello di riferimento, `morsetto_da_banco_gt.obj`, si effettua l'eliminazione di alcune regioni della mesh prese in maniera casuale, scegliendo tuttavia aree in prossimità dei bordi interno e degli spigoli. Inoltre in questo modello risulta interessante la guida a coda di rondine, per tale motivo, l'eliminazione dei triangoli ha interessato anche questa regione, come mostrato in figura.

Dato il campionamento effettuato su Creo, risultano superflue ulteriori operazioni di remeshing e decimazione, avendo già ottenuto un livello di approssimazione della superfice del componente.

2. Mesh ricostruita tramite scansione 3D:

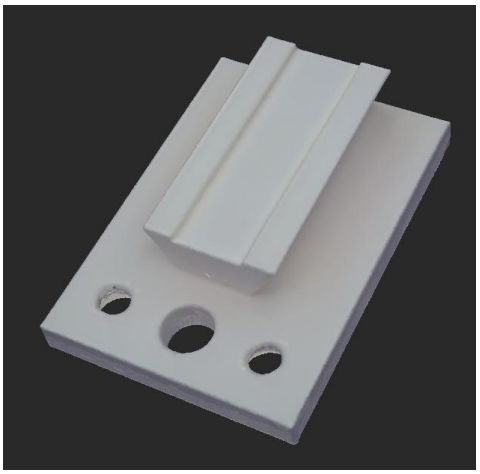


Figura 132. Original scanner 3D.

La figura mostra il risultato finale della scansione 3D del pezzo stampato tramite l'utilizzo dello scanner Revopoint Mini. Il componente 3D è stato digitalizzato mediante l'acquisizione della nuvola di punti e la successiva elaborazione tramite il software Revoscan 5, attraverso il quale viene ricostruita la geometria completa.

Al fine di ottenere una scansione completa per una qualità maggiore dell'output finale, risulta necessario effettuare due scansioni parziali, in quanto alcune caratteristiche della geometria – come il doppio diametro dei fori – non si possono acquisire in un'unica scansione perdendole. L'unione delle due parti è mostrato nella figura seguente:

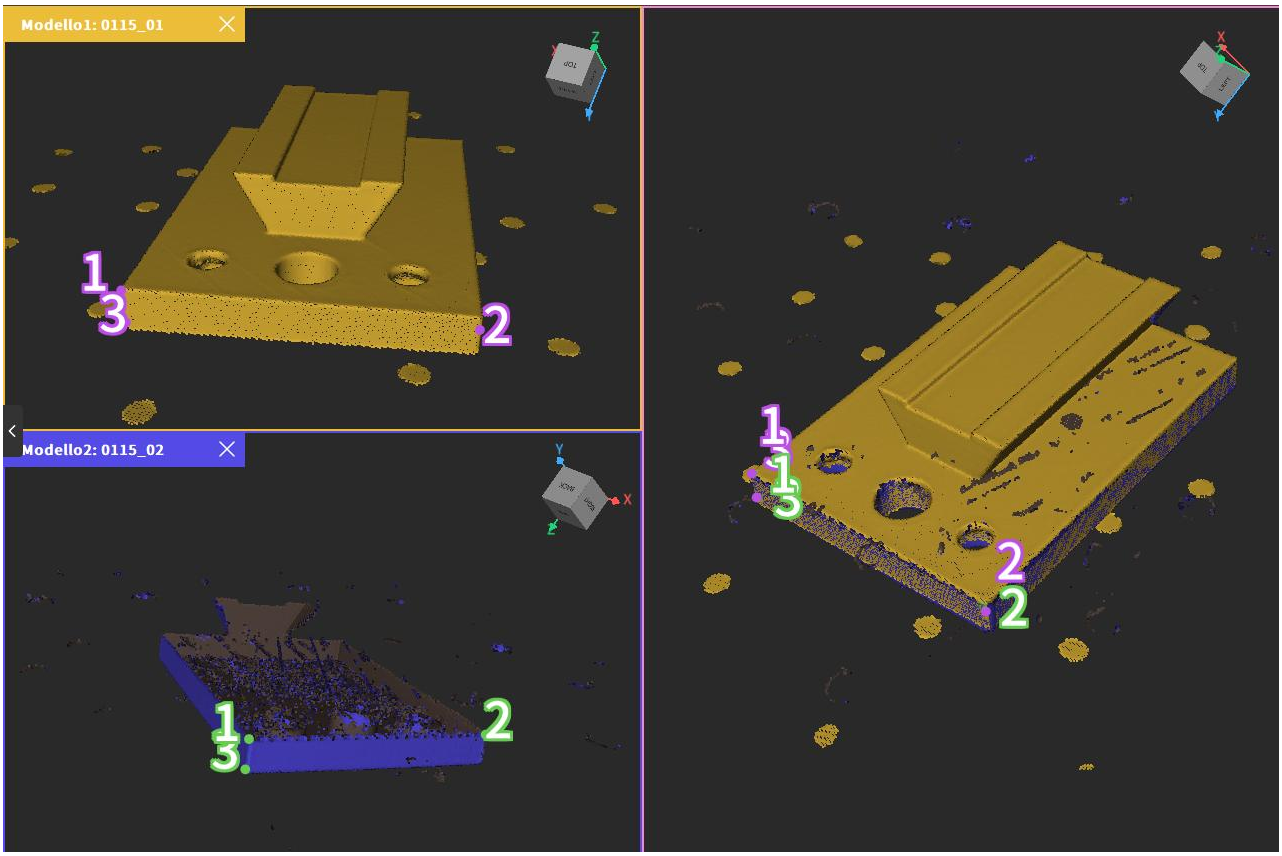


Figura 133. Unione delle scannerizzazioni parziali.

Ottenuta la geometria complessiva, si esporta tramite Revoscan 5 il file in formato .obj e si procede con l'elaborazione della mesh mediante Blender. Nella figura seguente viene mostrato il risultato finale del file `original.obj`:

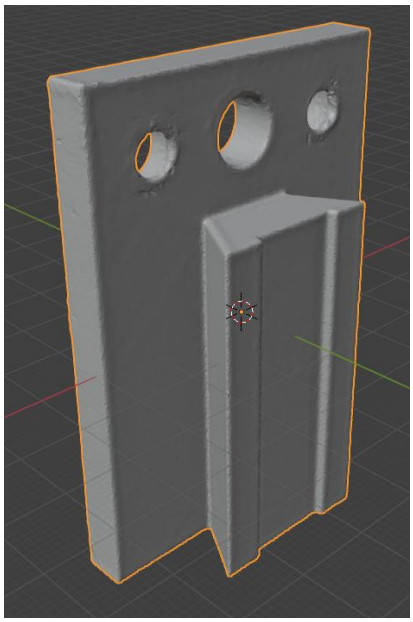


Figura 134. Remesh finale.

Caratteristiche della mesh poligonale:

- Numero vertici: 25419;
- Numero spigoli: 76375;
- Numero facce: 50936.

Il file creato attraverso una prima fase di remeshing seguita da una decimazione ha prodotto una mesh di buona qualità, che comunque presenta difetti superficiali e qualche distorsione, soprattutto nella parte interna dei fori.

7.6.3 Risultati

Terminata la fase di creazione dei file `original` costituenti l'input della rete neurale SeMIGCN, si procede con il training utilizzando l'algoritmo contenuto all'interno del file sgcn.py. In maniera analoga ai casi studio affrontati in precedenza, per poter effettuare un confronto, tra i due modelli CAD dati in pasto alla rete, si settano nell'algoritmo i medesimi parametri trovati in precedenza. Inoltre in questa sezione vengono presentati i modelli CAD di output della fase di training, corrispondenti quindi all'epoca 1000.

1. Rottura manuale della mesh:

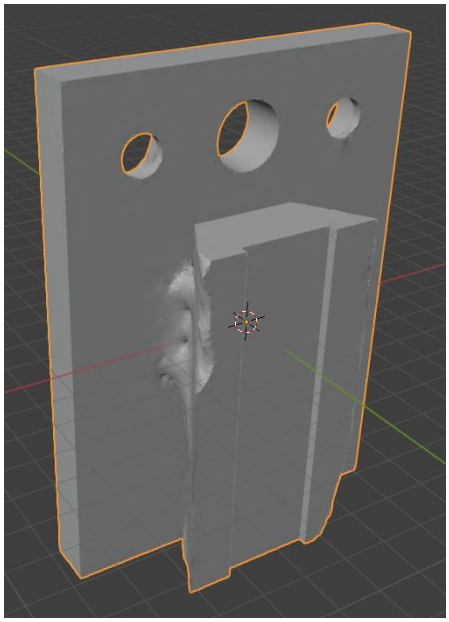


Figura 135. Output rottura manuale.

Caratteristiche della mesh poligonale:

- Numero vertici: 61604;
- Numero spigoli: 184824;
- Numero facce: 123216.

Analizzando la riparazione del reticolo proposta dalla rete neurale in relazione all'area asportata nel corrispondente file `original`, si può osservare dal risultato finale che la spaccatura in prossimità del bordo del foro più a destra risulta di buona qualità pur presentando una lieve distorsione. Per quanto riguarda la parte asportata nella guida a coda di rondine, la riparazione non viene eseguita correttamente, presentando una forma distorta non compatibile con la geometria iniziale. Inoltre anche la rottura del reticolo apportata nella parete inferiore ha generato una deformazione piuttosto marcata del reticolo poligonale, come si vede in figura in basso a destra.

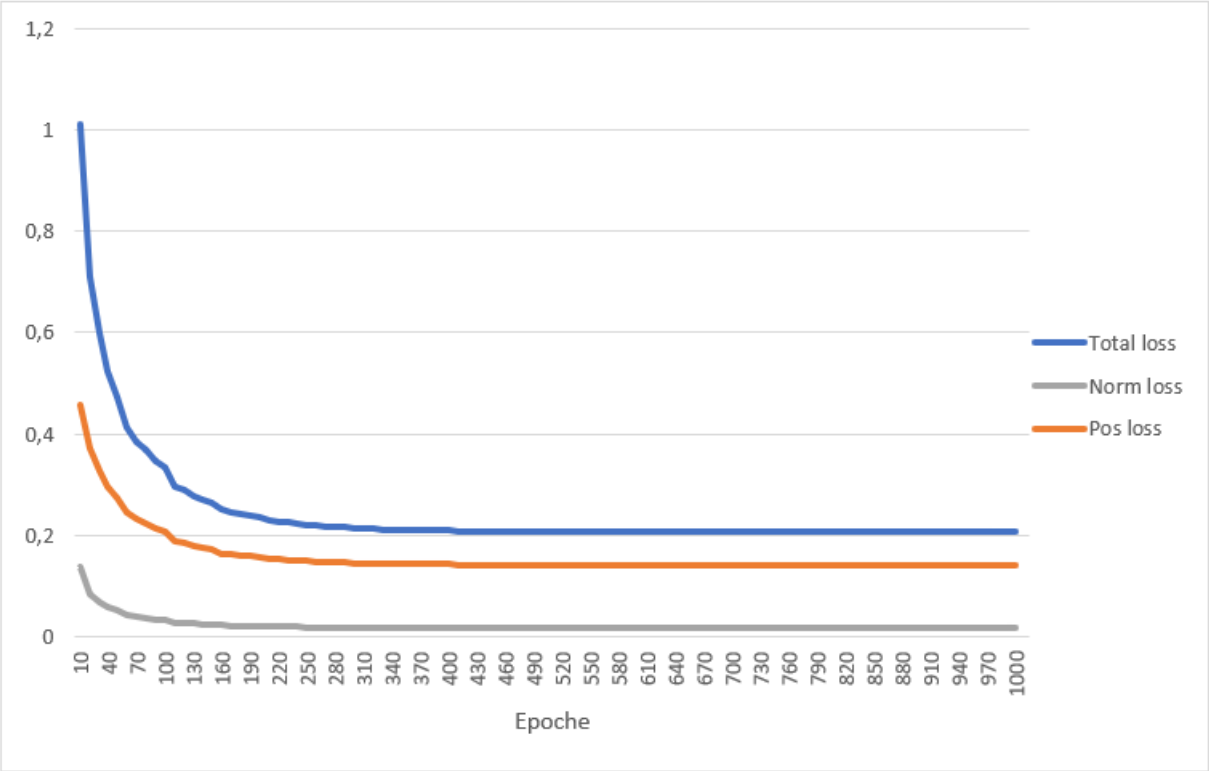


Figura 136. Andamento loss.

2. Mesh ricostruita tramite scansione 3D:

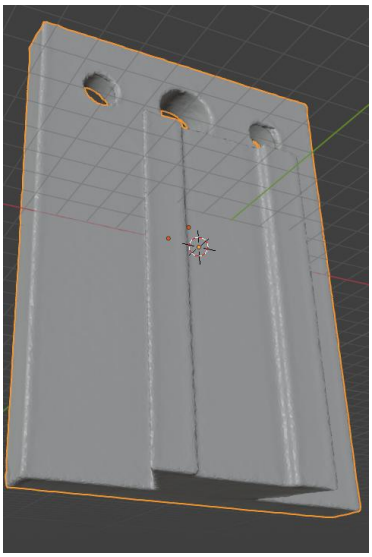


Figura 137. Output scannerizzazione.

Caratteristiche della mesh poligonale:

- Numero vertici: 65727;
- Numero spigoli: 197193;
- Numero facce: 131462.

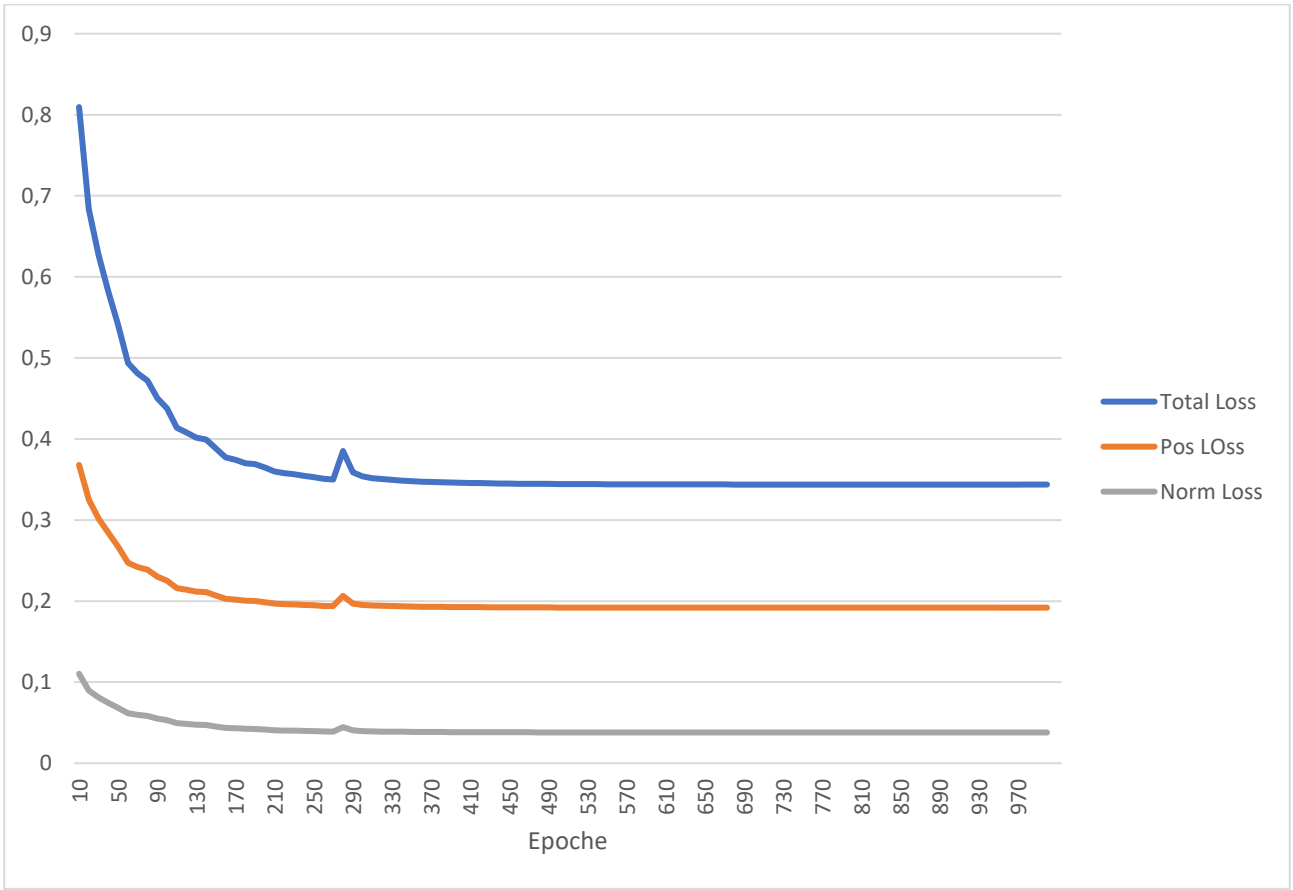


Figura 138. Andamento loss.

Il picco presente all'epoca 280 mette in evidenza l'instabilità della convergenza con il learning rate impostato. L'oscillazione comunque si assorbe fino a presentare un plateau della funzione di loss.

7.6.4 Problematiche riscontrate

Il caso in esame presenta nuovamente la difficoltà della rete neurale nel riparare correttamente una mesh poligonale danneggiata. La spaccatura del reticolo creata tramite Blender risulta essere di grande entità, rendendo così più difficoltosa l'interpretazione della geometria e, di conseguenza, anche la sua riparazione. A ciò si aggiunge la complessità della forma presentata dalla guida a coda di rondine.

Al fine di identificare la causa del problema, in particolare se l'errata ricostruzione della geometria da parte della rete neurale sia dovuta all'entità del danno superficiale creato oppure se sia la configurazione geometrica presentata dal pezzo, è necessario un ulteriore test su questo pezzo meccanico. Per l'esecuzione del nuovo test, si utilizza come mesh di input quella del ground truth, a cui si applicano delle nuove spaccature al reticolo poligonale di dimensione ridotta rispetto alle precedenti. In particolare, non si effettuano rotture sulla parte relativa alla coda di rondine, andando quindi ad agire solo sui fori e sulle facce piane. Nella figura sottostante viene presentato il nuovo modello CAD `original` con le nuove modifiche apportate.

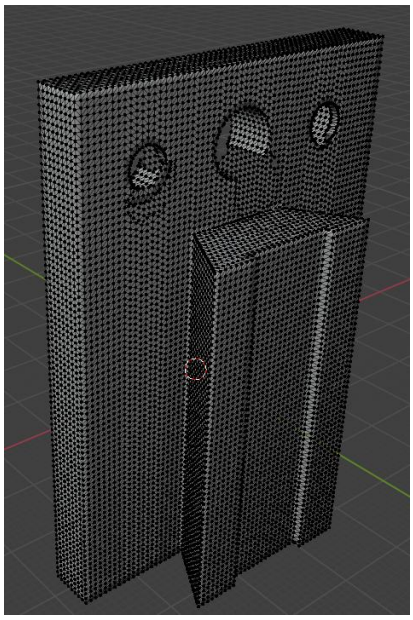


Figura 139. Rottura mesh.

Caratteristiche della mesh poligonale:

- Numero vertici: 16579;
- Numero spigoli: 49591;

- Numero facce: 33004.

Al termine della fase di training si ottiene il seguente risultato:

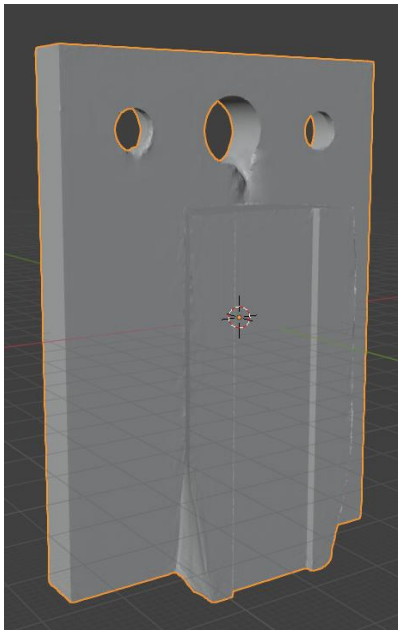


Figura 140. Output rottura manuale remesh.

Caratteristiche della mesh poligonale:

- Numero vertici: 61172;
- Numero spigoli: 183528;
- Numero facce: 122352.

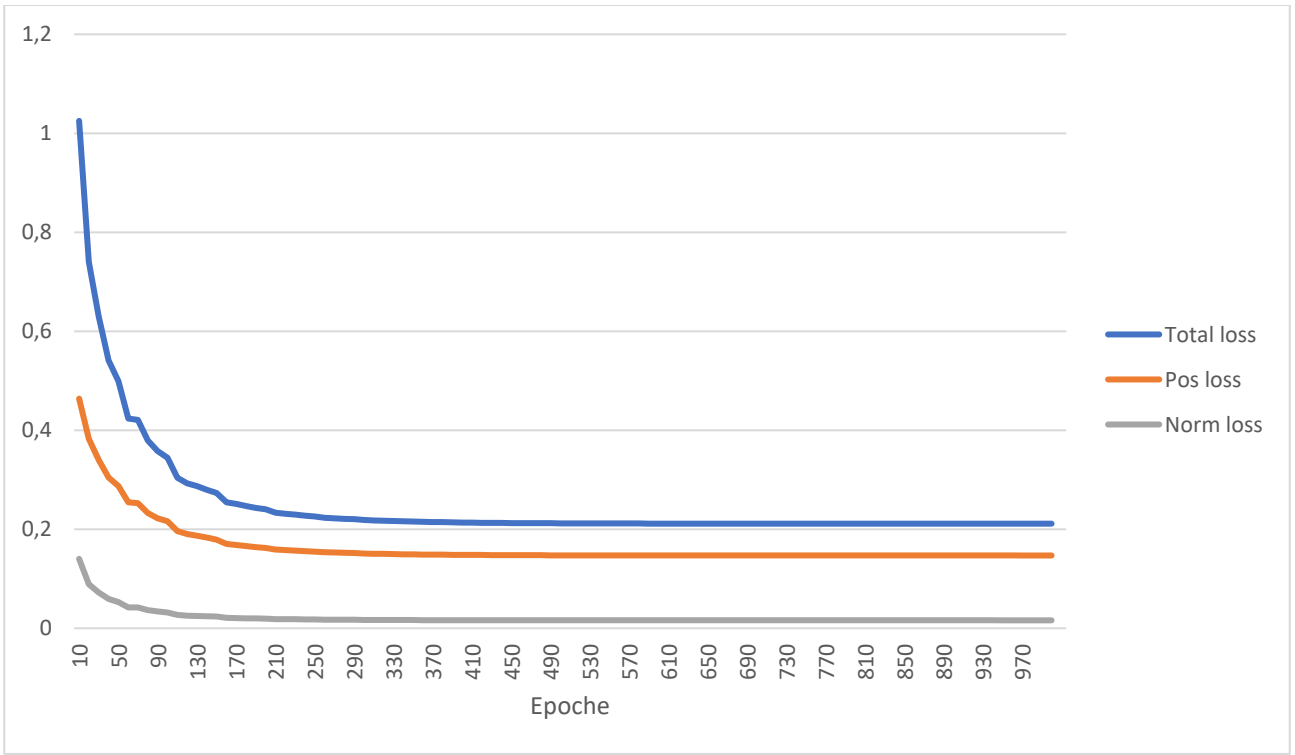


Figura 141. Andamento loss remesh.

Dai risultati ottenuti con il nuovo test si riscontra un miglioramento in termini visivi ottenendo un modello con una forma meno deformata. Tuttavia, dai grafici ottenuti, si osserva un infinitesimo aumento del loss della dimensione di qualche millesimo. Considerando la funzione di perdita, si possono ritenere i due test relativi alla spaccatura manuale della mesh quasi identici.

7.6.5 Valutazioni

Come si può osservare dal modello CAD di output della fase di training, le modifiche apportate al modello di input hanno portato ad un aumento della qualità, in quanto la geometria risulta più similare al ground truth. Tuttavia, nonostante non siano state create delle spaccature nella coda di rondine, questa risulta ancora deformata, andando a perdere progressivamente l'inclinazione delle facce dai fori fino all'estremità.

La forma della guida a coda di rondine risulta letta correttamente dalla rete neurale solamente nel file `original` provenite da scansione 3D, nel quale la mesh presenta solo lievi difetti superficiali, ma non vi sono parti mancanti del reticolo. Ne consegue che in presenza di geometrie complicate come quella in esame, la rete neurale necessita di una mesh chiusa per fornire una buona qualità del modello di output. Può però riparare piccoli difetti superficiali.

Sono di seguito riportati i risultati derivanti dal confronto tra i componenti, tramite MeshLab.

Confronto	Distanza minima [mm]	Distanza massima [mm]	Distanza media [mm]	RMS [mm]
A-C	1,099343	1,099343	-0,083586	0,141832
A-D	0,745759	0,745759	-0,196215	0,390465
B-D	0,512466	0,512466	0,002567	0,012124
A-B	1,068672	1,068672	-0,175053	0,303407

1) Caso AC: la figura mostra la una minima distorsione generale della geometria, con valori dell'ordine del centesimo di millimetro. Il difetto risulta localizzato nella coda di rondine, identificabile dall'area blu.

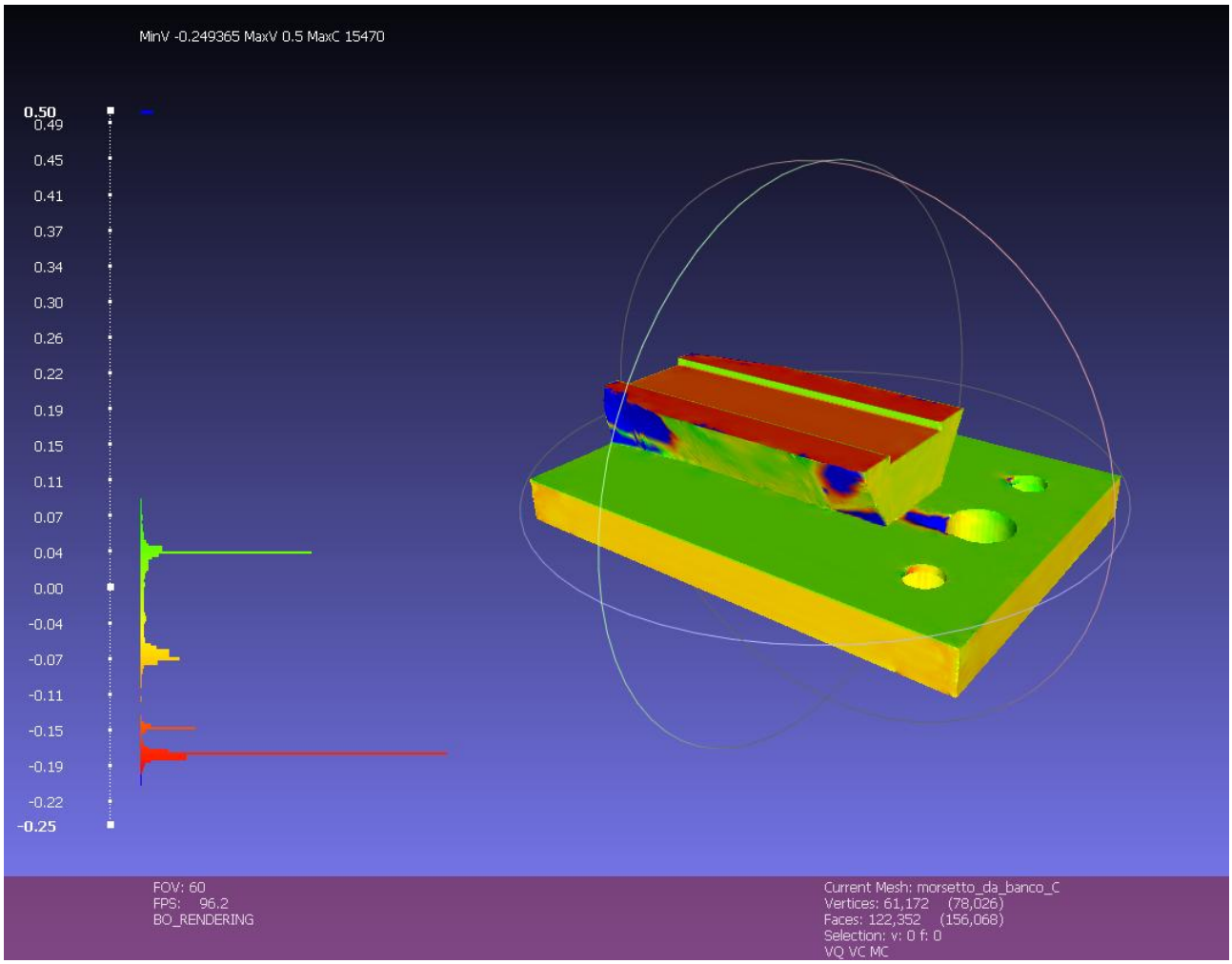


Figura 142 Confonto AC.

2) Caso AD: l'output relativo alla fase di training mostra una buona ricostruzione da parte della rete soprattutto nella guida a coda di rondine. Il resto della geometria presenta scostamenti contenuti non apprezzabili.

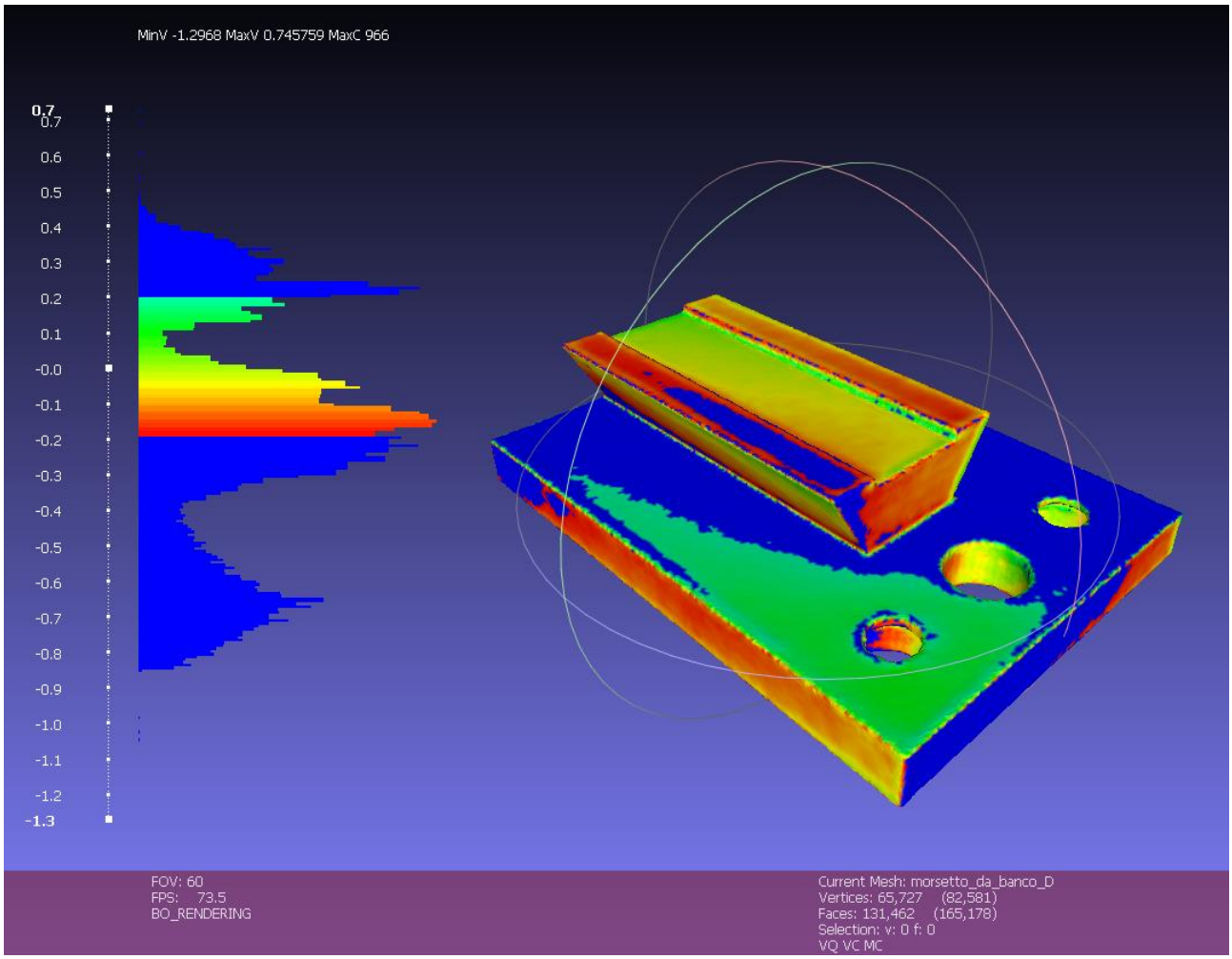


Figura 143. Confronto AD.

3) Caso BD: come riscontrato nei casi studio precedenti, l'alta qualità della scansione 3D, ha portato ad avere un modello estremamente preciso, nel quale la rete ha mantenuto pressoché inalterate le dimensioni del modello CAD datogli in ingresso.

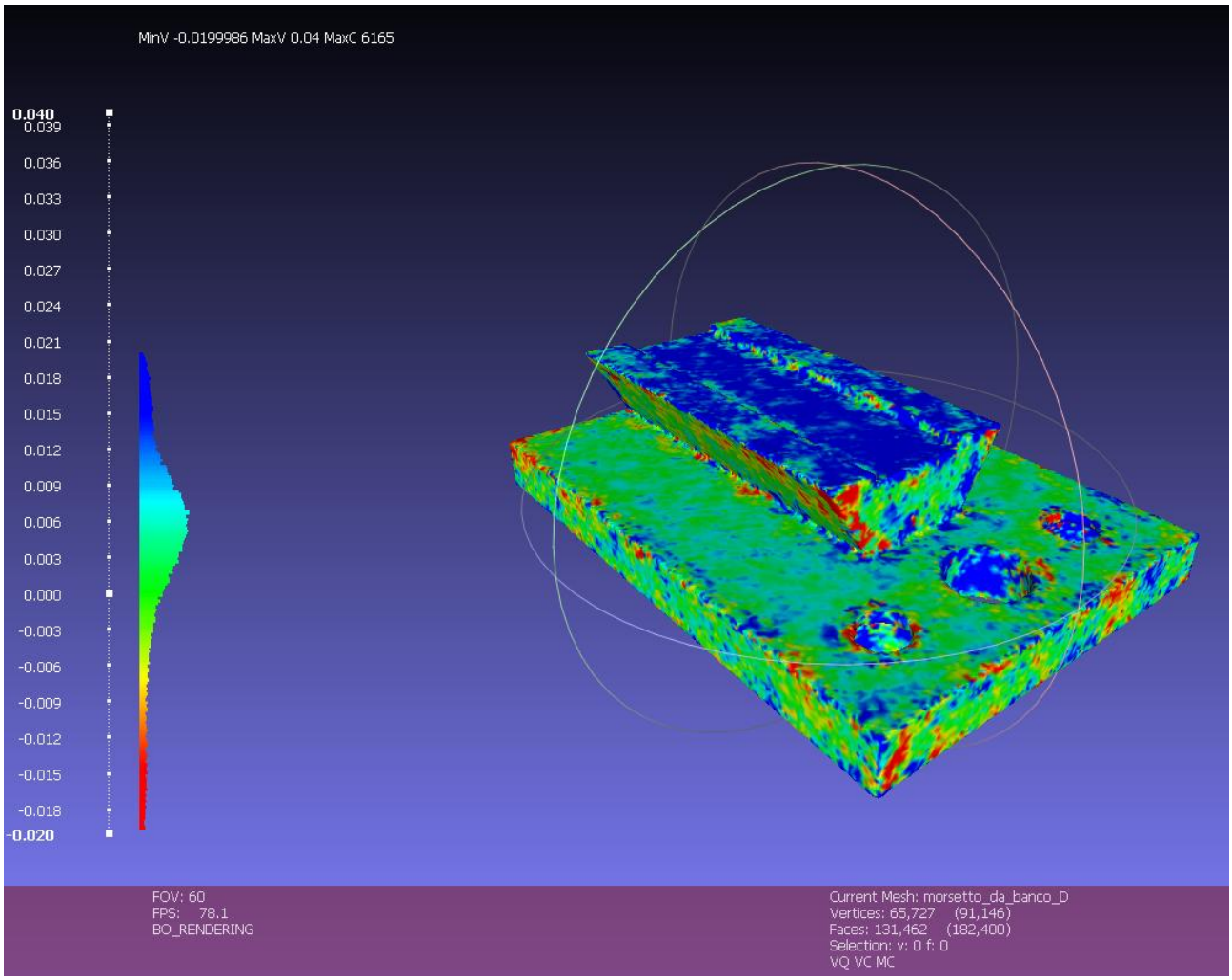


Figura 144. Confronto BD.

4) Caso AB: la figura sotto riportata mostra l'alta qualità del modello original ottenuto dalla scansione 3D presentando scostamenti contenuti. Le maggiori differenze si riscontrano nella parte superiore della guida a coda di rondine.

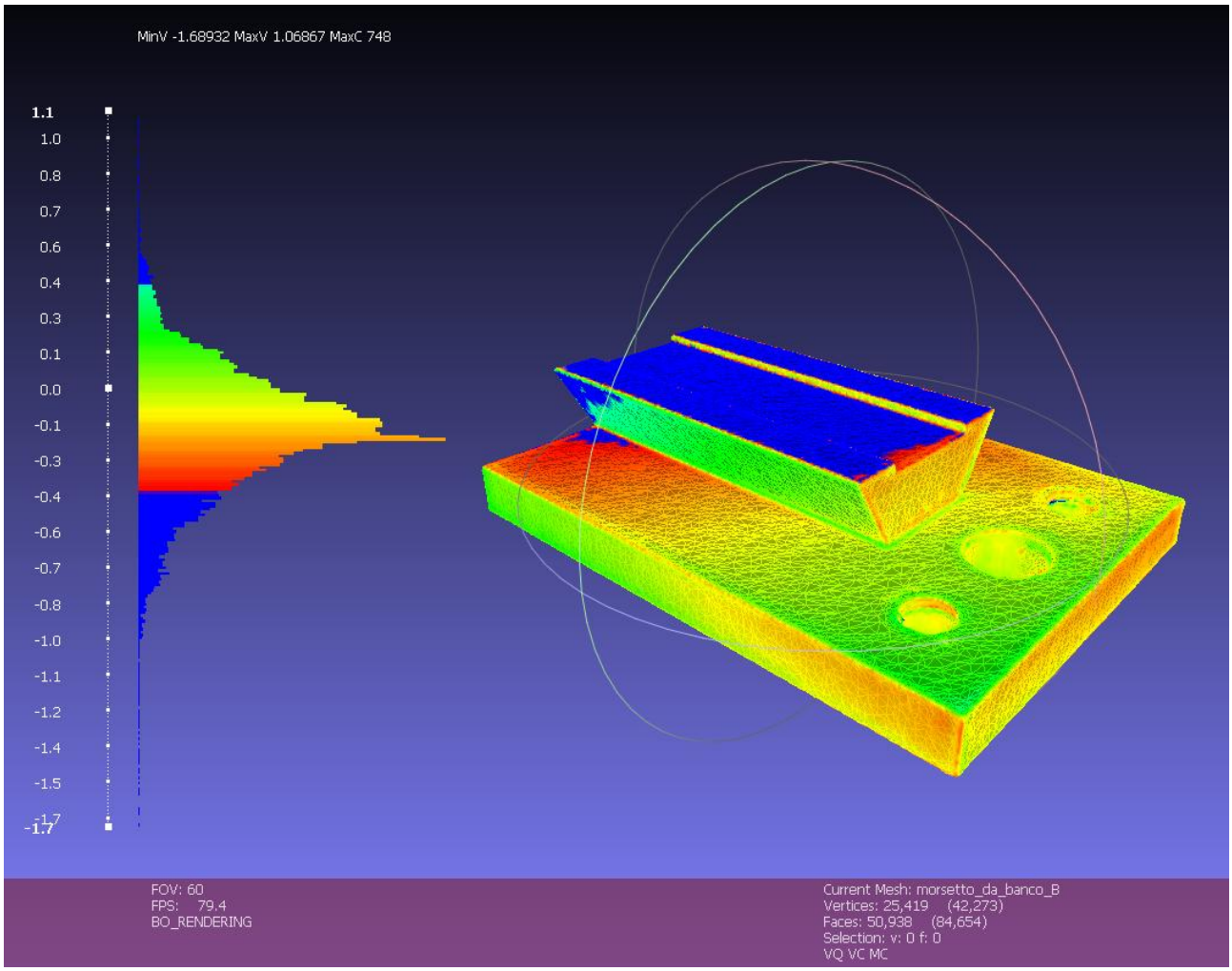


Figura 145. Confronto AB.

7.7 Camma

Si procede con l'analisi dei componenti meccanici, approfondendo in questa sezione il caso studio `camma`. In maniera analoga ai casi precedenti e come riportato in tabella si effettuano dei test, al fine di verificare i parametri di ottimo determinati tramite la sperimentazione della `flangia_forata`.

Test	Mesh_gt	Mesh_original	Tipologia	Learning rate	N°epoche	Batch size	k1	k2	kn	Dm_size	Drop rate	loss
Test1	4384	64376	real	0,005	100	5	4	4	4	40	0,6	0,505
Test2	4384	64376	real	0,005	1000	5	4	4	4	40	0,6	0,4037
Test3	4384	54228	real	0,005	1000	5	4	4	4	40	0,6	0,5173

I test sono stati condotti lasciando inalterati i parametri di learning rate e la tipologia.

I file `original` utilizzati nei primi due test provengono dalla scansione 3D del pezzo meccanico, mentre il Test3 è il risultato della rottura manuale del poligono. Come si può osservare dai valori del loss riportati in tabella, l'aumento del numero di epoche di un ordine di grandezza rispetto al valore iniziale, ha portato ad un valore di loss sensibilmente inferiore. Si può inoltre constatare che, come riscontrato per tutti i pezzi meccanici analizzati in precedenza, avere una mesh poligonale più fitta impatta maggiormente nella diminuzione della loss.

7.7.1 Camma ground truth

In figura si riporta il modello CAD di riferimento, il quale viene salvato in formato .obj secondo la nomenclatura utilizzata con il nome di `camma__gt.obj`.

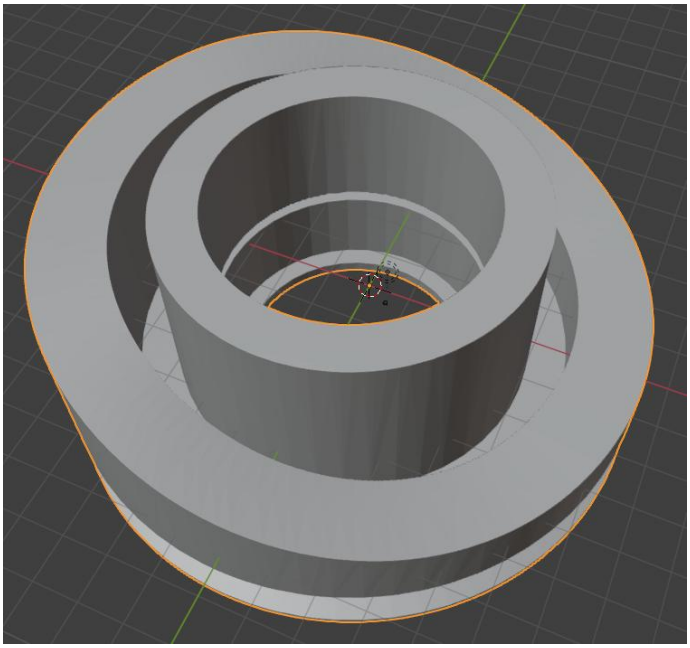


Figura 146. Ground truth.

Caratteristiche della mesh poligonale:

- Numero vertici: 2192;
- Numero spigoli: 6576;
- Numero facce: 4384.

Il numero di facce risulta notevolmente inferiore rispetto ai casi precedenti, presentando una mesh poligonale non eccessivamente fitta. Su Blender non sono state eseguite operazioni di remeshing e decimazione su questo modello CAD, in quanto il componente viene fornito esternamente mediante file Creo in formato .stl.

7.7.2 Camma original

In maniera analoga ai casi studio presentati in questo elaborato, si procede con la creazione dei file CAD di input della rete neurale SeMIGCN.

Al fine di poter essere correttamente letti ed elaborati dagli algoritmi di elaborazione forniti da GitHub, i modelli CAD in questione vengono salvati con il nome di `camma__original.obj`, in accordo con la designazione definita nei capitoli precedenti.

1. Rottura manuale della mesh:

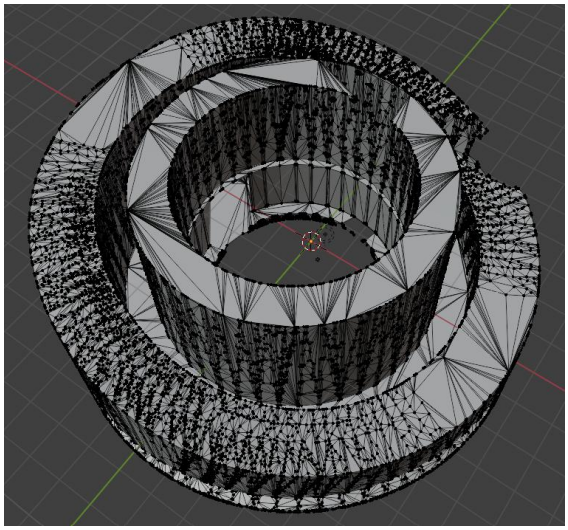


Figura 147. Original mesh poligonale.

Caratteristiche della mesh poligonale:

- Numero vertici: 27233;
- Numero spigoli: 81459;
- Numero facce: 54226.

Per creare questo file `original`, si utilizza come mesh poligonale di partenza quella fornita dal ground truth. Data la complessità delle forma geometrica, per apportare delle rotture del reticolo controllate, risulta necessaria una serie di operazioni preliminari volte a ricavare una nuova mesh più fitta. Vanno inoltre effettuate le spaccature e infine la decimazione del numero di triangoli per ridurre l'onere computazionale. Per quanto riguarda l'ultima operazione si è scelto arbitrariamente un valore dei target nell'intorno di 50000 triangoli.

Nella figura sopra riportata viene mostrato il risultato finale di tale procedura.

2. Mesh ricostruita tramite scansione 3D:

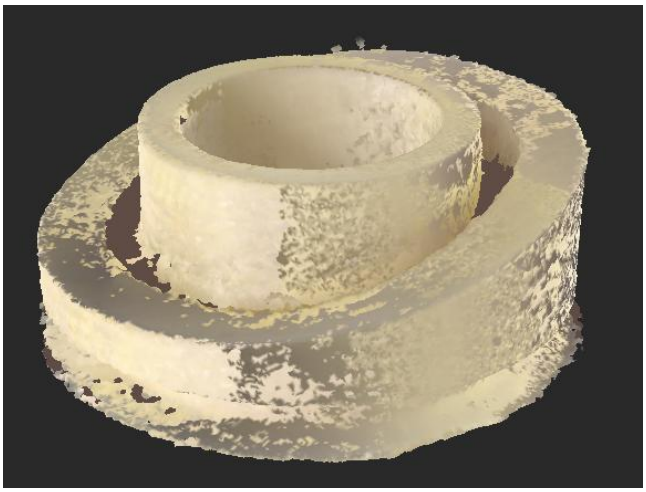


Figura 148. Original scanner 3D.

La figura mostra il risultato finale della scansione 3D tramite lo scanner Revopoint Mini. Il modello tridimensionale del pezzo è stato digitalizzato tramite l'acquisizione della nuvola di punti e la successiva elaborazione avvenuta per mezzo del software Revoscan 5.

Data la complessità della forma del componente, si rendono necessarie due scansioni parziali al fine di poter ottenere una geometria più completa possibile. Una volta acquisite le due parti, queste vengono unite tramite l'utilizzo del comando che permette l'allineamento automatico. Questa procedura che viene gestita interamente dal software ottenendo il risultato finale mostrato in figura.

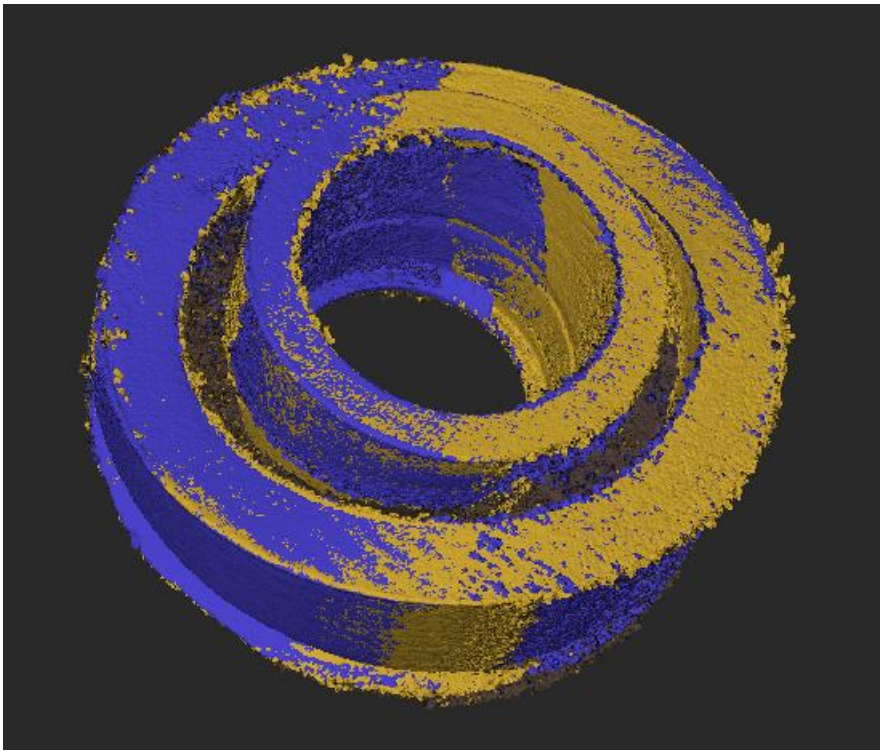


Figura 149. Unione delle scannerizzazioni parziali.

La funzione dell'allineamento automatico non è stata utilizzata per le unioni delle singole parti precedenti perché queste devono essere orientate secondo lo stesso verso. In altre parole per utilizzare questa funzione durante le scansioni non è possibile ribaltare il pezzo per registrare la nuvola di punti relativa alla parte opposta.

Inoltre, si può notare la presenza di rumore a bassa frequenza in corrispondenza dei bordi esterni.

Di seguito viene presentato il file dato alla rete neurale come input denominato secondo la nomenclatura come `camma_original.obj`.

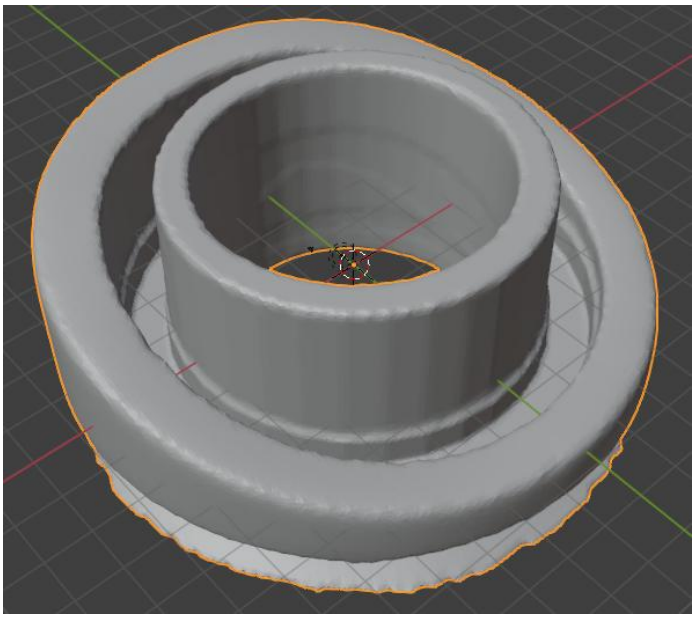


Figura 150. Remesh finale.

Caratteristiche della mesh poligonale:

- Numero vertici: 32188;
- Numero spigoli: 96564;
- Numero facce: 64376.

Al termine della fase di riparazione manuale su Blender, si effettua una decimazione della mesh poligonale, ponendo come target un valore nell'intorno di 60000 triangoli.

7.7.3 Risultati

Terminata la fase di preparazione dei modelli CAD di input, si procede con l'elaborazione da parte della rete neurale. Come per i casi studio precedenti, al fine di poter confrontare i modelli di output ottenuti al termine della fase di training, si impostano i medesimi parametri all'interno del file della rete neurale sgcn.py.

In questa sezione si riportano i modelli CAD corrispondenti all'epoca 1000 della fase di training.

1. Rottura manuale della mesh:

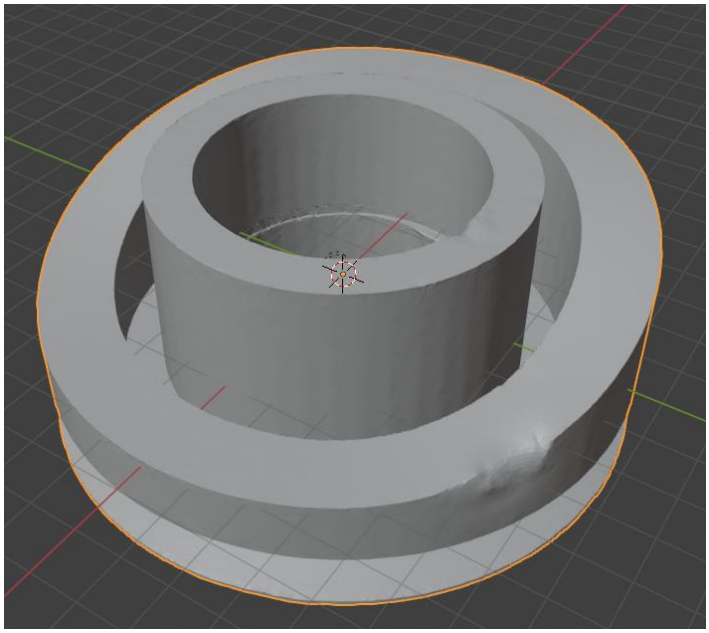


Figura 151. Output rottura manuale.

Caratteristiche della mesh poligonale:

- Numero vertici: 88307;
- Numero spigoli: 264921;
- Numero facce: 176614.

Analizzando la spaccatura creata nel file `original` dato in ingresso alla rete, si osserva che le aperture create per danneggiare il reticolo non sono di entità trascurabili. Tuttavia, sia nella corona circolare interna, sia in quella esterna, la riparazione della mesh effettuata dalla rete neurale risulta di buona qualità, presentando solo una lieve distorsione del reticolo, interpretabile come un'ammaccatura del pezzo.

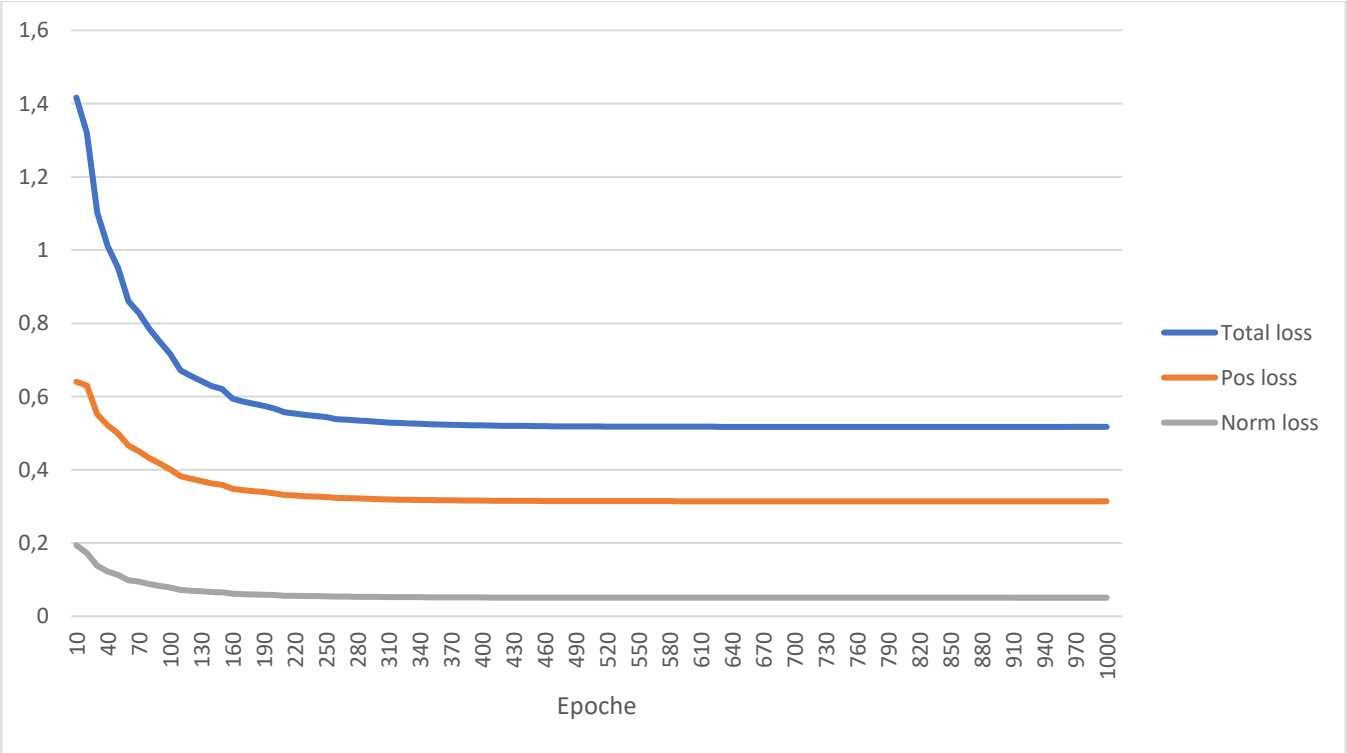


Figura 152. Andamento loss.

2. Mesh ricostruita tramite scansione 3D:

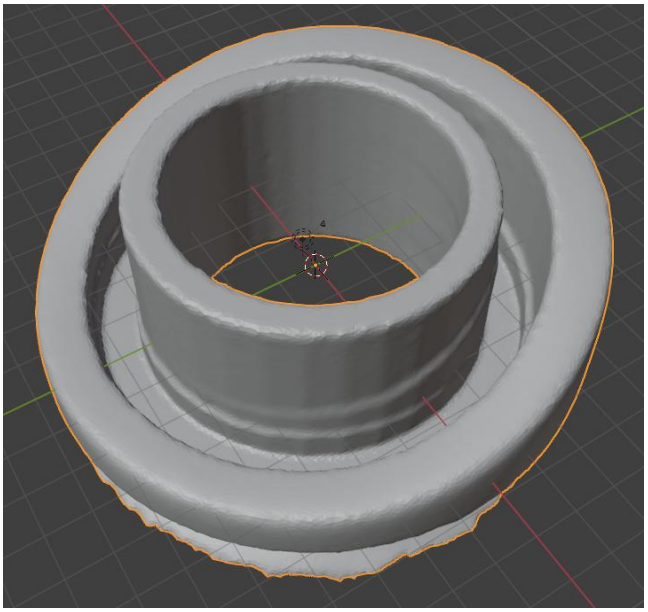


Figura 153. Output scannerizzazione.

Caratteristiche della mesh poligonale:

- Numero vertici: 85291;
- Numero spigoli: 255873;
- Numero facce: 170582.

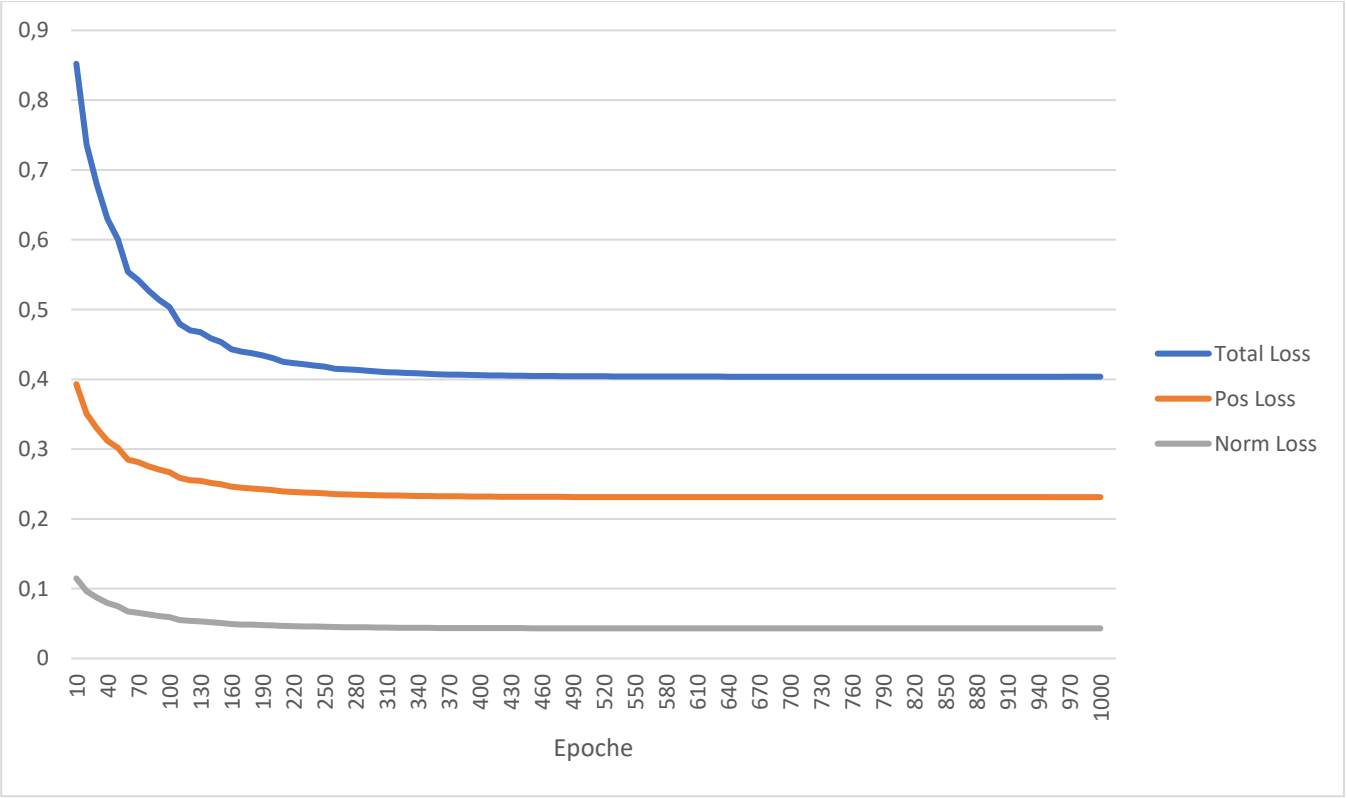


Figura 154. Andamento loss.

7.7.4 Problematiche riscontrate

Come si può osservare nella seguente figura, il risultato ottenuto dalla scansione presenta molte aree danneggiate.

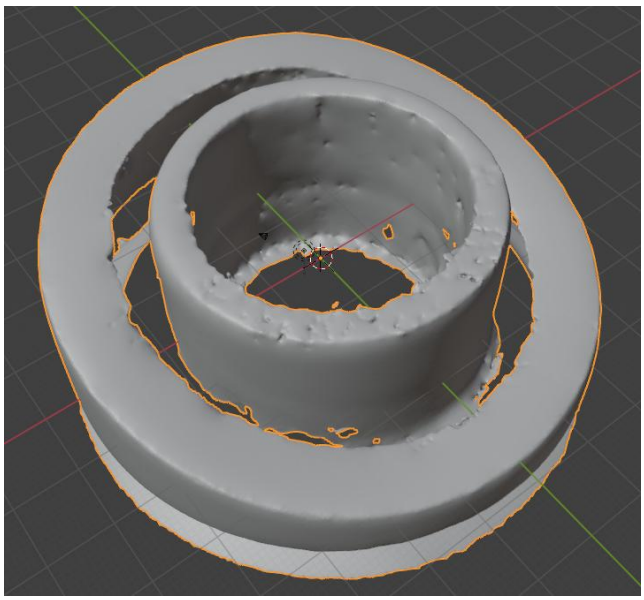


Figura 155. Output scanner importato su Blender.

Il modello acquisito dallo scanner presenta numerose parti mancanti, in particolare tutta la regione relativa allo scasso circolare presente alla periferia del foro centrale. Questo file CAD `original` così com'è non risulta utilizzabile dalla rete neurale, a causa della difficoltà di ricostruire parti completamente mancanti come affermato dagli autori di SeMIGCN, dal momento che viene prodotto un output avente distorsioni rilevanti rispetto alla forma inziale.

Per risolvere tale problema occorre una complessa fase di ricostruzione manuale della geometria attraverso Blender. Il software fornisce delle forme mesh, come per esempio il cilindro, sul quale tramite l'utilizzo dei modificatori risulta possibile effettuare operazioni booleane sulla geometria. La riparazione è stata effettuata attraverso le seguenti fasi:

1) Si è ricavata una mesh per chiudere lo scasso tra le due corone circolari;
2) Tramite l'aggiunta di due componente mesh a forma cilindrica si costruisce un cilindro cavo di dimensioni pari al cilindro interno per tutta la sua altezza e si unisce questo cilindro al pezzo;
3) Si ricava una nuova mesh tramite il corrispondente modificatore, al fine di salvare le modifiche apportate;
4) Si crea un ulteriore cilindro cavo con dimensioni compatibili a quello dello scasso tra le due corone circolari e si utilizza questo elemento per l'operazione booleana di sottrazione;
5) Si elimina il cilindro creato e si ricava una nuova mesh;
6) Infine si definisce la parte inferiore tramite l'utilizzo di un nuovo cilindro cavo, aventi dimensioni opportune per tenere traccia del foro centrale e dello spessore. Segue una fase di remshing per salvare le modifiche.

Per eseguire tutte queste operazioni occorre una fase preliminare di decimazione della mesh per non incorrere in problemi di hardware dovuti al limite di memoria, a causa dell'elevato numero di triangoli.

Il risultato finale di questa riparazione manuale viene mostrato nella figura 150, corrispondente al file `original.obj` ottenuto per scansione 3D nel paragrafo precedente.

7.7.5 Valutazioni

Il modello ottenuto si discosta lievemente dalla geometria del file CAD `original` di partenza. È importante notare che rispetto al risultato ottenuto dalla scansione 3D, il modello dato in input alla rete neurale risulta avere una geometria completa, presentando una mesh impermeabile. Infatti grazie all'elaborazione manuale per ricostruire le parti completamente mancanti, la rete neurale SeMIGCN, è riuscita ad

acquisire tutte i dettagli della superfice della `camma_scami`. Come osservato anche nel modello precedente la componente conica situata nella parte inferiore, risulta più raccordata e con i bordi terminali non ben definiti. Questa rappresentazione può essere dovuta alla presenza di rumore a bassa frequenza non completamente eliminato nella fase di preprocess di SeMIGCN.

Si riportano di seguito i risultati derivanti dal confronto tramite MeshLab.

Confronto	Distanza minima [mm]	Distanza massima [mm]	Distanza media [mm]	RMS [mm]
A-C	1,983886	1,983886	-0,035109	0,419023
A-D	3,244267	3,244267	-0,334877	1,28748
B-D	1,820631	1,820631	0,219727	0,554063
A-B	2,224653	2,224653	-0,569898	1,212845

1) Caso AC: in figura si nota una maggiore differenza in corrispondenza delle rotture della mesh create, in particolare sulla corona circolare esterna. Le discrepanze sono comunque contenute in gran parte nell'ordine del decimo di millimetro.

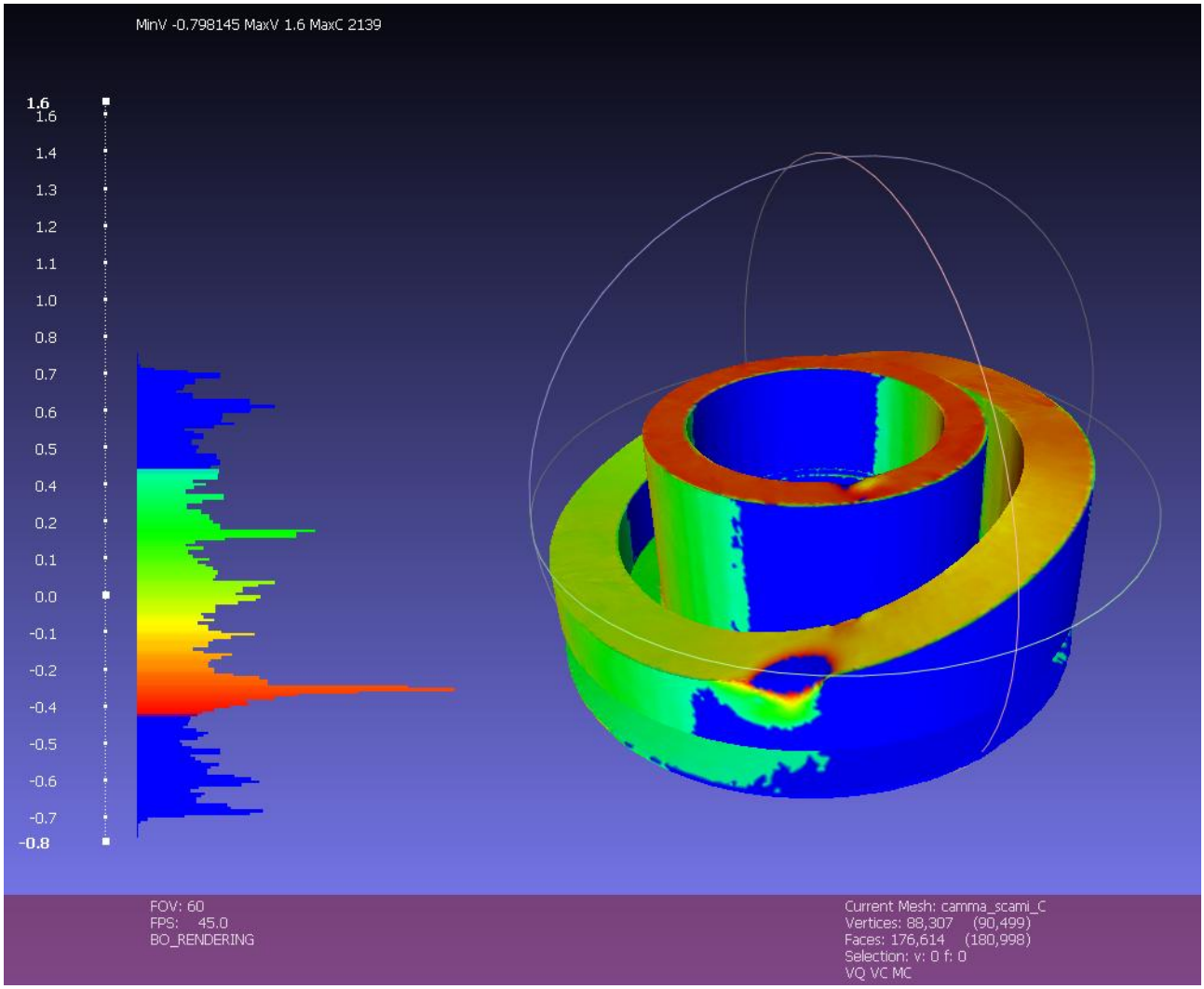


Figura 156. Confronto AC.

2) Caso AD: il risultato ottenuto risulta di buona qualità, tuttavia i valori di scostamento così alti sono dovuti alla preelaborazione manuale per ricostruire la mesh gravemente danneggiata ottenuta dalla fase di scansione. Da ciò si deduce che le modifiche apportate su Blender compromettono la precisione del risultato finale.

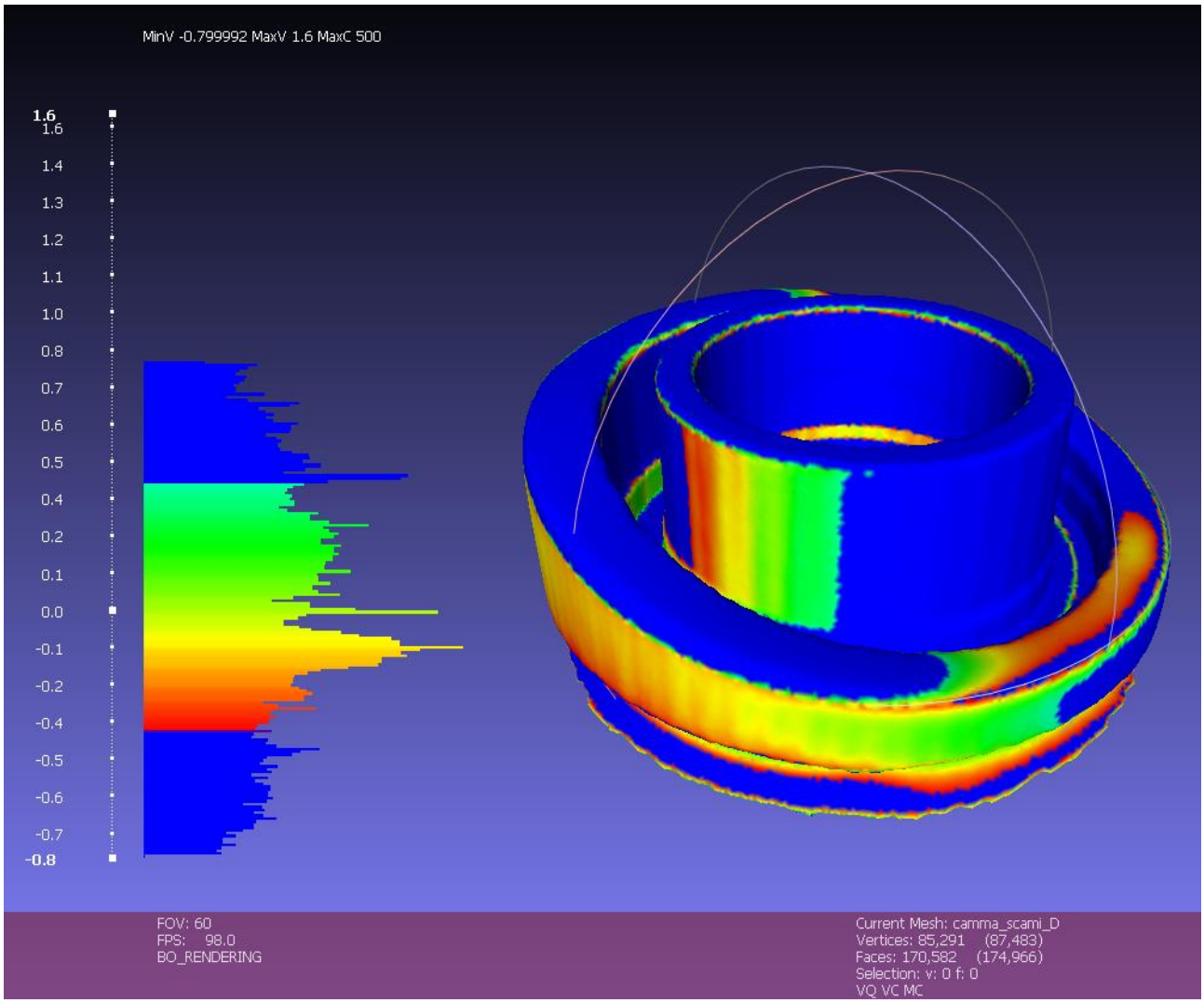


Figura 157. Confronto AD.

3) Caso BD: la rete neurale effettua una ricostruzione del componente con una precisione inferiore rispetto ai casi precedenti.

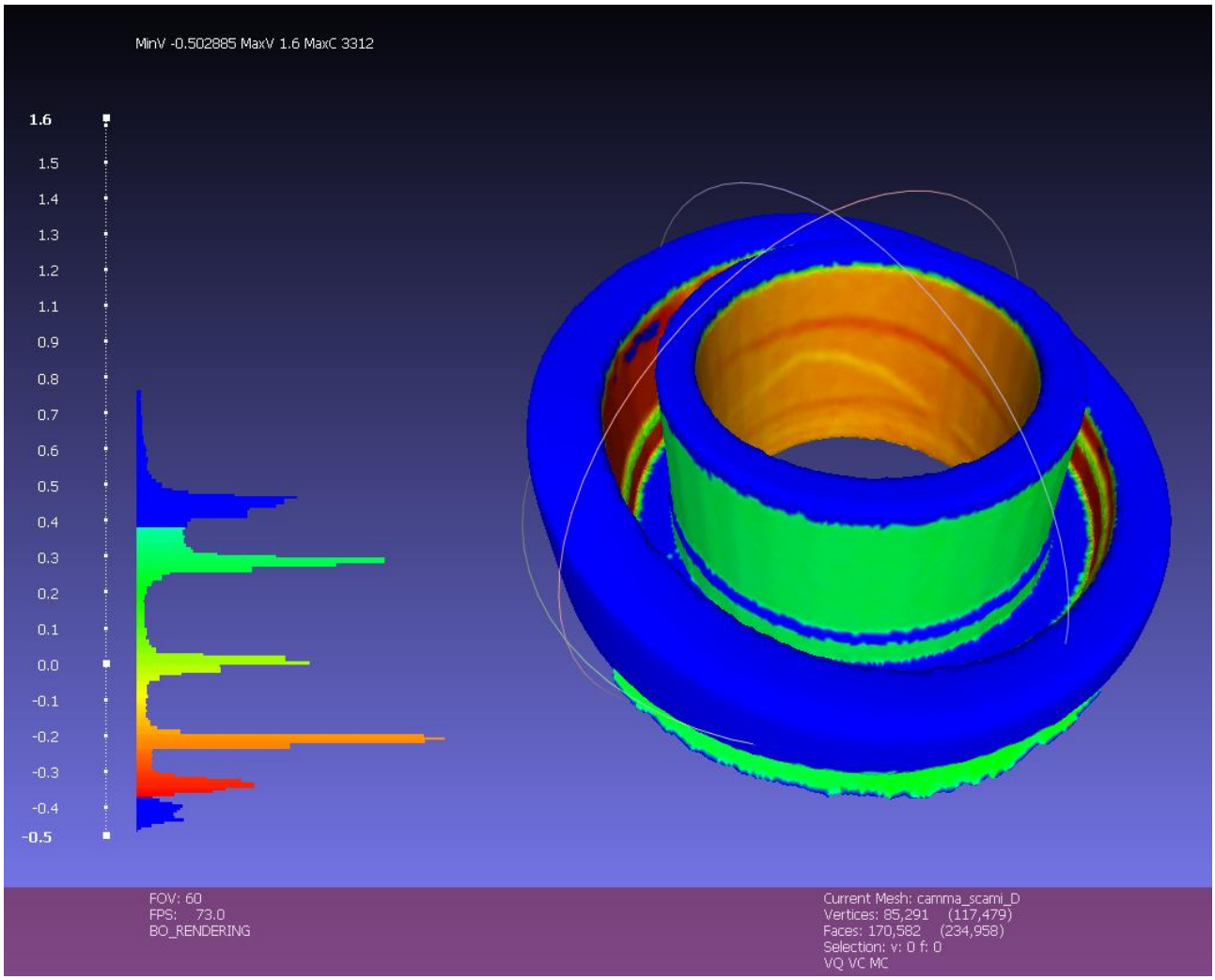


Figura 158. Confronto BD.

4) Caso AB: nella figura si evidenzia la predita della precisone di acquisizione in fase di scansione, a causa dalla preelaborazione manuale di ricostruzione effettuata su Blender.

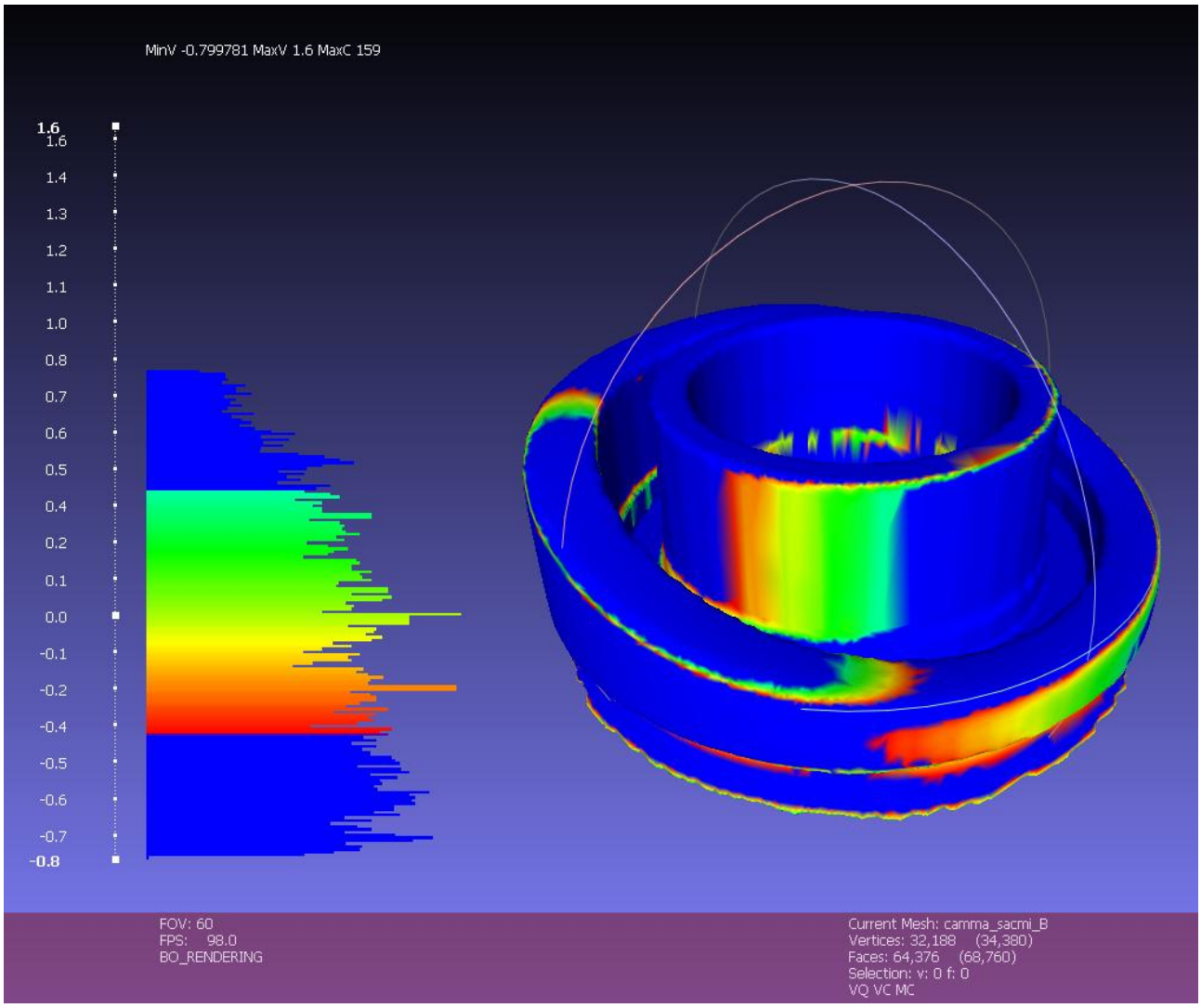


Figura 159. Confronto AB.

8. Conclusioni

In questo elaborato si è esaminato la potenzialità dell'applicazione dell'intelligenza artificiale al reverse engineering. Al fine di raggiungere l'obiettivo preposto, sono stati progettati componenti meccanici aventi caratteristiche geometriche differenti e si è analizzato il modo con cui la rete neurale gestisce queste forme, evidenziando punti di e limitazioni.

Per quanto riguarda la parte di modellazione CAD, i software a disposizione quali Creo Parametric 11, Blender 4.2, MeshLab e Revoscan 5 hanno fornito un buon supporto, permettendo di costruire con rapidità i singoli componenti, modellare agevolmente la mesh poligonale al fine di ottenere il campionamento desiderato e infine confrontarli tra loro.

Le uniche limitazioni rilevate in questa parte dell'elaborazione sono relative alla componente hardware. In particolare per quanto riguarda lo scanner Revopoint Mini, emerge la difficolta nel registrare la nuvola di punti quando si hanno fori troppo profondi (profondità > 50 mm) e di piccolo diametro. Inoltre, risulta molto difficoltosa la scansione di dettagli troppo interni come nel caso degli scassi profondi. Questa problematica si è presentata nell'esempio della `camma`.

I difetti presenti sulla finitura superficiale delle stampe tridimensionali dei pezzi hanno inevitabilmente influito sul processo, determinando nel confronto finale disallineamenti minimi che tuttavia risultano conformi alla precisione reale ottenuta con lo scanner 3D e pari ad un range dell'ordine del decimo di millimetro. Questo è un valore sensibilmente più elevato di quello fornito dal costruttore (0.02 mm), ma per via della procedura pratica di scansione questa precisione si va a perdere ad ogni minimo movimento del pezzo durante l'operazione. Questa considerazione sul confronto delle mesh mostra comunque la validità dei risultati di output ottenuti.

Un'ulteriore limitazione dello scanner 3D è rappresentata dalle dimensioni massime dei pezzi, che per ottenere una buona qualità devono stare all'interno di un cubo di 100 mm di lato.

Per quanto riguarda la rete neurale SeMIGCN, questa presenta dei vantaggi legati al suo utilizzo, come il non dover costruire un dataset di dati numeroso, in quanto la rete riceve in input una sola mesh incompleta. Inoltre come evidenziato dai modelli di output, la rete neurale risulta in grado di riparare la mesh danneggiata di piccole porzioni mancanti con una buona precisione e di migliorare la qualità superficiale di input, quando questa presenta delle lievi deformazioni, come si può osservare nel caso della `flangia_forata`.

Tuttavia, come evidenziato nell'articolo fornito dagli autori e anche durante la fase di testing, la rete neurale presenta delle limitazioni, quali l'impossibilità di ricostruire la mesh poligonale di parti completamente mancanti e l'impossibilità di modificare il campionamento del reticolo adattandolo alla configurazione geometrica del componente. Quest'ultima questione si osserva nel più elevato numero di vertici presenti alla fine del training rispetto al modello `original` di partenza, infatti si ottengono dei pezzi con mesh molto fini, che vanno a gravare sulla richiesta di memoria delle risorse computazionali.

In relazione ai grafici ottenuti al termine della fase di testing, sono stati graficati i tre valori di loss forniti dall'algoritmo e dal loro andamento nessuno dei casi affrontati presenta overfitting, in quanto le tre funzioni seguono sempre lo stesso andamento decrescente. Alcuni modelli presentano un andamento oscillatorio nel grafico dovuto al valore settato di learning rate.

In aggiunta importando alcuni modelli alla fine del training, si riscontra una variazione della scala dimensionale rispetto al modello del ground truth, mantenendo però inalterate le proporzioni. Questa problematica risulta di facile risoluzione tramite il comando scala su Blender.

Dato il continuo sviluppo delle reti neurali, queste limitazioni evidenziate costituiscono un buon punto di partenza per ottenere modelli sempre più performanti, in maniera da semplificare e ridurre i tempi di lavoro. È inoltre importante considerare che usufruendo di uno scanner 3D più performante e con un'attenta raffinazione del pezzo stampato, sarebbe possibile ottenere risultati migliori.

9. Bibliografia

- https://tarini.di.unimi.it/teaching/cg2020/Lez05A.Mesh_poligonali.pdf
- Mario Botsh, Leif Kobbelt, Mark Pauly, Pierre Alliez, Bruno Lévy. Polygon Mesh Processing. 2010.
- https://www.stampa3d-forum.it/articoli/guide/regole-modellazione-3d/
- https://help.autodesk.com/view/ACD/2023/ITA/?guid=GUID-A6232957-5039-4AB7-8B1D-8FD0AD98F77B
- https://amslaurea.unibo.it/id/eprint/3763/1/TESI.pdf
- https://www.sciencedirect.com/topics/computer-science/polygonal-mesh
- https://tarini.di.unimi.it/teaching/cg2020/Lez05A.Mesh_poligonali.pdf
- https://3dstudio.co/it/polygon-mesh/
- https://www.danthree.studio/it/blog-cgi/cos'%C3%A8-un-modello-mesh-3d-esempi-di-definizione
- https://alessiomorselli.github.io/DeepLearning/web/pages/teoria/basi.html
- **Scolari, Stefano** (2022) *Geometric Deep Learning per il denoising di mesh 3D.* Università di Bologna.
- https://www.it-impresa.it/blog/cosa-sono-le-reti-neurali/
- https://www.ibm.com/it-it/topics/neural-networks
- https://www.di.univr.it/documenti/OccorrenzaIns/matdid/matdid833568.pdf
- https://blog.alleantia.com/it/reti-neurali-tipologie-e-applicazioni
- https://medium.com/@nerdjock/convolutional-neural-network-lesson-9-activation-functions-in-cnns-57def9c6e759
- https://www.di.univr.it/documenti/OccorrenzaIns/matdid/matdid833568.pdf
- https://github.com/astaka-pe/SeMIGCN
- S. Hattori, T. Yatagawa, Y. Ohtake and H. Suzuki, "Learning Self-Prior for Mesh Inpainting Using Self-Supervised Graph Convolutional Networks," in *IEEE Transactions on Visualization and Computer Graphics*, vol. 31, no. 2, pp. 1448-1464, Feb. 2025
- Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. 5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings, pages 1‑14, 2017.
- https://www.revopoint3d.com/pages/industry-3d-scanner-mini2
- https://eu.store.bambulab.com/it-it/collections/pla/products/pla-basic-filament
- https://eu.store.bambulab.com/it-it/collections/3d-printer/products/x1-carbon
- https://3dmano.it/stampanti-3d/stampante-3d-bambu-lab-x1-carbon/890.html
- https://solidworld.it/blog/cose-il-reverse-engineering-e-quali-sono-i-suoi-vantaggi/
- https://www.meshlab.net/