ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Seconda Facoltà di Ingegneria
Corso di Laurea Magistrale in Ingegneria Informatica
Tesi di Laurea in: Fondamenti di Computer Graphics

UN METODO MULTILIVELLO PER LA SEMPLIFICAZIONE DI MESH POLIGONALI DA SCANNER 3D

Tesi di Laurea di:
LUCA GUERRA

Relatore:
Prof. SERENA MORIGI
Co-relatore:
Ing. MARCO RUCCI

SESSIONE I
ANNO ACCADEMICO 2011–2012

PAROLE CHIAVE

Mesh Poligonale

Semplificazione

Edge-Collapse

Scanner 3D

Alla mia famiglia e alla Lucia

Indice

Introduzione

Al giorno d'oggi il mondo della computer grafica utilizza come modelli geometrici in larga parte delle strutture chiamate mesh poligonali. Queste strutture vengono utilizzate per la modellazione digitale di modelli tridimensionali complessi, hanno molteplici utilizzi e per questo vengono sfruttate in vari ambiti. Il più conosciuto tra questi è sicuramente l'animazione digitale, basti pensare al successo di aziende come la Pixar, nata alla fine degli anni 70,acquistata da Steve Jobs nel 1986 e venduta nel 2006 alla Disney per 7,6 miliardi di dollari.
Possiamo dire che, dopo l'uscita nelle sale di Toy Story, nel 1995, il primo film d'animazione creato interamente con la computer grafica, l'animazione digitale ha di fatto soppiantato l'animazione tradizionale portandola ad un livello nettamente superiore. Ma questo è solo il più conosciuto degli ambiti in cui le mesh vengono utilizzate, sono molto usate naturalmente nei videogiochi, in architettura, in ambito industriale, medico, e nelle rappresentazioni del territorio. Possiamo certamente dire che al giorno d'oggi l'utilizzo della grafica digitale è pervasivo in molti ambiti.
Le mesh poligonali non sono nient'altro che delle approssimazioni di superfici arbitrarie formate da un insieme di *faccette* che possono avere varie forme. Questa rappresentazione a faccette piane permette comunque di approssimare anche delle superfici con zone ad alta curvatura, utilizzando un elevato numero di facce laddove richiesto.
Le mesh possono essere ottenute in vari modi e con varie tecniche:

- Tramite **punti**: è il metodo più comune, si utilizzano algoritmi di triangolazione e tetraedralizzazione (scanner 3D, fotogrammetria, dati territoriali...).
- Tramite **superfici**: algoritmi di tassellazione (per il rendering di superfici curve).

- Tramite **volumi di dati**: poligonalizzazione (dati volumetrici,isosuperfici).

La richiesta di modelli sempre più complessi ha aumentato nel corso degli anni il carico computazionale richiesto per gestire il rendering di questo tipo di strutture. Questo problema si può risolvere fondamentalmente in due modi, o si utilizzano calcolatori sempre più potenti a livello hardware (con l'introduzione di acceleratori grafici sempre più potenti), o ci si affida a tecniche di semplificazione, ovvero rappresentazione di un modello geometrico con un numero ridotto di faccette. Questo lavoro di tesi si occuperà del secondo approccio.
Le mesh che più necessitano di semplificazione sono quelle provenienti dall'acquisizione tramite scanner 3D. Questo tipo di acquisizione permette di disporre in un tempo relativamente breve di modelli tridimensionali per la cui realizzazione, diciamo così, "a mano" sarebbe stato necessario un tempo considerevolmente più lungo. Tutto ciò al prezzo di una fase obbligata di ottimizzazione relativamente inevitabile dopo l'acquisizione in quanto questo tipo di modelli sono quasi sempre affetti da *rumore* e *buchi* sulla superficie nelle zone in cui il laser dello scanner non è riuscito ad acquisire dei punti.
L'ottimizzazione delle mesh è fondamentale, sia per eliminarne i difetti iniziali di acquisizione sia per renderle più trattabili a seconda dell'ambito in cui devono essere utilizzate. La semplificazione delle mesh ha l'obiettivo è diminuire il "peso" di queste strutture in termini computazionali, per facilitarne l'utilizzo, la manipolazione e velocizzare il rendering. Il tutto senza andare a danneggiare la mesh in termini di qualità.
Un esempio su tutti, se una mesh è in primo piano nella scena, la qualità dovrà essere alta e la semplificazione quasi nulla (fatta eccezione per gli interventi di smoothing) ma se una mesh deve apparire lontana dall'osservatore, sullo sfondo, allora sono giustificati interventi di semplificazione in quanto lo spazio occupato dalla figura nella scena sarà molto minore e di conseguenza non ci sarà bisogno di avere un'immagine ben definita e di qualità. Tutto questo faciliterà il lavoro di rendering abbassando, ad esempio, i tempi di resa.
L'algoritmo di semplificazione proposto in questa tesi agisce in modo iterativo su due livelli, in una prima fase alla mesh viene applicato un operatore di smoothing e alla mesh così elaborata viene poi applicata la decimazione. L'alternanza di questi due operatori permette di avere mesh semplificate che mantengono però una qualità maggiore rispetto al risultato che si avrebbe invece applicando solo la decimazione. A partire da una mesh ad alta riso-

luzione, ad ogni iterazione viene quindi creata una mesh con una risoluzione inferiore fino all'ultima iterazione che produce la mesh coarse (a bassa risoluzione) desiderata. Si viene quindi a creare una struttura di semplificazione multilivello, utile in molte applicazioni di computer graphics.
Nel primo capitolo verrà fornita una descrizione più accurata delle mesh poligonali, necessaria per comprendere il resto dell'elaborato; oltre a questo verranno spiegate un po' più nel dettaglio le operazioni di ottimizzazione che possono essere effettuate su queste strutture, come lo *smoothing*, il *fairing*, l'*hole filling*. Inoltre verranno fornite informazioni sulle più comuni tecniche di semplificazione e le loro applicazioni.
Nel secondo capitolo verrà descritta l'esperienza di acquisizione delle mesh da scanner 3D e il procedimento svolto poi su di esse per poterle utilizzare.
Nel terzo capitolo verrà spiegato il funzionamento dell'algoritmo di semplificazione proposto.
Nel quarto capitolo verranno presentati i test svolti sull'algoritmo comparati con quelli ottenuti utilizzando uno dei software più utilizzati per effettuare questo tipo di operazioni.
Infine un capitolo conclusivo in cui saranno raccolte le conclusioni personali dell'autore sul lavoro svolto per realizzare questa tesi e i risultati ottenuti.

Capitolo 1

Semplificazione Mesh Poligonali

Il metodo più diffuso per rappresentare superfici tridimensionali in computer grafica è quello di scomporle in superfici elementari poligonali e piane. La rappresentazione di una superficie viene quindi realizzata in forma esplicita per mezzo di un insieme di vertici (punti nello spazio in cui si trova la superficie) uniti appunto da poligoni elementari. Questi ultimi devono essere facilmente trattabili a livello matematico così da accelerare tutte le operazioni che possono dover essere svolte su di essi, ad esempio lo spostamento. Per questo motivo di solito si prediligono i triangoli che forniscono anche una discreta flessibilità nella costruzione di oggetti 3D complessi. È immediatamente comprensibile come, data la natura piana di questa primitiva, la modellazione accurata di superfici tridimensionali complesse richieda l'uso di un elevato numero di facce. In ambito di computer graphics l'utilizzo sempre crescente di modelli 3D ad alta risoluzione per aumentare il realismo di scene tridimensionali ha come contropartita la difficoltà di visualizzare tali scene interattivamente. Gli strumenti a disposizione sono due: l'incremento delle prestazioni delle piattaforme hardware disponibili e l'utilizzo di tecniche di semplificazione, come quella sviluppata in questa tesi.

1.1 Mesh Poligonali

Una mesh poligonale è essenzialmente una collezione di vertici, spigoli (edge) e facce che definiscono la forma di un oggetto tridimensionale nella

computer grafica 3D e nella modellazione solida.
Un vertice è essenzialmente la rappresentazione di una posizione nello spazio. Un edge è l'entità che connette due vertici. Una faccia è un insieme di punti nello spazio racchiuso tra edge e vertici. L'insieme di queste facce può determinare poligoni o strutture molto più complesse. Le facce consistono solitamente di triangoli per una migliore trattabilità e flessibilità, ma anche di quadrilateri o altri semplici poligoni convessi. Possono essere composti anche da poligoni concavi più generici, o poligoni con buchi.
Le mesh sono quindi primitive grafiche che consentono di rappresentare un poliedro qualsiasi o di approssimare superfici curve. Lo studio di mesh poligonali è molto importante nella computer grafica. Le rappresentazioni con mesh poligonali sono utilizzate per diverse applicazioni e con diverse finalità.

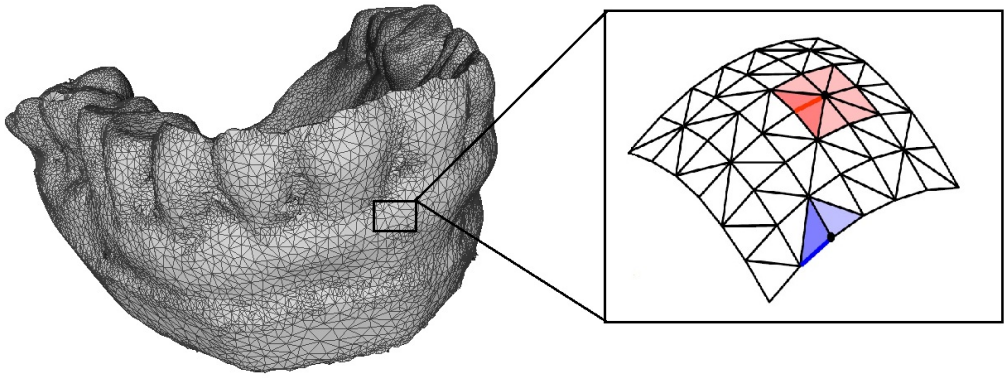

Figura 1.1: Struttura di una mesh, in rosso i vertici e gli edge interni, in blu le zone di boundary

In una mesh, per poter essere definita tale, un lato è condiviso da al massimo due facce ed è compreso tra due vertici. Mentre un vertice è condiviso da almeno due edge.
Ogni mesh ha un particolare genere topologico. In particolare, il genere di una superficie viene definito come il numero più grande di curve semplici chiuse che possono essere disegnate sulla superficie senza separarla in due parti non connesse, questo numero è assimilabile al numero di buchi presenti

sulla superficie della mesh, per cui un cubo ed una sfera hanno genere zero, una ciambella genere uno ed un otto genere due (vedi figura 1.2).

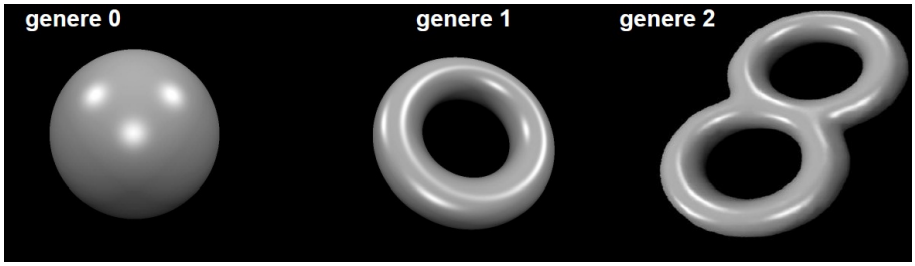

Figura 1.2: Esempi di mesh di diverso genere

Una mesh viene definita *chiusa*, se non vi sono buchi sulla sua superficie, altrimenti è consuetudine dire che la mesh è *aperta*. In questi casi tutti gli elementi (edge,vertici..) che si vengono a trovare sul bordo vengono definiti boundary.
La topologia locale di una faccia, lato o vertice del mesh descrive invece come tale elemento è connesso con il resto della mesh, ossia quali vertici, lati o facce sono parte di esso o incidenti in esso. La mesh fa parte della tipologia cosidetta 2-manifold se localmente ogni punto sulla superficie della mesh è topologicamente equivalente ad un disco unitario. Se la mesh rispetta queste caratteristiche è definita *consistente*.
Nel caso di mesh triangolari questo disco è un anello (ring) di triagoli che circonda il punto/vertice, se quest'ultimo è boundary (parliamo quindi di mesh aperte) allora basterà che abbia solo un half-ring intorno (vedi figura 1.1).
In una mesh triangolare 2-manifold ogni lato ha esattamente due facce incidenti. Un 2-manifold con boundary consente la presenza di alcuni lati con un solo triangolo incidente. Questo tipo di mesh è relativamente semplice da trattare.
Esistono naturalmente anche mesh non manifold, cioè mesh la cui struttura presenta caratteristiche non corrette, come giunzioni a T, edge condivisi da più di due facce o vertici condivisi da più superfici (vedi figura 1.3).

Oltre alla topologia, abbiamo anche altre informazioni associate alle mesh. La geometria, tramite le coordinate dei vertici, e l'orientamento, tramite l'ordinamento dei vertici di ogni faccia.

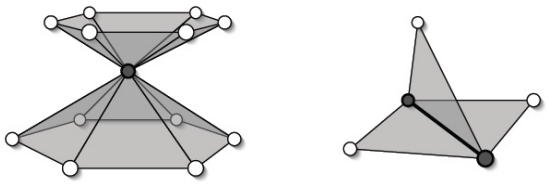

Figura 1.3: Esempio di un vertice e di un edge non manifold

Le mesh possono essere rappresentate in diversi modi, utilizzando diverse strutture dati atte a memorizzare vertici, edge e facce. La più famosa di queste strutture è il **winged-edge**, in cui ad ogni edge sono associate informazioni sui due vertici tra i quali è racchiuso, le due facce che lo condividono e i quattro edge collegati ai suoi vertici estremi; che andrammo poi a formare appunto le due facce sopracitate.
È una struttura dati lievemente ridondante ma è ottimale per velocizzare la navigazione della mesh.

In figura 1.4, vediamo il sistema di riferimenti in questione relativo ad un cubo. Come si può notare ogni faccia ha associati i riferimenti relativi agli edge che la delimitano, ogni vertice ha le sue coordinate e i suoi edge incidenti. Gli edge infine hanno informazioni relative ai vertici tra i quali sono racchiusi, le due facce tra cui si trovano e i quattro edge che vanno a delimitare appunto queste due facce.
Vi sono altre strutture dati per la memorizzazione dei dati inerenti alle mesh:

- **Half-Winged-Edge**: variazione orientata del winged-edge, ogni half-edge ha associato un orientamento e le informazioni associate ad esso sono relative solo al vertice incidente, la faccia incidente e all'half-edge precedente, successivo e opposto (vedi figura 1.5).

- **Face-vertex**: una semplice lista di vertici, insieme ad una lista di facce associate alle informazioni sui vertici che le racchiudono (vedi figura 1.8).

- **Quad-edge**: vengono salvati gli edge, half-edge e i vertici, senza nessun riferimento alle facce.

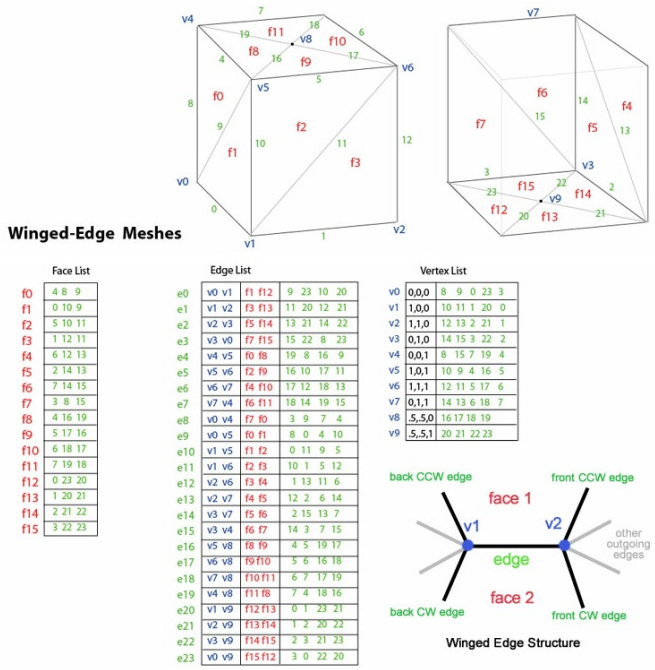

Figura 1.4: Sistema di riferimenti in una struttura Winged-Edge per una mesh cubica

- **Vertex-vertex**: si hanno riferimenti solo ai vertici, le info su edge e facce sono implicite (vedi figura 1.7).

1.2 Operazioni su Mesh

Le mesh vengono utilizzate in larghissima parte per rendere delle superfici curve più o meno accurate. Più queste sono complesse più è alto il numero di facce richiesto per approssimare con cura queste superfici.
Per questo si sono resi necessari dei metodi che permettano di manipolare e rielaborare queste mesh per diminuire il carico computazionale mantenendo al tempo stesso inalterata la topologia.
Un caso in cui c'è bisogno di molta ottimizzazione è proprio il caso preso in esame da questa tesi, ovvero quando si ha a che fare con dati tridimensionali acquisiti da scanner 3D. Questi dati sono infatti quasi sempre *oversampled*,

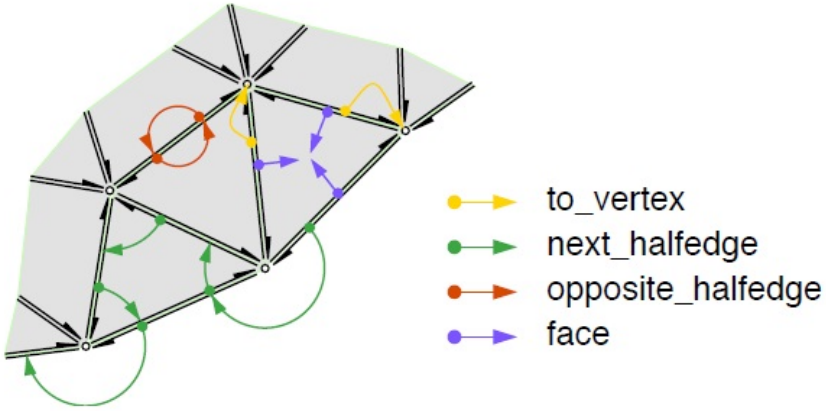

Figura 1.5: Struttura Half-Winged-Edge

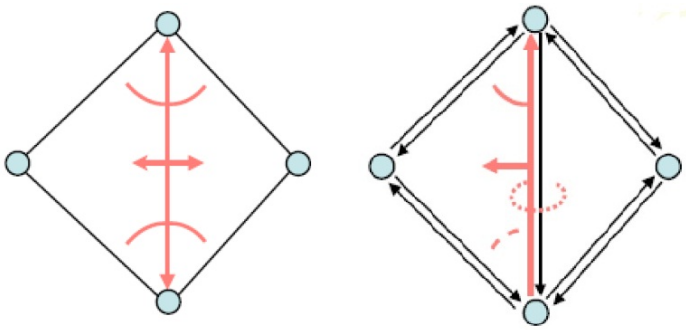

Figura 1.6: Strutture Winged-Edge e Half-Winged-Edge a confronto

cioè sovracampionati e soffrono anche la presenza di rumore introdotto dalle varie scansioni. Inoltre presentano quasi sempre dei buchi.
Le tecniche più usate per andare ad agire su questi problemi sono tecniche appunto di *denoising* come il fairing o smoothing, che eliminano il rumore ed agiscono su eventuali irregolarità nelle superfici rendendole più *smooth*, cioè più lisce. Oppure tecniche di *hole filling*, per riempire eventuali buchi. Queste ultime però ottimizzano la mesh più che altro da un punto di vista puramente visivo, se si vuole andare ad agire in maniera un po' più aggressiva sulla mesh e renderla più trattabile per eventuali manipolazioni, velocizzare il rendering o anche diminuire l'occupazione di spazio, bisogna affidarsi a tecniche di semplificazione.

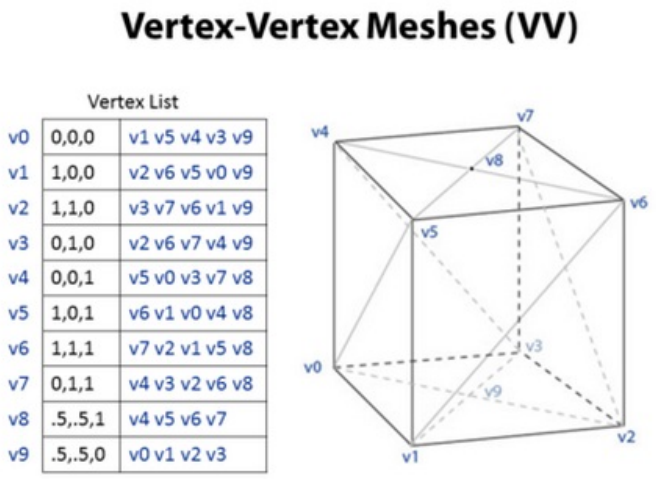

Figura 1.7: Sistema di riferimenti in una struttura Vertex-Vertex per una mesh cubica

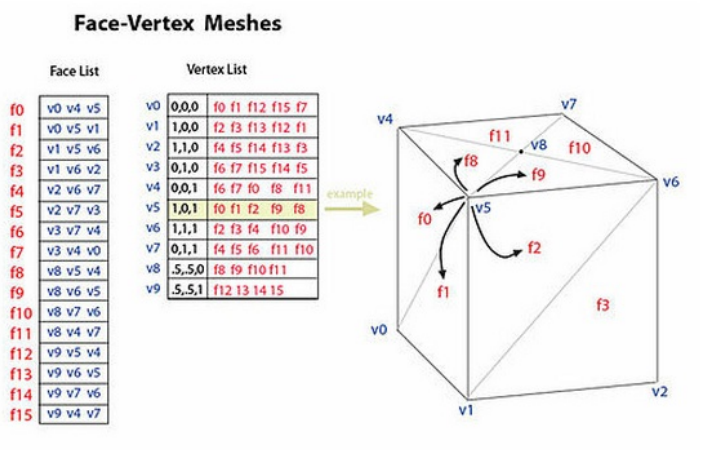

Figura 1.8: Sistema di riferimenti in una struttura Face-Vertex per una mesh cubica

La semplificazione mesh può essere sfruttata in diversi contesti applicativi, basti pensare alle approssimazioni LOD (Level of Detail) che utilizzano mesh a diversi livelli di semplificazione a seconda del contributo di queste

nella scena da rendere.

1.3 Smoothing

In computer grafica per fare operazioni di smoothing di superfici si utilizzano diverse tecniche, alcune di esse sono basate sull'uso di operatori differenziali, come l'operatore di Laplace-Beltrami, opportunamente discretizzati su mesh.
L'effetto di questi modelli è di minimizzare specifici funzionali di energia definiti sulla superficie (total curvature, thin-plate, membrane...)e, in parole povere, *lisciarne* la superficie, come una specie di filtro passa-basso. Gli operatori differenziali sono però continui e bisogna quindi discretizzarli per poterli applicare a mesh arbitrarie. Definendo ogni vertice $v_i = (x_i, y_i, z_i)$ della mesh si definiscono in [13] le coordinate laplaciane δ_i del vertice stesso come la differenza tra le sue coordinate assolute e il centro di massa dei suoi immediati vicini e pesando questo valore utilizzando la valenza di v_i, d_i:

$$\delta_i = v_i - \frac{1}{d_i} \sum_{j \in N(i)} v_j$$

dove $N(i)$ sono i vicini di v_i collegati da un edge. La trasformazione del vettore delle coordinate Cartesiane assolute nel vettore delle coordinate δ può essere rappresentata in forma matriciale. Definendo A come la matrice di connettività della mesh:

$$A_{ij} = \begin{cases} 1 & (i,j) \in E \\ 0 & \text{altrimenti} \end{cases}$$

Definendo D come la matrice diagonale per la quale $D_{ii} = d_i$, la matrice di trasformazione dalle coordinate assolute a quelle relative diventa:

$$L = I - D^{-1}A$$

Spesso però è più conveniente usare la versione simmetrica di L ovvero $L_s = DL = D - A$.

$$(L_s)_{ij} = \begin{cases} d_i & (i = j) \\ -1 & (i,j) \in E \\ 0 & \text{altrimenti} \end{cases}$$

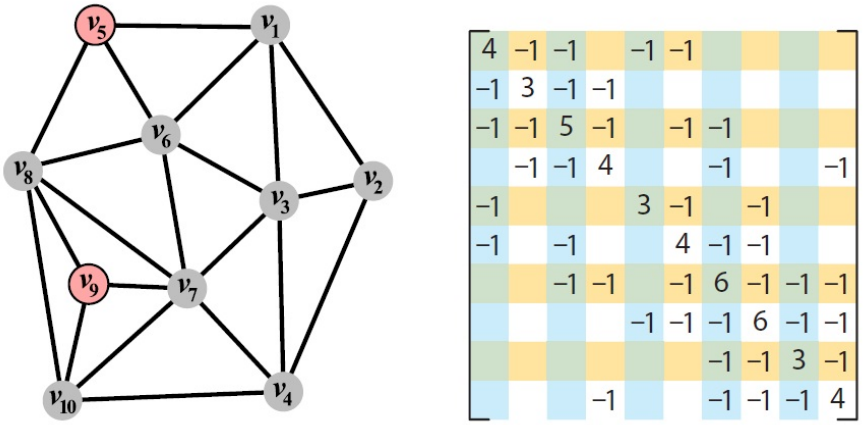

Figura 1.9: Esempio di mesh triangolare con la matrice L_s ad essa associata

L_s è chiamato appunto Laplaciano della mesh.
In figura 1.9 si illustra una mesh sulla sinistra e associata matrice di connettività L_s. Dal punto di vista della geometria differenziale le coordinate δ possono essere viste come la discretizzazione dell'operatore continuo di Laplace-Beltrami, se assumiamo che la mesh M sia l'approssimazione lineare a tratti di una superficie liscia. Per questo motivo si può scrivere il vettore delle coordinate δ del vertice v_i definito in (x) come:

$$\delta_i = \frac{1}{d_i} \sum_{j \in N(i)} (v_i - v_j)$$

Questo è il tipo di discretizzazione più semplice e viene chiamata **Umbrella**. Esistono diversi tipi di discretizzazioni di questo tipo, con diverse proprietà (in termini di accuratezza, performance, tipo di mesh che gestiscono...). Un'altra discretizzazione molto comune del laplaciano si ottiene utilizzando le cotangenti relative ai due angoli opposti all'edge i, j (vedi figura 1.10):

$$\delta_i = \frac{1}{|\Omega_i|} \sum_{j \in N(i)} \frac{1}{2} (\cot \alpha_{ij} - \cot \beta_{ij})(v_i - v_j)$$

Il laplaciano pesato è un operatore locale e viene applicato a tutti i vertici in base alla situazione del loro intorno. Dalla formula dell'operatore precedente è facile notare come il laplaciano di un vertice v_i non sia altro che

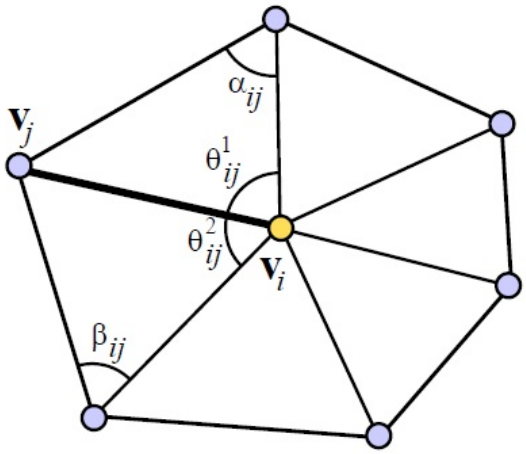

Figura 1.10: Gli angoli usati nel calcolo del Laplaciano

il vettore che va da v_i al baricentro dei suoi vicini v_j. Spostando il vertice v_i lungo tale vettore non si fa altro che *appiattire* localmente la mesh.
Un altro tipo di discretizzazione è quella che prende il nome del ricercatore che la propose, **Fujiwara**, che tiene conto delle lunghezze dei lati coinvolti:

$$\delta_i = \frac{1}{\sum_{j \in N(i)} l_j} \sum_{j \in N(i)} \frac{v_j - v_i}{l_j}$$

dove l_j rappresenta la lunghezza del lato da v_i a v_j.

In figura 1.11 si illustra l'applicazione di una tecnica di smoothing ad una mesh acquisita con scanner 3d, che presenta perturbazioni sui vertici dovute alla fase di acquisizione. Il risultato dello smoothing è mostrato in figura 1.12.

1.4 Hole Filling

Si parla di operazioni di Hole Filling quando si vuole andare a riempire un eventuale buco nella mesh, dovuto alla mancanza di dati non acquisiti durante la scannerizzazione. Una delle tecniche più usate in questi casi è la ritriangolazione del buco.

Figura 1.11: Mesh acquisita da scanner con problemi di noising

Figura 1.12: Mesh di figura 1.11 dopo la fase di fairing

Non è una tecnica particolarmente complicata e ci sono vari modi per metterla in atto, a titolo di esempio è stato preso in esame l'approccio utilizzato da Meshlab (`http://meshlab.sourceforge.net/`) uno dei tool freeware più utilizzati per la modellazione tramite mesh poligonali.
In pratica una volta selezionato il buco su cui andare ad agire, viene scelta volta per volta una coppia di border edge, ovvero di quegli edge che fungono al momento da bordo del buco. La scelta può essere fatta con vari criteri, in

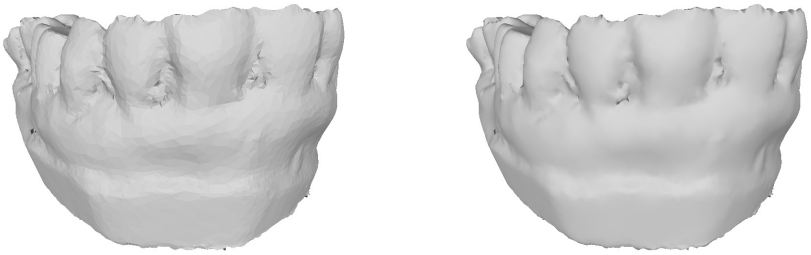

Figura 1.13: Altro esempio di mesh prima e dopo il denoising

questo caso si tratta di un semplice algoritmo greedy che sceglie ogni volta la coppia migliore in base alla scelta delle impostazioni fatta dall'utente.

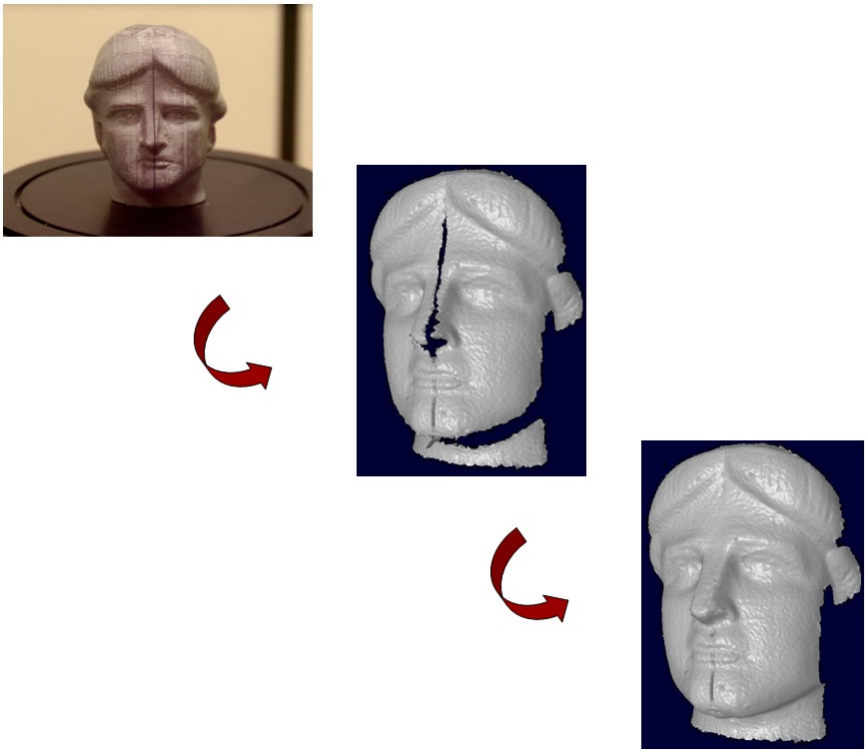

Figura 1.14: Esempio di Hole Filling su mesh acquisita

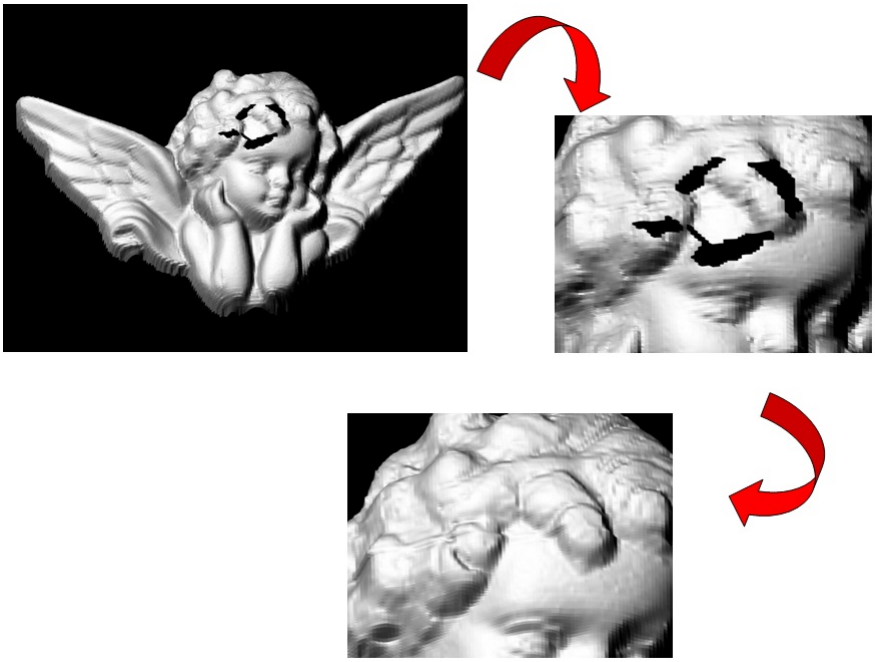

Figura 1.15: Altro esempio di Hole Filling su mesh acquisita

In genere comunque si cerca di scegliere i lati che permettono di creare nuove facce rispettando la curvatura in quel punto ed evitare eventuali errori topologici, come lati non-manifolds e autocompenetrazione della mesh.
Tra questi due edge viene appunto aggiunta una faccia, ossia vengono uniti i due vertici non in comune dei due edge scelti tramite un nuovo edge. Si continua in questo modo in maniera iterativa fino a quando il buco non viene richiuso o non ci sono più coppie di edge adatte.
Un approccio di questo tipo però non è in grado di richiudere tutti i buchi, possono infatti presentarsi situazioni geometricamente e/o topologicamente troppo complesse per i normali algoritmi di triangolazione come questo. Per questi motivi la letteratura scientifica sta proponendo ultimamente molti nuovi metodi alternativi alla triangolazione che siano in grado di trattare anche situazioni più complesse.
Uno di questi metodi è basato sulle funzioni a base radiale (Radial Basis Functions o RBF), l'algoritmo proposto in [8] riempie i buchi espandendo progressivamente le informazioni del *neighborhood*.

1.5 Level of Details

Per ottimizzare la fase di rendering, per esempio, il metodo più comunemente utilizzato in questi casi è la generazione di diversi livelli di dettaglio degli oggetti della scena, ovvero il LOD a cui si è fatto cenno precedentemente. Rappresentando un oggetto distante con un LOD basso e oggetti più vicini con LOD più alti, numerose applicazioni (come videogiochi o tools per la visualizzazione di modelli CAD) possono accelerare in modo significativo il rendering e migliorare quindi l'interattività.
Il concetto di base di questo metodo è molto semplice, si usano versioni più semplici dell'oggetto a seconda del contributo di questo all'immagine da rendere (vedi immagini 1.16,1.17,1.18 e 1.19). La ragione è molto semplice, rappresentare un oggetto su schermo significa che i poligoni che lo compongono devono essere resi e proiettati in un certo numero di pixel, a seconda della grandezza dell'oggetto nella scena. Se quest'oggetto è sullo sfondo e quindi è molto piccolo significa che, senza le approssimazioni LOD, ci si troverebbe a rendere milioni di poligoni i pochissimi pixel e questo genererebbe un overhead intollerabile.

Figura 1.16: Versioni semplificate dello stesso oggetto, il primo è composto da 10,108 poligoni, il secondo da 1,383 poligoni, il terzo da 474 poligoni e l'ultimo da soli 46 poligoni

Per applicare il LOD alle mesh si procede in tre fasi:

- Generation: vengono generati diversi modelli dello stesso oggetto sempre più semplificati (ad esempio utilizzando le progressive meshes, vedi sezione 1.6.2)

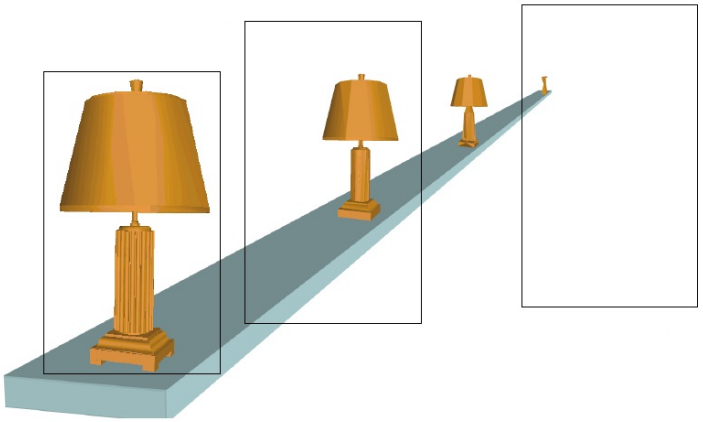

Figura 1.17: Posizionamento delle immagini di figura 1.16 in un immagine

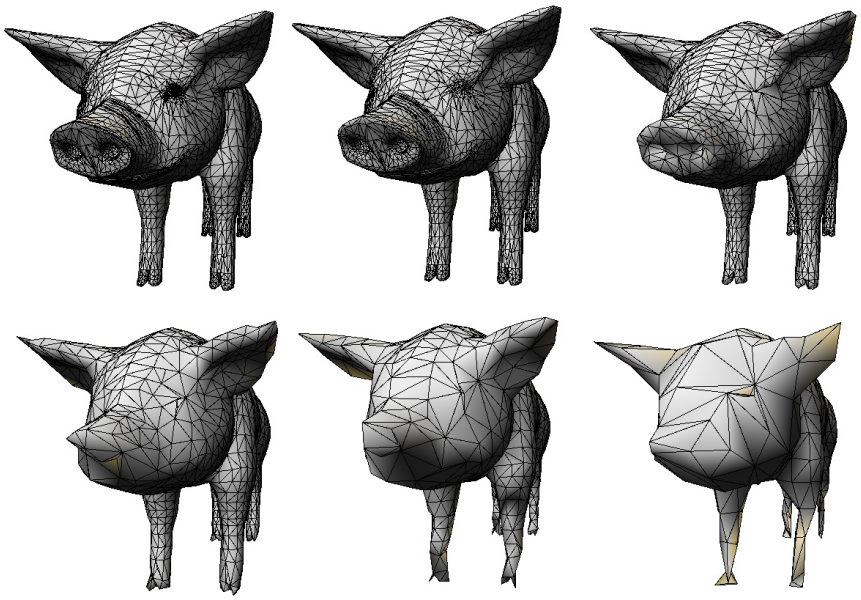

Figura 1.18: Versioni semplificate dello stesso oggetto create dal nostro algoritmo, dall'alto verso il basso e da sinistra verso destra: mesh originale, mesh semplificata al 10%, mesh semplificata al 30%, mesh semplificata al 50%, mesh semplificata al 70%, mesh semplificata al 90%

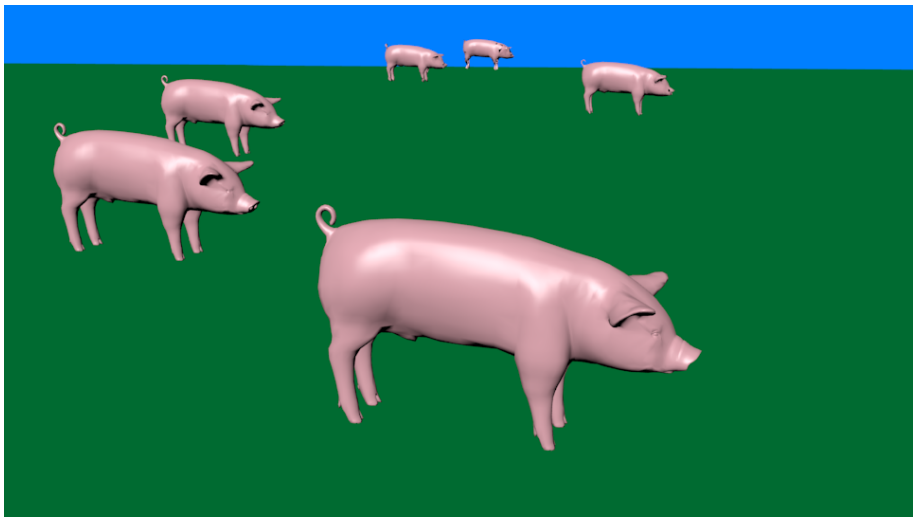

Figura 1.19: Posizionamento delle mesh di figura 1.18 in un immagine

- Selection: si seleziona il modello da impiegare in base ad un certo criterio, ad esempio la semplice distanza dall'osservatore (cosidetto Range-based, ogni LOD è associato ad un set di distanze sempre maggiori) oppure l'area occupata su schermo (detto Projected Area-based, la dimensione che l'oggetto deve avere è dimezzata al raddoppiare della distanza dal piano di proiezione), vedi figura 1.20.

- Switching: riguarda il metodo utilizzato per passare da un modello all'altro in base ai range.

  - Discrete Geometry LOD: switch da un LOD al frame *i* al successivo LOD nel frame *i+1* (crea *popping*).
  - Blend LOD: Vengono resi entrambi i livelli di LOD ma con differenti livelli di trasparenza mano a mano che mi avvicino al punto di switch. In parole povere è come se *sfumassi* da un modello al successivo. È un metodo costoso perché invece di rendere un solo modello ne devo rendere due ma fortunatamente la computazione è scaricabile sull'HW.
  - Continuous LOD: Sfrutto le progressive meshes e la reversibilità delle operazioni di edge collapse e vertex split (vedi sezione 1.6.2).

– Geomorph LOD: Allo switch interpola i vertici del modello complesso tra le posizioni originali e quelle della versione più semplice.

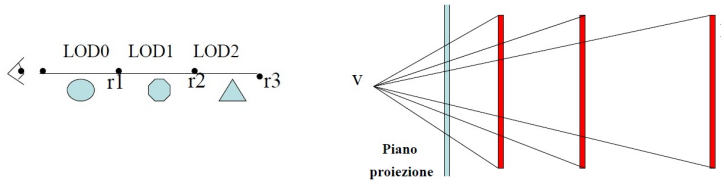

Figura 1.20: Metodi di selezione del livello di LOD, a destra Range-based a sinistra Projected Area-based

1.6 Mesh Decimation

La mesh decimation o semplificazione di mesh, descrive una classe di algoritmi che a partire da una mesh in input ne creano un'altra con meno facce, edge e vertici [12]. Nella letteratura scientifica si possono trovare svariati metodi per la semplificazione di mesh poligonali, la differenza risiede nella classe di problemi a cui sono destinati.
Nel caso preso in esame da questa tesi l'obiettivo è l'ottimizzazione dei tempi di rendering.
Nell'applicazione vista LOD è necessario per esempio avere una sequenza di mesh semplificate dall'originale M_0 ad alta risoluzione, alla mesh a più bassa risoluzione M_n:

$$M_0 \longrightarrow M_1 \longrightarrow ... \longrightarrow M_n$$

attraverso il processo di semplificazione. Poiché la scelta di un algoritmo di semplificazione non è banale, spesso è necessario mantenere una certa accuratezza nel modello semplificato. È necessario infatti definire dei criteri per la valutazione dell'errore commesso a seguito di una semplificazione del modello originale. Alcuni algoritmi controllano il processo di riduzione per mezzo del calcolo della distanza di Hausdorff fra i vertici del modello semplificato e la superficie originale (informalmente, due insiemi di punti A e B sono a distanza di Hausdorff d se ogni punto di A è distante non più di d

dal più vicino punto di B e viceversa). Altri algoritmi utilizzano un limite sulla variazione di volume fra la mesh semplificata e quella originale. Alcuni metodi preservano il genere topologico del modello mentre altri tentano di semplificare anch'esso in modo controllato.
Il metodo proposto in questo elaborato preserva il genere topologico della mesh controllando che non si generino, durante il processo di riduzione, delle situazioni che possono portare a situazioni topologiche non corrette (vedi capitolo 3). In alcuni casi il tempo di esecuzione degli algoritmi di semplificazione può essere una variabile vincolante. Per modelli costituiti da milioni di poligoni, la creazione di mesh semplificate a diverse risoluzioni diventa un processo dispendioso che può durare ore o talvolta addirittura giorni. A seconda dell'applicazione, tempi simili possono rappresentare un lieve svantaggio o qualcosa di assolutamente improponibile.
In sintesi, se il tempo di elaborazione è una variabile importante nella creazione del modello semplificato è consigliabile scegliere di utilizzare algoritmi semplici che forniscano delle semplificazioni più o meno aggressive.

1.6.1 Classificazione delle tecniche di semplificazione

Il trattamento della topologia di una mesh durante il processo di semplificazione permette di dare già una prima importante classificazione degli algoritmi proposti in letteratura.
Praticamente la maggior parte degli algoritmi di semplificazione proposti in letteratura si applicano a mesh manifold, al più con boundary, mentre solo alcuni considerano anche mesh non manifold. È importante notare che non tutti gli algoritmi e software di modellazione garantiscono un output manifold, l'algoritmo proposto da questo elaborato invece fornisce questa garanzia.
Sfortunatamente nella pratica molti modelli non sono esattamente manifold, ed hanno imperfezioni topologiche come spaccature, giunzioni a T, e punti o lati non manifold (vedi capitolo 3). L'algoritmo di semplificazione proposto in questa tesi è topology preserving, cioè mantiene la connettività manifold ad ogni passo. Un tale algoritmo non richiude eventuali buchi nella mesh per cui mantiene anche la topologia globale, quindi il genere della superficie. Siccome nessun buco appare o scompare durante la semplificazione, l'accuratezza visuale dell'oggetto semplificato tende ad essere relativamente

buona (come si potrà constatare nei capitoli successivi).
Tali vincoli, d'altra parte, rappresentano un limite al livello di aggressività che può essere messo in atto dalla semplificazione poiché oggetti di genere elevato non possono essere semplificati oltre ad un certo limite senza chiudere alcuni buchi. Inoltre, un approccio topology-preserving come quello messo in atto richiede che l'input abbia topologia manifold.
Esistono anche algoritmi topology-tolerant, che molto semplicemente ignorano le zone del mesh che violano i requisiti necessari e passano oltre, lasciando tali zone inalterate.
Gli algoritmi che invece modificano la topologia non mantengono necessariamente l'eventuale topologia manifold originale. Questi algoritmi possono quindi richiudere buchi o aggregare componenti connesse durante il processo di semplificazione, permettendo quindi di ottenere drastiche riduzioni del numero di elementi rispetto agli schemi topology preserving. Spesso però con una semplificazione come questa si ottengono risultati di scarsa qualità, con comportamenti poco naturali come l'apparizione o sparizione di buchi da una mesh alla successiva a minor risoluzione.

Praticamente tutte le tecniche di semplificazione proposte in letteratura si basano su una qualche variazione o combinazione di due meccanismi primitivi di base: decimazione e unione di vertici. Esistono tuttavia anche approcci basati su campionamento e suddivisione adattiva. Questa è una possibile classificazione di questi metodi [10]:

- Le tecniche basate su **decimazione** rimuovono iterativamente vertici o facce dalla mesh, ritriangolando il buco risultante ad ogni passo. Tali algoritmi sono relativamente semplici da implementare e possono essere molto veloci. La maggior parte utilizzano solo modifiche locali, il che fa preservare il genere per cui, di nuovo, possono essere limitati nel rapporto di semplificazione. Questi algoritmi sono eccellenti per l'eliminazione di elementi ridondanti, come ad esempio le facce complanari.

- Gli schemi basati sull'**unione di vertici** (**vertex-merging**) collassano due o più vertici in un unico nuovo rappresentante che a sua volta può essere unito ad altri vertici in una fase successiva. Il merge dei vertici di un triangolo equivale all'eliminazione del triangolo stesso, ossia alla riduzione del numero di facce. L'efficienza di queste metodologie

dipende spesso da quale schema si utilizza per decidere quali vertici devono essere uniti ed in quale ordine. In altre parole, in questa classe di algoritmi si trovano metodi grezzi e veloci ma anche accurati e lenti. Una sottoclasse di questi metodi è basata sul *edge-collapsing* in cui ad ogni passo vengono collassati i due estremi di un lato, con conseguente eliminazione di due triangoli. In genere questa sottoclasse contiene metodi topology-preserving, ma alcune versioni di edge-collapsing che possono modificare la topologia sono state proposte.

- Gli algoritmi che usano il **campionamento** catturano la geometria del modello iniziale o direttamente sulla superficie o come approssimazione data dall'intersezione del mesh con una griglia di voxel. Questi approcci sono fra i più elaborati e complicati e spesso le maggiori difficoltà si incontrano nel preservare le alte frequenze per via della perdita di informazione dovuta al ricampionamento. Tali algoritmi sono particolarmente efficienti quando si trattano forme naturali e piuttosto lisce (senza spigoli vivi).

- I metodi basati su **suddivisione adattiva** calcolano una mesh di base semplice che può essere suddiviso ricorsivamente per approssimare sempre più il modello originale. Un tale approccio è ottimo quando il modello base è semplice da calcolare. Ad esempio, il modello di base per un terreno è tipicamente un rettangolo. La realizzazione di mesh a risoluzione crescente richiede che il modello di base preservi le caratteristiche più importanti dell'oggetto, e questo non è sempre banale. Questi metodi preservano la topologia, il che può limitare il loro utilizzo in caso siano richieste semplificazioni aggressive. D'altra parte sono particolarmente adatti per l'editing multirisoluzione di superfici, poiché le modifiche apportate ai livelli di dettaglio grezzi si propagano automaticamente fino ai livelli più alti.

1.6.2 Principali algoritmi di semplificazione

I principali algoritmi proposti in letteratura sono per lo più riconducibili a tre approcci base:

- **Vertex Clustering**
- **Incremental Remeshing**

- **Incremental Decimation**

Tutti e tre fanno uso di diverse combinazioni dei principali operatori di decimazione, che sono:

- **Vertex Remove**: il vertice scelto viene rimosso e di conseguenza anche tutti gli edge e le facce ad esso associate, creando così un buco nella topologia che andrà ritriangolato.
- **Vertex Merging**: due vertici vengono uniti in un unico vertice con conseguente adeguamento della topologia nella zona circostante.
- **Edge Collapse**: l'edge scelto viene collassato in un unico vertice, le due facce tra le quali si trovava vengono eliminate anch'esse.

Altri operatori che non sono propriamente di decimazione in quanto aggiungono elementi alla mesh invece di toglierli sono i seguenti:

- **Vertex Split**: l'operazione duale dell'edge collapse, da un vertice ne vengono creati due, con conseguente edge che li unisce e relative facce.
- **Edge Split**: viene aggiunto un nuovo vertice su un edge prescelto per dividerlo in due parti più corte.
- **Edge Flip**: l'edge scelto viene "ribaltato" andando a connetter due vertici differenti, viene utilizzato per modificare la valenza dei vertici, cioè il loro numero di lati incidenti.

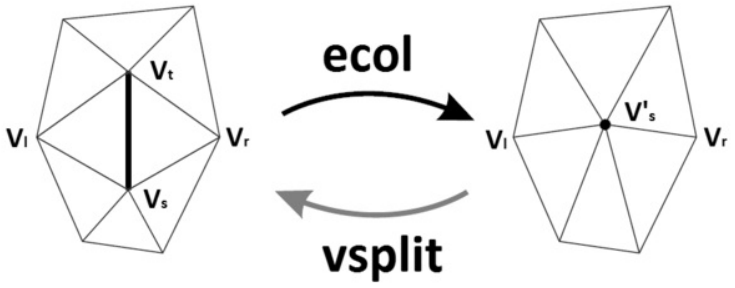

Figura 1.21: Effetti dell'operazioni duali edge-collapse e vertex split

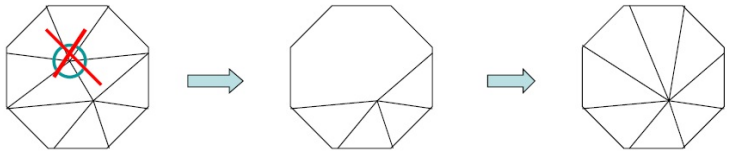

Figura 1.22: Operazione di vertex remove

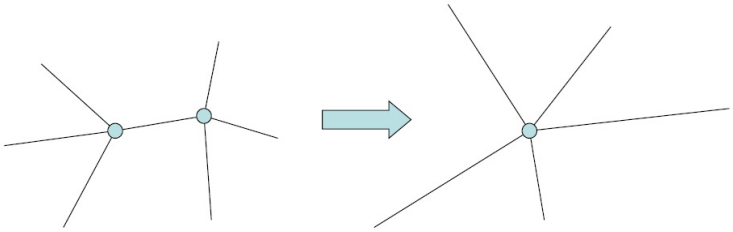

Figura 1.23: Operazione di vertex merging

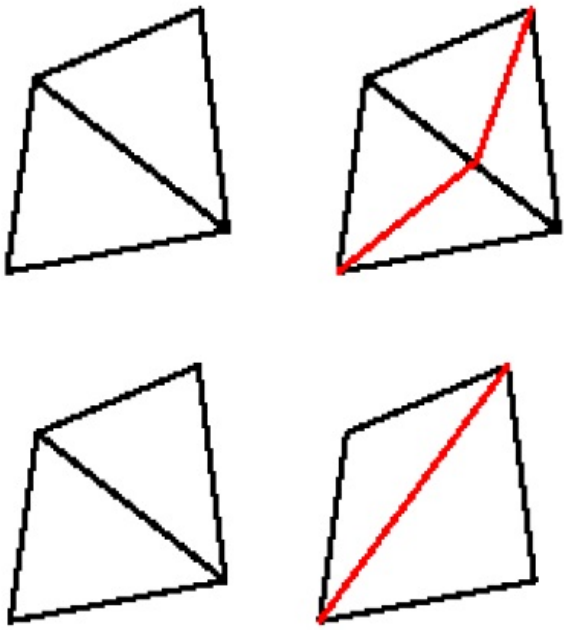

Figura 1.24: Operazioni di edge split (prima riga) ed edge flip (seconda riga)

1.6.3 Vertex Clustering

Questo algoritmo, proposto inizialmente da Rossignac e Borrel [1], è topology insensitive, per cui non richiede né preserva una particolare topologia. Può quindi trattare in modo particolarmente robusto modelli con elementi degeneri che darebbero seri problemi alla maggior parte degli altri algoritmi. L'algoritmo di Rossignac e Borrel inizializza ogni vertice con un valore relativo alla sua importanza all'interno della mesh; i vertici attaccati a facce di grandi dimensioni e vertici ad alta curvatura sono considerati più importanti.
Successivamente l'algoritmo sovrappone una griglia 3D all'oggetto da semplificare e tutti i vertici appartenenti ad una cella vengono collassati in quello (unico) più importante. Naturalmente la qualità della semplificazione risultante dipende dalla risoluzione della griglia utilizzata; una griglia grezza semplificherà il modello in modo aggressivo, mentre una griglia fine darà una riduzione minima. Se durante la decimazione alcuni triangoli si trovano ad avere tutti i loro vertici collassati in un unico rappresentante, questi diventano naturalmente degeneri e vengono eliminati.
Low e Tan [2] hanno proposto un metodo basato anch'esso basato sul clustering ma differente dal normale vertex clustering, chiamato floating cell clustering. Questa tecnica ordina i vertici per importanza ed una cella di dimensioni specificate dall'utente è posizionata sul vertice più importante. L'algoritmo collassa tutti i vertici interni alla cella ed anche in questo caso elimina i triangoli degeneri come nel caso precedente qualora se ne venga a formare qualcuno.
A questo punto si posiziona una nuova cella sul successivo vertice più importante e si ripete l'operazione. L'eliminazione della griglia sottostante riduce notevolmente la sensibilità dellalgoritmo alla posizione ed all'orientamento del modello.
Recentemente, l'algoritmo di Rossignac e Borrel è stato migliorato [3] usando la metrica proposta da Garland e Heckbert (vedi paragrafi seguenti) per la valutazione dell'errore e conseguentemente per il posizionamento ottimale del vertice rappresentativo di ogni cella.
Il metodo originale di Rossignac e Borrel (e la sua estensione descritta) che classifica i vertici all'interno di una griglia 3D, è estremamente veloce e di complessità lineare proporzionale al numero di vertici. La variante di Low e Tan è appena più lenta per via dell'ordinamento dei vertici per importanza

($O(n\ log\ n)$), con n numero dei vertici). Questa categoria di metodi ha alcuni svantaggi in quanto non preservano la topologia e non garantiscono nulla circa l'errore introdotto a seguito di semplificazione.

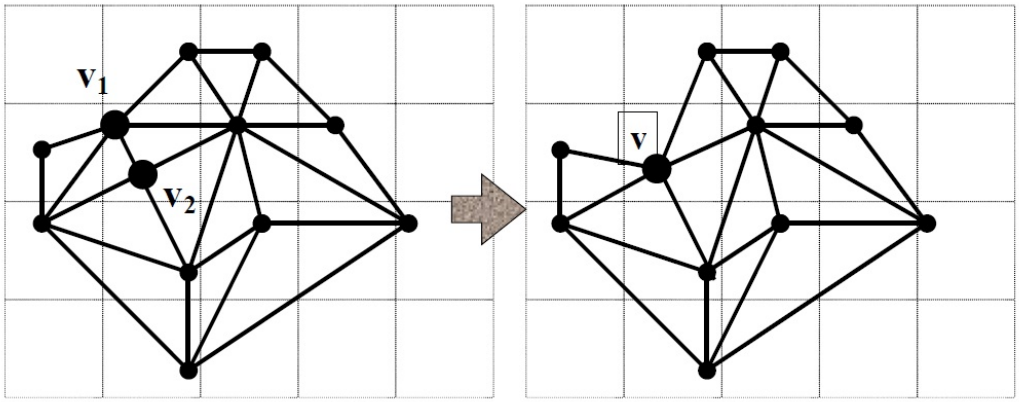

Figura 1.25: Esempio di clustering per i vertici appartenenti ad una cella

1.6.4 Incremental Remeshing

Il metodo di Incremental Remeshing, proposto da Kobbelt e Botsch nel 2004 [11], pone come obiettivo l'uniformazione della lunghezza degli edge componenti la mesh, sulla base di una lunghezza specificata inizialmente. Per raggiungere questo risultato si agisce seguendo un ciclo composto da una serie di step che utilizzano gli operatori mostrati precedentemente.
Il primo passo consiste nell'applicare l'edge split a tutti quegli edge la cui lunghezza supera una certa soglia massima, dividendoli così in edge più piccoli. Successivamente vengono collassati (edge collapse) tutti i lati la cui lunghezza non supera la lunghezza minima richiesta e si applica l'operatore di edge flip per cercare di portare tutti i vertici ad avere una valenza uguale a sei.
L'ultima operazione applicata è il **Tangential Smoothing**, ed è un operazione che si applica ai vertici. Il vertice scelto viene spostato sul suo piano tangente verso il centro di massa del first-ring a cui appartiene, in questo modo si uniforma la distribuzione dei punti sulla superficie della mesh.

A questo punto si riparte dal primo step iterando fino a che non si è raggiunto l'obiettivo voluto.

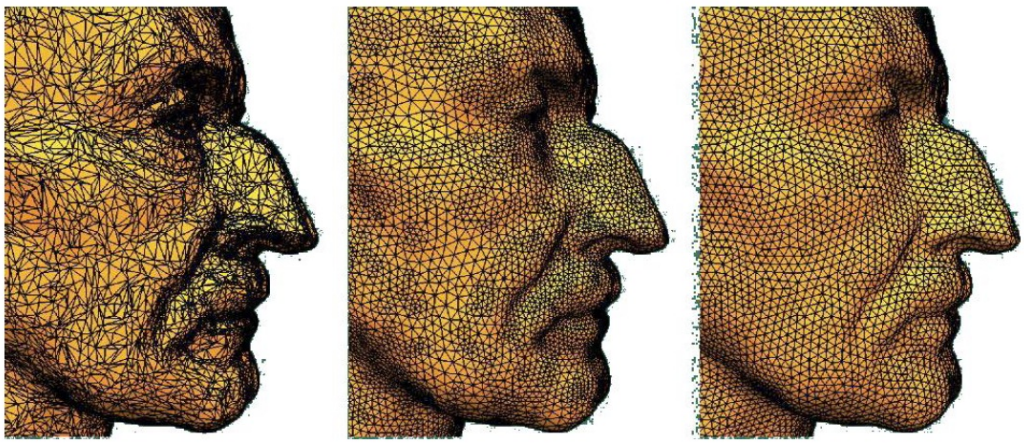

Figura 1.26: Esempio di incremental remeshing

1.6.5 Incremental Decimation

Quelli basati sull'Incremental Decimation sono forse gli algoritmi più famosi.
L'idea di base è piuttosto semplice. La prima operazione è una stima per ogni vertice,edge e faccia del costo di rimozione in base all'errore introdotto così facendo. Ci sono varie tecniche per stimare l'errore: distanza tra vertici, distanza fra normali, distanza vertice-piano.
Viene compilata una lista di tutti gli elementi ordinata per costo crescente. Da questa si parte con la decimazione, se l'errore introdotto dalla decimazione corrente è inferiore ad una certa tolleranza allora questa viene applicata e si passa alla successiva. L'algoritmo termina quando nessuna decimazione è più possibile per la tolleranza richiesta.

1.6.6 Quadric Error Metrics

Questo algoritmo basato sul vertex merging, proposto da Garland e Heckbert nel '97, rappresenta un buon compromesso fra robustezza, velocità e

resa.
L'algoritmo effettua iterativamente l'unione di coppie di vertici, non necessariamente unite da un lato. Le coppie di vertici, per così dire, eleggibili per essere unite sono tutte quelle unite da un lato ed anche tutte quelle distanti meno di un certo valore di tolleranza t, fissato dall'utente.
Come accennato nella sezione precedente, il contributo più significativo di questo lavoro è il modo in cui l'errore viene calcolato ed accumulato passo dopo passo. Inizialmente viene calcolato l'errore E_j relativo ad ogni vertice p_j, tramite la seguente formula:

$$E_j(p_j) = \sum_{i=1}^{n} (n_i p_j - d_i)$$

Che non è altro che il calcolo della distanza vertice-piano. Gli isocontorni di questa funzione sono ellissoidi e di conseguenza la misura dell'errore risultante è chiamata *quadric error metric* (QEM) [1,3]. Questa operazione viene eseguita tramite una sommatoria su tutte le facce adiacenti a p_j. L'errore E_j è nullo sulla mesh originale. Il dato che interessa è però la distanza media quindi E_j viene normalizzato dividendolo per la valenza di p_j. A questo punto quando l'edge tra due vertici p e q andrà collassato, il quadric error per il nuovo vertice r sarà dato da:

$$E_r = (E_p - E_q)/2$$

La posizione ottimale per r si ottiene risolvendo questa formula:

$$(\sum_{i=1}^{n} (n_i n_i^T)) = (\sum_{i=1}^{n} (n_i d_i))$$

I valori vengono conservati all'interno di una matrice 4x4, rappresentante appunto la somma delle distanze al quadrato del vertice dai piani dei triangoli in esso incidenti.

L'errore (ossia la matrice) introdotto da un'operazione di vertex merging è associato al nuovo vertice, ed è semplicemente la somma delle matrici dei vertici appena uniti. All'inizio tutte le coppie di vertici sono ordinate in una coda a priorità a seconda dell'errore che si introdurrebbe a seguito di una loro eventuale unione. L'algoritmo unisce prima la coppia che introduce

Figura 1.27: In questa immagine sono mostrati tre livelli di dettaglio ottenuti utilizzando il metodo di Garland e Heckbert

l'errore minimo, aggiorna la coda di conseguenza e ripete l'operazione fino a quando non ci sono più coppie valide. Le quadric error metrics forniscono un metodo semplice e veloce per guidare il processo di semplificazione con costi di memorizzazione relativamente contenuti. L'algoritmo non richiede topologia manifold, e non preserva la topologia, fornendo semplificazioni anche molto aggressive.

1.6.7 Progressive Meshes

Questo tipo di approccio è stato proposto inizialmente da Hoppe, nel '96. I modelli poligonali vengono rappresentati come sequenza di operazioni di edge-collapse. Hoppe ha dapprima introdotto le mesh progressive sotto forma di algoritmo di semplificazione dinamico per manifolds poligonali generici [5], in un successivo lavoro ha ottimizzato il processo di semplificazione tenendo conto del punto di vista [6].
Quindi una progressive mesh consiste appunto in una sequenza di mesh a risoluzione decrescente, ottenute a partire da una mesh originaria tramite una serie di operazioni di edge-collapse. È possibile risalire alla mesh precedente all'applicazione dell'edge-collapse tramite una sequenza di vertex-splits, ovvero l'operazione duale dell'edge-collapse. Ogni split sostituisce ad un vertice una nuova coppia di vertici connessi da un lato e due nuovi triangoli incidenti nel nuovo lato. Se si tiene traccia degli edge collapse eseguiti, sarà quindi possibile ricostruire la mesh originale a partire dalle mesh a bassa risoluzione.
Di fatto, la sequenza di operazioni di edge-collapse codifica una sequenza di livelli di dettaglio dal più definito, equivalente alla mesh originaria, al più grezzo, equivalente alla mesh "base" ottenuta alla fine. Le operazioni di

collapse e di split sono abbastanza veloci da essere applicate in tempo reale per ottenere semplificazioni dipendenti dal punto di vista.

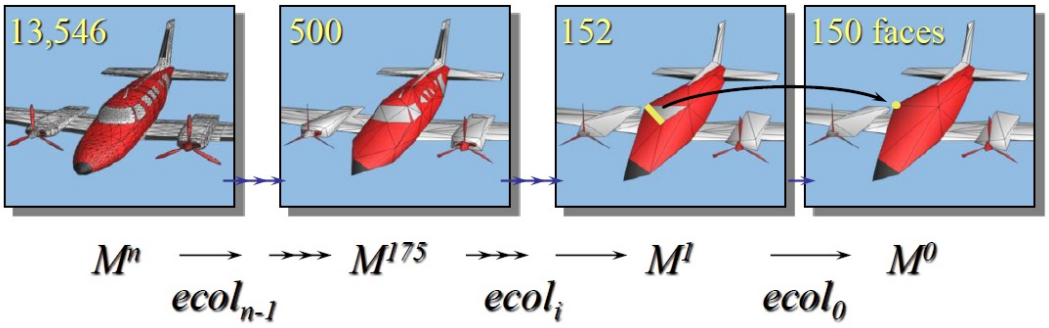

Figura 1.28: Esempio di progressive meshes

Capitolo 2

Acquisizione oggetti con scanner 3D

Durante una prima fase della tesi, l'obiettivo è stato acquisire dei modelli mesh poligonali da oggetti reali, da poter poi rielaborare e semplificare con l'algoritmo finale. Questa fase si è quindi svolta nel laboratorio di High Performance Graphics, Computing and Visualization del CIRAM, presso la sede di ingegneria di Bologna, per poter utilizzare lo **Scanner Laser 3D - NextEngine** ed una volta acquisita una certa praticità con lo strumento, si è svolta una serie di acquisizioni di oggetti di varia natura e differenti caratteristiche.
In questo capitolo verranno mostrate le varie acquisizioni fatte, complete dei dati ricavati durante l'elaborazione.

2.1 Scanner Laser NextEngine

Lo scanner presente nel laboratorio del CIRAM è dotato di un software proprietario (ScanStudio) per la ricostruzione della mesh a partire dalle nuvole di punti acquisiti ed è fornito di due piattaforme per effettuare le scansioni, la prima, denominata **Autodrive** (vedi figura 2.2) è una piastra che ruota sull'asse z e permette di fare delle acquisizioni di oggetti a 360° ed ha un solo grado di libertà, per questo è indicata nei casi in cui non siano richiesta scansioni della parte superiore degli oggetti. Permette di fare acquisizioni di oggetti che variano per dimensione tra 7,62x12,7 cm e 25,4x33,2 cm (modalità macro e wide).

Figura 2.1: Scanner Laser 3D - NextEngine

In caso contrario è assolutamente consigliato di utilizzare il secondo device, il **MultiDrive** (figura 2.2), che permette di fare acquisizioni anche della parte superiore e inferiore degli oggetti: infatti il supporto per le acquisizioni può ruotare su due assi. Su un asse possiamo fare acquisizioni a 360° mentre sull'altro possiamo variare l'angolo di incidenza della scansione da -35°, acquisendo quindi la parte sottostante agli oggetti, a +45° acquisendo la parte superiore degli oggetti.
Le due rotazioni possono essere combinate per coprire quindi quasi tutti i lati dell'oggetto da scansionare. Purtroppo però con questodevice è solo possibile acquisire oggetti che hanno dimensione massima di 7,62x12,7 cm (solo modalità macro).

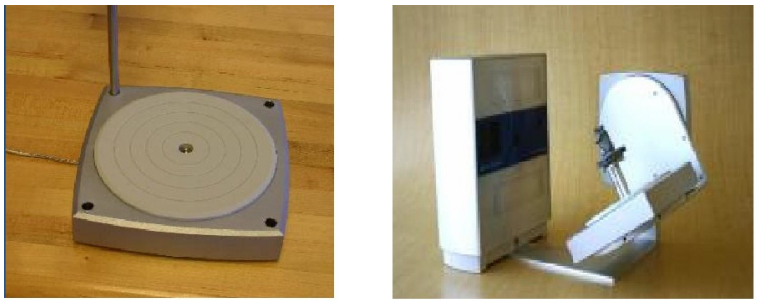

Figura 2.2: Autodrive e MultiDrive

Si capisce subito che questa è una forte limitazione, infatti si è visto durante le acquisizioni che scansionare le parti superiori di oggetti con misure "al limite" è veramente complicato e richiede molteplici scansionamenti delle stesse zone dell'oggetto.
La modalità **macro** richiede il posizionamento dell'oggetto ad una distanza indicativa di circa 16-17cm ed è possibile acquisire oggetti non più grandi di 7,62x12,7 cm circa. Le acquisizioni hanno precisione di circa 0,0127 cm. La risoluzione varia tra i 6200 a 77 punti su centimetro quadrato e la dimensione dei triangoli generati varia dai 0,032 a 0,290 centimetri quadrati. Diminuendo la qualità della scansione aumenta considerevolmente la velocità di acquisizione dei dati di circa un 30%. Per le acquisizioni **wide** invece gli oggetti sono posizionati a circa 40-45 cm e le acquisizioni hanno precisione di circa 0,0381 cm con una finestra di scansione grande circa 25,4x33,2 cm. La risoluzione si abbassa considerevolmente e varia tra 680 e 8 punti per centimetro quadrato e la dimensione dei triangoli generati va da 0,096 a 0,871 centimetri quadrati (vedi tabella 2.8).

2.2 Acquisizioni

In figura 2.3 è illustrata la pipeline di acquisizione. Tramite lo scanner vengono acquisite una serie di *range image* che coprono l'intera superficie dell'oggetto.
Queste vengono allineate in modo da ricostruire la nuvola tridimensionale di punti dell'oggetto. A partire da questa è possibile costruire poi il modello digitale dell'oggetto scansionato e andare ad agire su di esso ri-

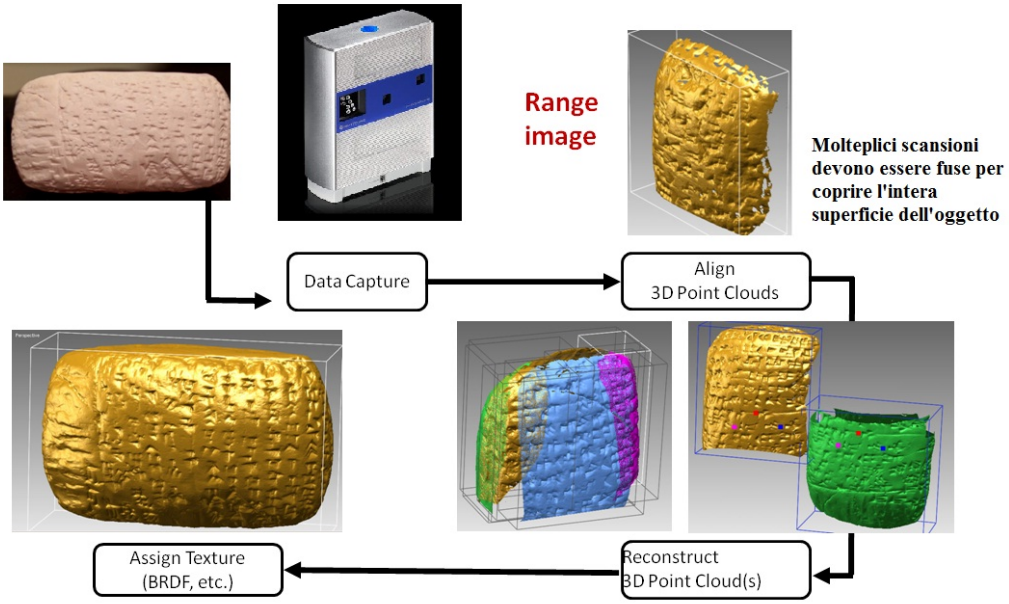

Figura 2.3: Pipeline di acquisizione

muovendo le irregolarità e i punti in eccesso(in figura 2.4 un altro esempio di acquisizione).

Si e scelto di utilizzare il device Multidrive per le acquisizioni. Con questo device per ogni singolo oggetto acquisito vengono eseguite una serie di scansioni successivamente raggruppate in una *scan family* e contrassegnate da una lettera dell'alfabeto, vedi figura 2.5.

Ogni scan family è caratterizzata da un angolo β di acquisizione dello scanner (Tilt), vedi fig. 2.6. Tenendo fisso β l'oggetto viene fatto ruotare n volte (**Divisions**) sull'asse Z, andando a coprire quindi una rotazione completa a 360°, catturando cosi tutti i lati dell'oggetto. Ogni oggetto scansionato è dunque formato da più *scan family* a diversi *Tilt*, e ogni *Tilt* è caratterizzato da n scansioni.

Il software di default permette di allineare automaticamente le scansioni (scelta consigliata) o di allinearle manualmente (Scan → Disable Scantime autoalignment).

È possibile eseguire scansioni singole, per esempio delle zone in cui la

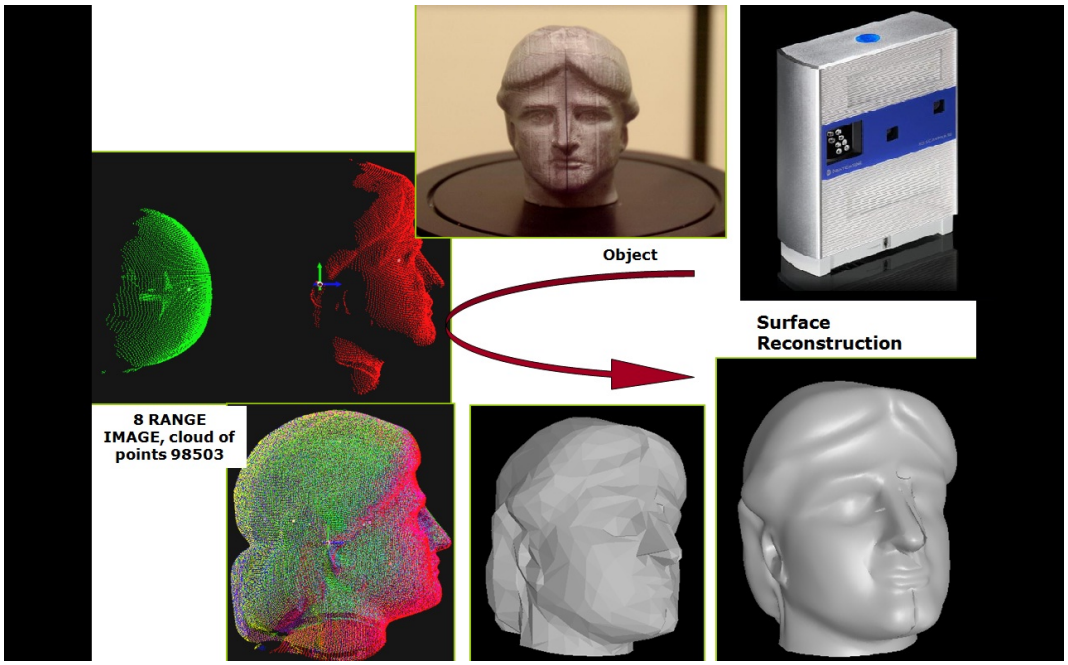

Figura 2.4: Altro esempio di pipeline di acquisizione

Figura 2.5: Selezione delle scan family

scansione non è risultata soddisfacente. Lo scanner inoltre permette di acquisire una texture relativamente alla scansione in corso e di associarla automaticamente alla scansione. Ci sono diversi parametri che si possono settare relativamente all'immagine che si possono trovare nel menù "Scan → Settings → Texture Capture Mode".

Il software ScanStudio permette di specificare (vedi figura 2.7) la natura della scansione: 360 gradi, Bracket (3 scansioni, da davanti, da destra e da sinistra, in pratica da 3 angoli differenti che vanno a formare circa 180°) e Singola.
Le ultime due opzioni selezionabili dal menù sono il **Target** e la qualità. Il primo offre tre possibilità: dark (per le superfici molto scure), neutral e light (da utilizzare quando la l'illuminazione ambientale è bassa, ad esempio

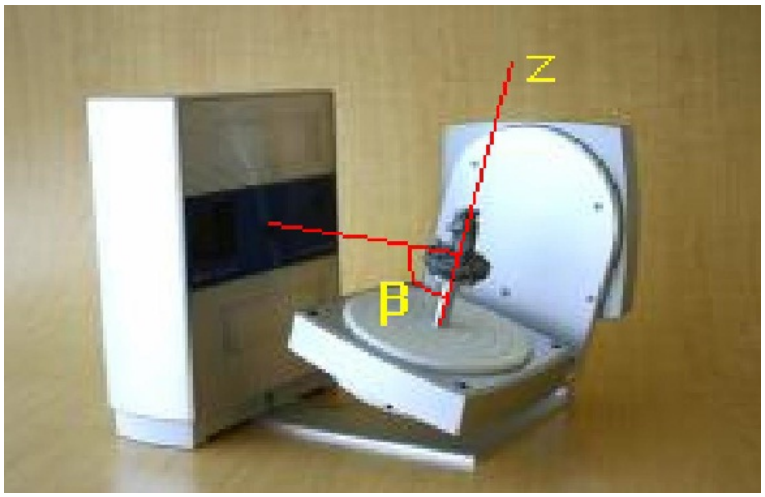

Figura 2.6: Dispositivo Multidrive inclinato di un angolo β

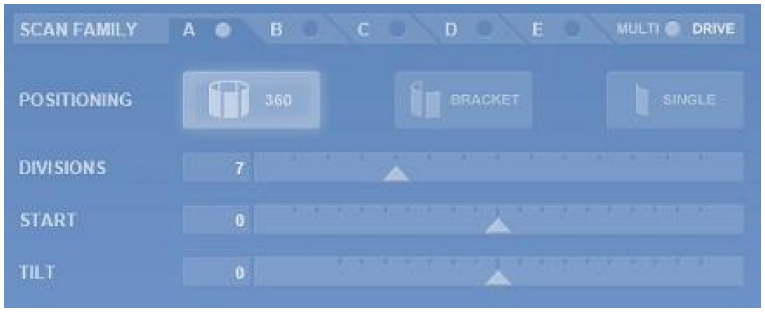

Figura 2.7: Pannello di selezione iniziale

per fare scansioni in camera oscura).

L'opzione **quality** permette di scegliere il numero di punti/vertici che andranno a comporre la nostra *range image*, si può scegliere tra una scansione veloce (quick), SD e HD. Tutte e tre sono formate da tre diversi gradi di qualità; ScanStudio utilizza una nomenclatura basata su scala numerica, da 1 a 9:

- 1,2,3 $\Rightarrow$ *Quick*
- 4,5,6 $\Rightarrow$ *SD*

- 7,8,9 $\Rightarrow HD$

Naturalmente anche in questo caso più è alta la qualità richiesta più aumenta il tempo richiesto dalla scansione.

		Punti/in²	Grandezza triangolo in²	Punti/cm²	Grandezza triangolo cm²	
9	HD	40000	0,0050	6200,0124	0,0323	
8		17000	0,0075	2635,0053	0,0484	
7		10000	0,0100	1550,0031	0,0645	
6	SD	4400	0,0150	682,0014	0,0968	
5		2500	0,0200	387,5008	0,1290	
4		1600	0,0250	248,0005	0,1613	
3	Quick	1100	0,0300	170,5003	0,1935	
2		700	0,0375	108,5002	0,2419	
1		500	0,0450	77,5002	0,2903	MACRO

9	HD	4400	0,0150	682,0014	0,0968	
8		2000	0,0225	310,0006	0,1452	
7		1100	0,0300	170,5003	0,1935	
6	SD	500	0,0450	77,5002	0,2903	
5		280	0,0600	43,4001	0,3871	
4		180	0,0750	27,9001	0,4839	
3	Quick	125	0,0900	19,3750	0,5806	
2		80	0,1125	12,4000	0,7258	
1		55	0,1350	8,5250	0,8710	WIDE

Figura 2.8: Tabella riassuntiva della precisione delle scansioni

Nella tabella di figura 2.8 viene illustrata la precisione delle varie scansioni ordinate in base alle modalità selezionabili tramite il pannello di selezione iniziale. A seconda della densità dei punti varia considerevolmente la dimensione dei file generati (da decine di MB per bassa risoluzione a centinaia di MB per lalta risoluzione).
Tutti i modelli mostrati nelle sezioni successive sono stati ricostruiti automaticamente dal software ScanStudio utilizzando la funzione di *auto-alignment* a partire dalle range image acquisite ad ogni scansione.

2.2.1 Palma

La prima acquisizione eseguita è stata dell'oggetto palma, un oggetto di prova fornito insieme allo scanner, in teoria doveva essere un oggetto relativa-

mente semplice da acquisire, in realtà si è dimostrato abbastanza complesso in quanto lo sua altezza eccede quasi le misure limite.

Figura 2.9: Palma, fotografata dallo scanner prima dell'aquisizione

Questo è il setting utilizzato (per eventuali dubbi sui valori della tabella si rimanda alla sezione precedente):

- Target : Neutral
- Quality : 7
- Divisions : 7 (Default)

Sono state effettuate 6 scansioni per poter acquisire bene l'oggetto, una a 360° e le ristanti singole in punti in cui si erano creati degli *hole*, ovvero dei punti nei quali lo scanner non era riuscito ad acquisire dei punti creando così un vero e proprio "buco" nella figura tridimensionale. Ecco i dati delle varie scansioni raggruppati per scan family:

- 1) 360°, start 0, tilt -11°
- 2) single, start 0, tilt -9°
- 3) single, start -76, tilt +45°
- 4) single, start -88, tilt -22°
- 5) single, start 88, tilt +25°
- 6) single, start -8, tilt +45°

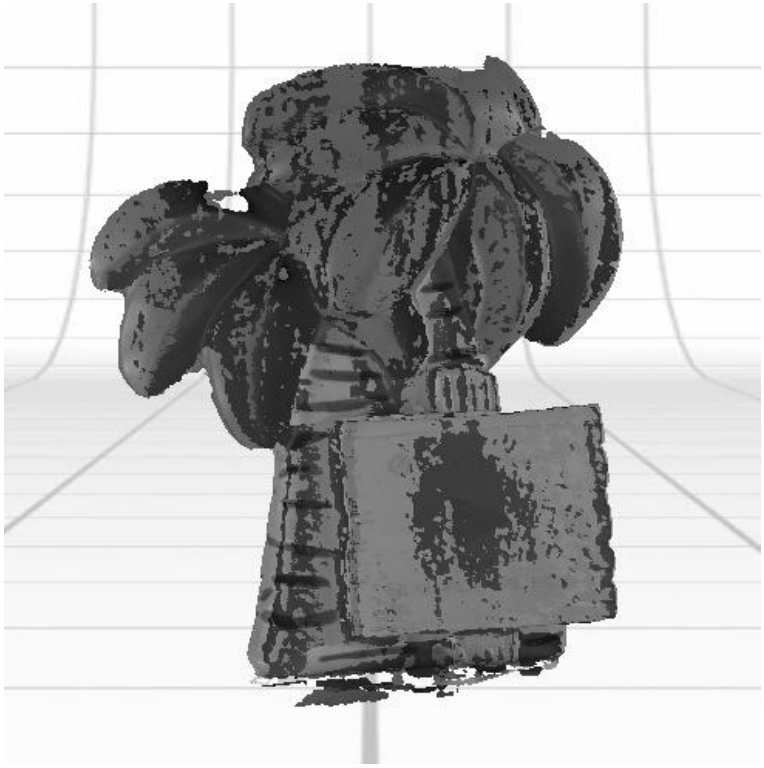

Figura 2.10: Palma, risultato finale, shaded

L'unione di tutte le scansioni in un'unica nuvola di punti è mostrato in figura 2.10. Si tratta di un modello con 765711 facce per un totale di 395349 vertici. Già da questa scansione si può notare come il numero di punti che compongono la nuvola creata sia enorme, questo perché sono

state necessarie numerose scansioni, spesso sulle stesse zone, per acquisire l'oggetto decentemente.

2.2.2 Puffi

Il secondo soggetto acquisito (mostrato in figura 2.11) è in realtà una composizione di più oggetti.

Figura 2.11: Puffi, fotografato dallo scanner prima dell'aquisizione

Per l'acquisizione è stato utilizzato questo setting:

- Target : Neutral
- Quality : 7
- Divisions : 7 (Default)

E le scansioni necessarie sono state sei:

- 1) 360°, start 0, tilt +15°
- 2) 360°, start 2, tilt -28°

- 3) single, start -8, tilt 11°
- 4) single, start -88, tilt 13°
- 5) come la 1 solo con target Dark
- 6) come la 2 solo con target Dark

In pratica sono state fatte due scansioni da 360° per la parte superiore e quella inferiore, le due singoli servivano a riempire qualche buco. Buchi che in realtà non sono stati riempiti del tutto in quanto c'erano zone molto scure di colore nero nelle quali non si riusciva ad ottenere una buona resa. Il problema è stato risolto effettutando due nuove scansioni a 360° uguali alle prima ma con target dark.
In totale la mesh ottenuta aveva 467806 triangoli per un totale di 257705 punti, dopo una fase di *trimming* per eliminare le parti inutili come il piedistallo si è passati a 368145 triangoli per 197430 punti.
In fig. 2.12 è mostrato il risultato finale in versione shaded e a colori:

Figura 2.12: Puffi, risultato a colori e shaded

Come si può vedere in figura sull'oggetto finale c'è molto *noise* che in casi come questo, di acquisizione con scanner 3D è molto frequente, può essere eliminato applicando delle tecniche di *fairing*.

2.2.3 Elefante

Per l'oggetto elefante, mostrato in figura 2.13, sono state effettuate due scansioni, in ordine temporale la prima è stata fatta con qualità bassa e la seconda in HD per vedere le differenze.

Figura 2.13: Elefante, fotografato dallo scanner prima dell'aquisizione

Le impostazioni iniziali sono state queste, come si può notare la qualità richiesta è molto bassa:

- Target : Neutral
- Quality : 3
- Divisions : 7 (Default)

E le scansioni 3:

- 1) 360°, start 0, tilt +1°
- 2) 360°, start 2, tilt +35°
- 3) single, start 38, tilt -11°

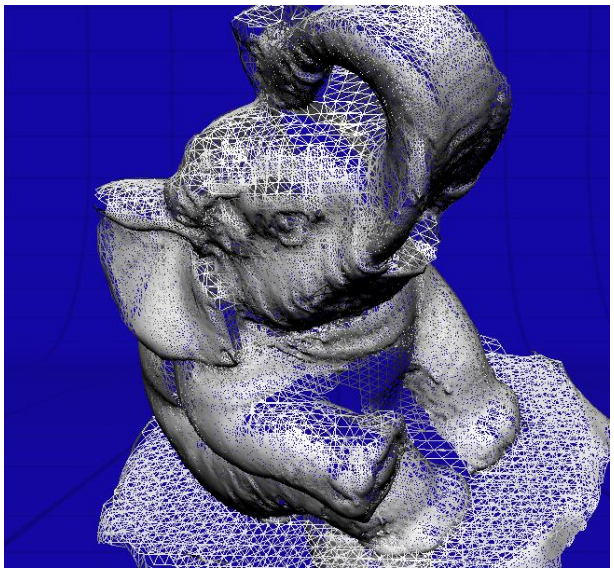

Figura 2.14: Elefante, mesh finale

La figura 2.14 mostra il risultato finale, notare come la densità di triangoli sia inadeguata per una mesh così complessa. La figura 2.15 mostra come lo scanner sia riuscito anche ad acquisire la polvere che ricopre l'oggetto in alcuni punti.
Considerando la qualità impostata inizialmente il risultato tutto sommato è buono anche se nelle concavità la scannerizzazione ha lasciato dei buchi. La bassa qualità si vede anche dal numero di punti finale 38931 per un totale di 68994 triangoli.

La seconda scansione è stata invece fatta in HD. Come preventivato ha richesto un tempo decisamente superiore rispetto alla prima, circa 45 minuti, contro i 30 minuti circa della prima.
Si è visto subito che impostando una qualità più alta è bastata una singola scansione a 360° per ottenere già un buon risultato. Le scansioni singole che

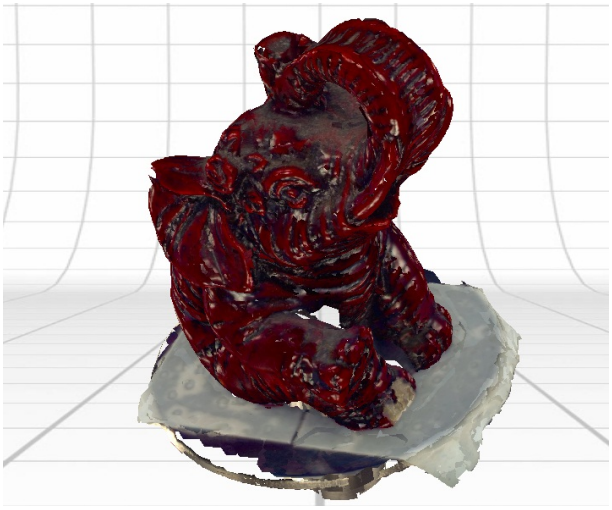

Figura 2.15: Mesh, immagine finale a colori

sono state fatte in seguito sono servite per ottenere una buona scansione delle parti nascoste dalla posizione della proboscide.

Setting iniziale:

- Target : Neutral
- Quality : 7
- Divisions : 7 (Default)

Scansioni:

- 1) 360°, start 0, tilt +15°
- 2) single, start -78, tilt 0°
- 3) single, start 0, tilt +45°
- 4) single, start -55, tilt +45°
- 5) single, start 0, tilt -45°

I risultati del confronto sono stati quelli attesi, il numero di vertici di cui è composta la mesh è 341571, superiore di circa dieci volte al numero di punti trovati nel primo caso, per un totale di 419957 triangoli. I valori sono in linea con i dati forniti nella tabella di figura 2.8.

Figura 2.16: Mesh, risultato finale della seconda aquisizione

2.2.4 Gatto

L'oggetto gatto mostrato in figura 2.17, è stato utilizzato per sperimentare l'acquisizione con scanner laser 3D su di un materiale riflettente, questo dovrebbe impedire agli scanner laser di fare una buona scansione e infatti così è stato. Quindi si è cosparso l'oggetto con la polvere opacizzante fornita con il laser e si è proceduto ad una seconda scansione i cui esiti sono stati molto più soddisfacenti.

Setting iniziale:

- Target : Neutral
- Quality : 7

Figura 2.17: Gatto, fotografato dallo scanner prima dell'aquisizione

- Divisions : 7 (Default)

Queste sono state le scansioni:

- 1) 360°, start 0, tilt 0°
- 2) 360°, start 1, tilt +15°
- 3) single, start 0, tilt 0°
- 4) single, start 1, tilt +15°

Già la prima scansione aveva acquisito la quasi totalità dell'oggetto, le altre tre sono state fatte nel tentativo di scansionare anche la parte inferiore dell'oggetto, che presentava una concavità molto accentuata, purtroppo lo scanner non è stato in grado di acquisire quella parte e quindi la mesh finale presenta un "buco" in corrispondenza della parte inferiore dell'oggetto.
Si è cercato di eseguire un'operazione di *hole filling* tramite la funzionalità corrispondente fornita dal software ma questa non è stata in grado di risolvere il problema.

Notare in figura 2.18 come la mesh sia affetta da noise, derivante dal materiale di cui l'oggetto è composto. Anche in figura 2.19 è possibile notare un errore di acquisizione sulla zampa del gatto e il buco derivante dai problemi di scansionamento dello scanner nella parte concava dell'oggetto.

2.2.5 Vishnu

Come si può vedere dall'immagine 2.20, l'oggetto in questione è fatto di un materiale opaco particolarmente scuro, per questo la gran parte delle scansioni sono state effettuate con il target settato su dark, questo per poter meglio acquisire le concavità dell'oggetto che con una scansione normale non erano state scansionate propriamente.

Il setting iniziale utilizzato è stato questo:

- Target : Neutral*
- Quality : 7
- Divisions : 7 (Default)

Figura 2.18: Gatto, risultato finale della scansione

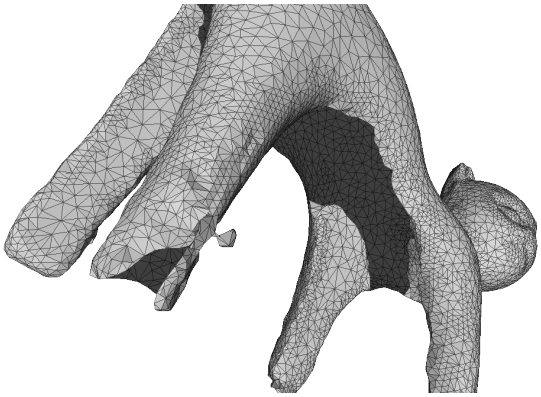

Figura 2.19: Gatto, errore di acquisizione e hole

Figura 2.20: Vishnu, fotografato dallo scanner prima dell'aquisizione

* A partire dalla terza scansione si è passati al Dark.
Queste sono state le scansioni:

- 1) 360°, start -13, tilt 19°
- 2) Bracket, start -6, tilt +25°
- 3) single, start 69, tilt +11°
- 4) single, start 44, tilt +45°
- 5) single, start 93, tilt -8°
- 6) single, start -26, tilt +25°
- 7) Bracket, start -8, tilt +17°

Il problema con questo oggetto, oltre alla colorazione è stata l'altezza, che era quasi al limite consentito, questo ha fatto sì che si siano dovute effettuare molte scansioni singole della parte superiore in punti dove lo scanner faceva fatica ad arrivare. L'alto numero di scansioni ha fatto si che il numero di punti crescesse molto (431852 punti per 800547 facce), per questo si è provveduto subito ad eliminare le parti inutili (piedistallo) e si è passati a 372150 punti per 702977 facce.

Figura 2.21: Vishnu, risultato finale dell'acquisizione

Figura 2.22: Vishnu, risultato finale shaded senza piedistallo

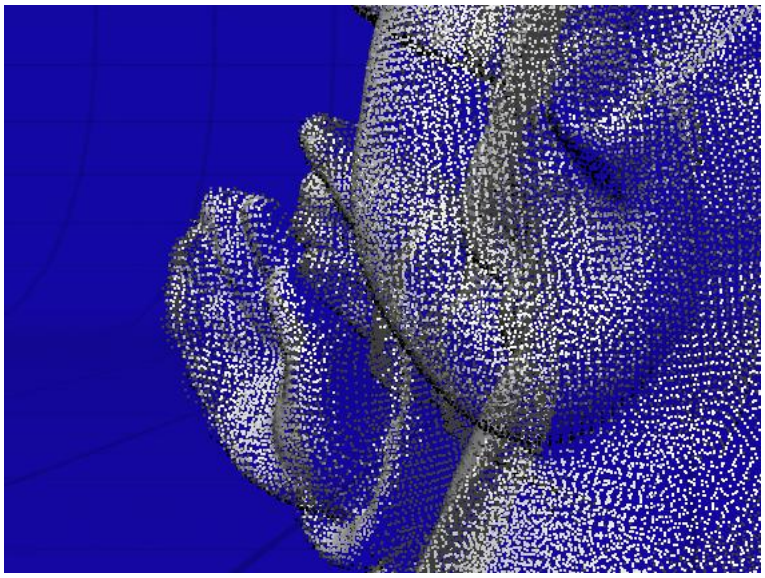

Figura 2.23: Vishnu, particolare della nuvola di punti sulla proboscide

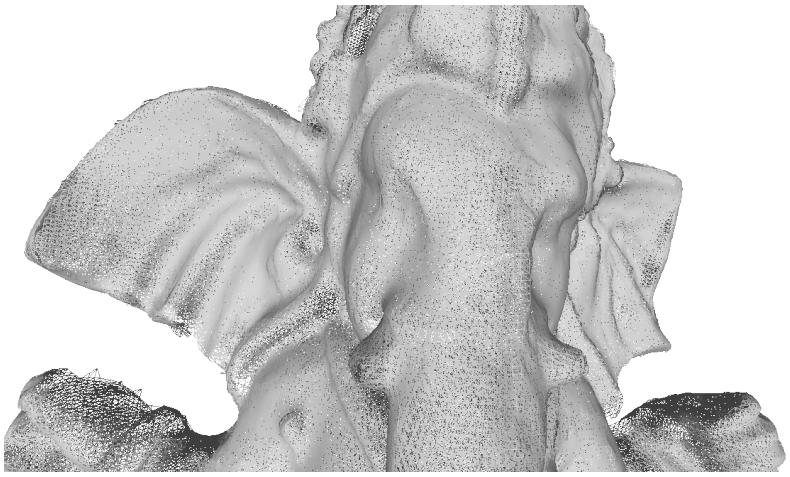

Figura 2.24: Vishnu, modalità wireframe, notare come la enorme densità di triangoli faccia sembrare la superficie chiusa

2.2.6 Dentiera

La dentiera mostrata in figura 2.25 è stato un oggetto abbastanza facile da acquisire, è bastato eseguire solo due scansioni, una a 360° leggermente inclinata per avere una scannerizzazione migliore della parte superiore dei denti e una singola per ricavare alcuni particolari dell'interno della bocca.

Figura 2.25: Dentiera, fotografata dallo scanner prima dell'aquisizione

Questo è stato il setting iniziale:

- Target : Neutral

- Quality : 7
- Divisions : 7 (Default)

E queste le scansioni:

- 1) 360°, start 1, tilt 30°
- 2) single, start -180, tilt 45°

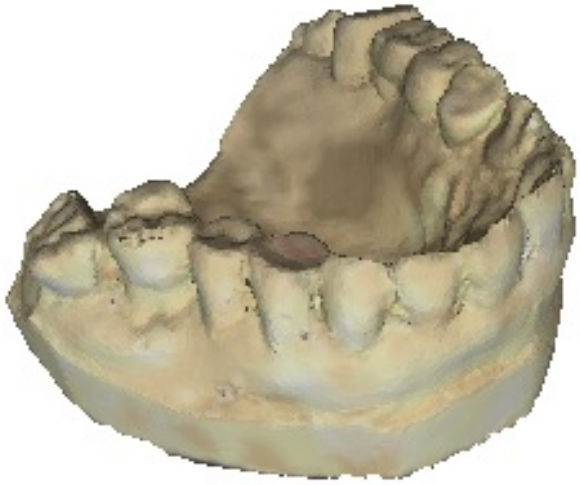

Figura 2.26: Dentiera, risultato finale dell'aquisizione

La mesh dentiera finale è mostrata in figura 2.26, ha 28954 vertici per un totale di 42926 facce.

2.2.7 Nano

Questo è stato l'ultimo oggetto acquisito, sembrava relativamente semplice ma si è rivelato abbastanza complicato in quanto anche questo, come il precedente, è al limite della tolleranza per quanto riguarda l'altezza.

Figura 2.27: Nano, fotografato dallo scanner prima dell'aquisizione

Setting iniziale:

- Target : Neutral
- Quality : 7
- Divisions : 7 (Default)

Scansioni:

- 1) 360 gradi, start 0, tilt 15
- 2) 360 gradi, start 0, tilt -15
- 3) single, start 0, tilt 45
- 4) single, start -180, tilt 45

Come ci si aspettava la parte superiore dell'oggetto presenta dei piccoli buchi, per effetto dell'altezza come già spiegato. Tutto sommato però l'acquisizione è venuta discretamente.

Figura 2.28: Nano, risultato finale dell'aquisizione

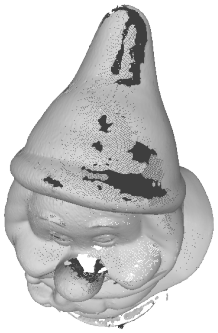

Figura 2.29: Particolare della parte superiore dell'oggetto

In figura 2.29 è possibile notare i buchi nella parte superiore della mesh acquisita e la grande densità di triangoli in quelle zone. Ciò è dovuto alle ripetute scansioni eseguite su quelle zone per cercare di riempire quei buchi acquisendo nuovi punti, purtroppo senza successo.

2.3 Post Processing

Successivamente alla fase di acquisizione c'è la fase di elaborazione delle scansioni. È possibile effettuare una molteplicità di operazioni, dall'allineamento alle varie tipologie di semplificazione. Di seguito presenteremo alcune di queste. Alcune operazioni lavorano soltanto con una singola scansione, o con la fusione di diverse scansioni in una mesh unica, quindi preliminarmente si è fatta una fusione delle scansioni per ottenere una mesh unica e poi si sono utilizzati i vari tool messi a disposizione dal software ScanStudio. Di seguito si mostreranno i risultati delle operazioni facendo riferimento ad un dettaglio della Palma, in fig. 2.30.

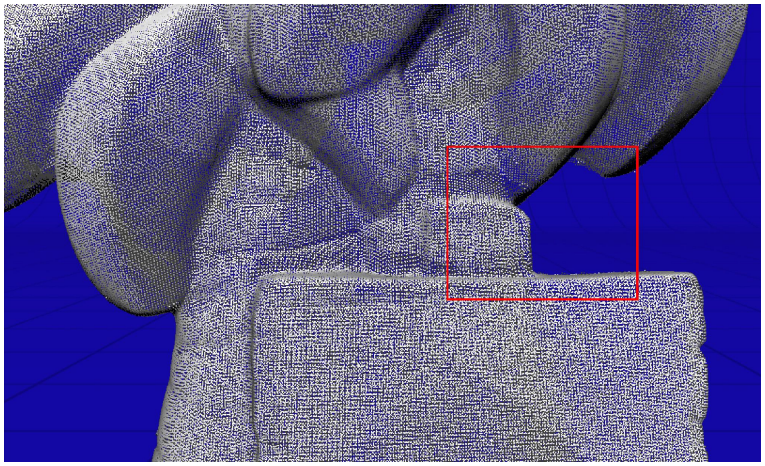

Figura 2.30: Vista della Palma in modalità wireframe

2.3.1 Modalità Fuse

Dal menù "Fuse⇒Fuse scans": attraverso questo strumento è possibile unire le varie scansioni in un'unica mesh. Questo tool fa il merge di diverse

scansioni, fa Hole Filling e semplifica il modello riducendo il numero di punti rispetto a quelli globalmente acquisiti e generando di conseguenza meno triangoli. È possibile specificare una tolleranza su questa semplificazione (distanza fra due punti), da un minimo di 0.00254 millimetri (0 nessuna semplificazione) ad un massimo di 5.08 millimetri. Non viene menzionato l'algoritmo usato nella semplificazione: si è riscontrato che si può passare da un modello dei dati di ingresso uniforme ad un modello scattered (punti densi solo dove necessario) oppure uniforme. Il software garantisce una semplificazione intelligente, lasciando un minor numero di punti nelle zone a bassa curvatura e molti più punti in aree dove invece la curvatura è più alta.

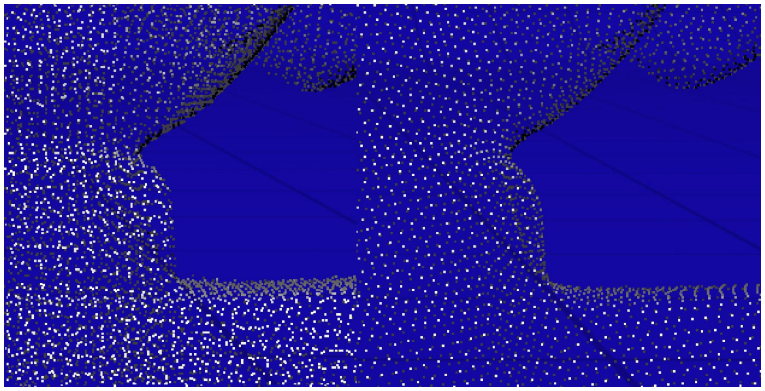

Figura 2.31: A sinistra il modello originale, senza semplificazioni e a destra la semplificazione con tolleranza 0.2

	Totale	Tollerance = 0,2	Tollerance = 0
Punti	97719	53289	62359
Triangoli	184306	85949	119823

Figura 2.32: Tabella riassuntiva con i valori relativi alla tolleranza massima e minima relative alla mesh Palma

Nella tabella 2.32 possiamo notare una significativa semplificazione: abbiamo utilizzato la massima tolleranza selezionabile, cioè cercando la massima semplificazione possibile e abbiamo ottenuto circa la metà dei punti e dei triangoli rispetto al modello iniziale. Selezionando la minima tolleranza

possibile il numero dei punti presi in considerazione aumenta insieme al numero di triangoli generati. Altrimenti è possibile impostare altri parametri per ottenere una singola mesh con o senza hole-filling (specificando ulteriori parametri sulle modalità di chiusura dei buchi, se flat, piuttosto che smooth Fuse⇒Fuse scans⇒Settings).

2.3.2 Modalità Regenerate Scans

In un qualsiasi momento è possibile rigenerare la nuvola di punti, partendo dalle scansioni effettuate inizialmente. Dal menù "Fuse⇒Re-generate Scan(s)" possiamo impostare una differente ricostruzione sui dati originali acquisiti. Possiamo scegliere di ottenere superfici più lisce, oppure possiamo decidere di tenere più o meno buchi, di generare più o meno triangoli. Facciamo notare che la rigenerazione in questo caso opera in maniera differente dall'operazione di Fuse: qui vengono rigenerati i punti di tutte le scansioni e selezionando la minima semplificazione è possibile ottenere una nuvola di punti più densa, mentre prima quello che succedeva era la fusione di diverse scansioni in una mesh unica. Partendo dall'acquisizione della palma mostriamo la rigenerazione delle nuvole di punti settando i parametri al minimo, quindi cercando la minima semplificazione possibile, cioè il massimo dettaglio: Semplification = 1.

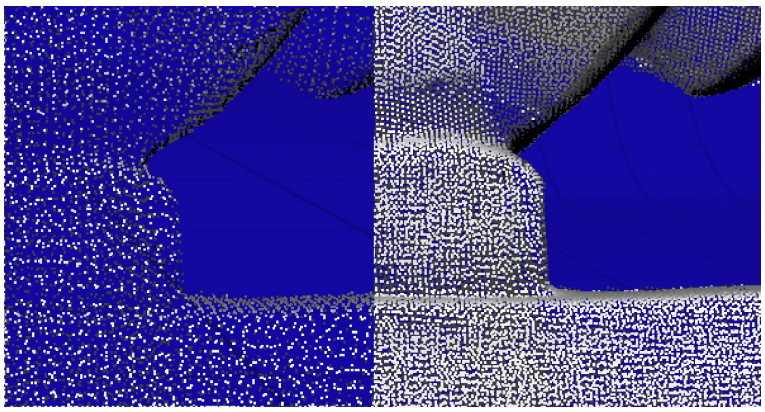

Figura 2.33: A sinistra nuvola di punti originale, a destra nuvola di punti rigenerata (S=1)

Con sorpresa il numero dei punti rigenerati riportati in tabella 2.35 è quintuplicato facendo crescere di un ordine di grandezza il numero dei triangoli creati. Si nota anche nella figura 2.33 la densità maggiore della nuvola di punti dopo la rigenerazione. Impostando tutti i parametri al massimo (Semplification = 9) cioè semplificando e riducendo il numero di punti e triangoli otteniamo la mesh di figura 2.34.

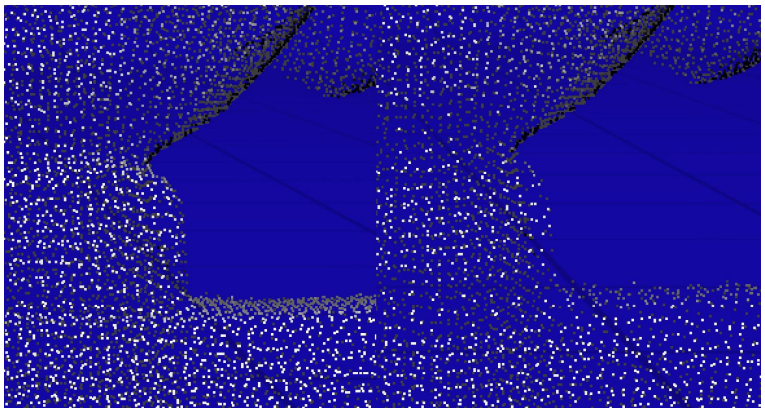

Figura 2.34: A sinistra nuvola di punti originale, a destra nuvola di punti rigenerata (S=9)

	Iniziale	Re-gen(s=1)	Re-gen (S=9)
Punti	97719	594359	55936
Triangoli	184306	1149122	104009

Figura 2.35: Tabella riassuntiva con i valori prima e dopo le due rigenerazioni

In tabella 2.35, si può notare che i punti generati sono dello stesso ordine di grandezza rispetto alla prima colonna, ma si vede abbastanza chiaramente in figura che la nuvola di punti ha una distribuzione differente dei punti sulla superficie. In prossimità di zone ad alta curvatura la nuvola di punti è meno densa. Supponiamo che questo strumento di semplificazioni utilizzi un metodo differente di semplificazione. Andando ad utilizzare il metodo Fuse su questa nuova nuvola di punti otteniamo i risultati di figura 2.36.

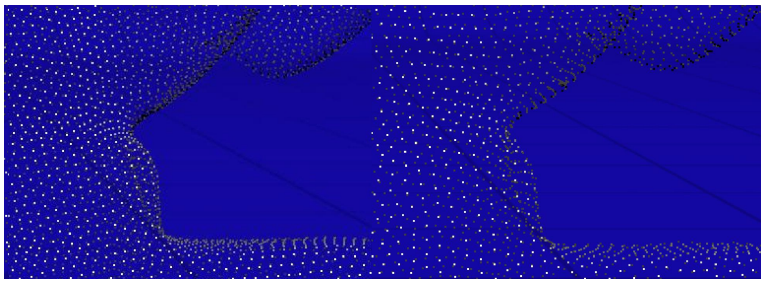

Figura 2.36: A sinistra nuvola di punti originale dopo Fuse scan (tollerance = 0,2); a destra nuvola di punti generata dopo strumento Re-generate Scan (semplification=9) e Fuse Scan (tollerance = 0,2)

	Totale	Tollerance 0,2	Totale	Tollerance = 0,2
Punti	97719	53289	55936	35973
Triangoli	184306	85949	104009	57242

Figura 2.37: Tabella riassuntiva con i valori trovati

In tabella 2.37 sono illustrati i risultati ottenuti applicando l'operazione di Fuse Scans ai modelli mostrati in fig. 2.36, cioè al modello originale e a quello rigenerato.

2.3.3 Modalità Trim

Lo strumento trim è da utilizzare quando nell'acquisizione di un oggetto ci sono parti superflue che possono essere eliminate dalla scansione, come per esempio la base o eventuali sostegni.

In figura 2.38 sono selezionate in rosso le parti acquisite in scansione che non fanno parte dell'oggetto e che possono essere eliminate senza perdita di informazione sul modello da acquisire. È possibile selezionare le parti da eliminare da una sola range image o dal modello intero.

2.3.4 Modalità Semplificazioni

Lo strumento di semplificazione permette di lavorare solo su una singola scansione o su un modello che è stato precedentemente fuso in un'unica mesh chiusa. Abbiamo tre principali funzioni:

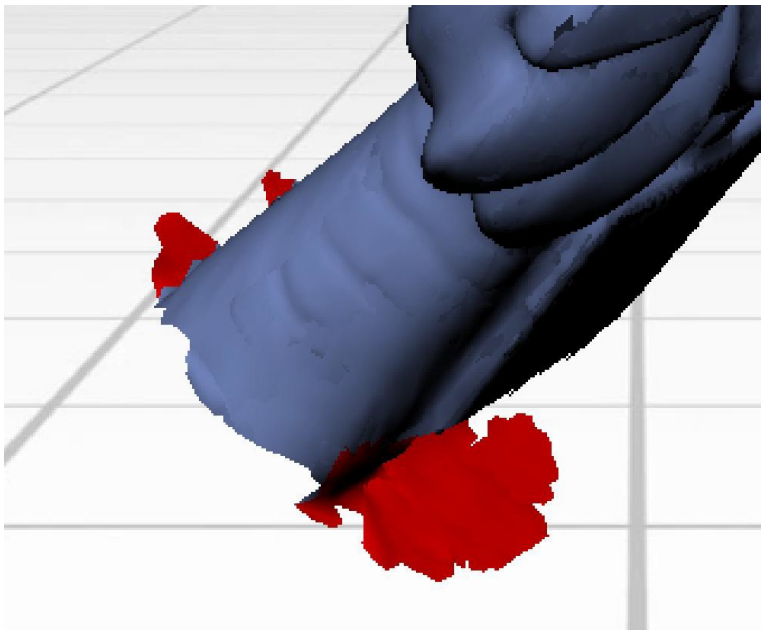

Figura 2.38: Dettaglio palma, in rosso parte da eliminare

- **Hole Filling**: Permette di riempire eventuali buchi che sono rimasti dopo aver fuso le diverse scansioni insieme. Selezionando le zone da chiudere possiamo specificare con quale modalita effettuare tale operazione:
  - Flat: riempie i buchi lungo gli spigoli creando una bordo piatto.
  - Smooth: riempie i buchi cercando di creare un bordo liscio.
  - Curvature: riempie i buchi analizzando la geometria vicina e cercando di far corrispondere la chiusura alla curvatura complessiva.

  Ci sono anche altri parametri da selezionare che permettono di specificare ulteriori criteri di chiusura. In maniera automatica l'hole filling è utilizzato anche con lo strumento di Fuse Scans.

- **Buffing**: Attraverso questo strumento e possibile *lucidare o lisciare* le superfici. L'acquisizione di un oggetto puo essere soggetta a rumore, è quindi possibile utilizzare questo strumento sull'intera mesh o su parti di essa per ridurre i disturbi.

- **Simplify**: Attraverso questo strumento e possibile semplificare il modello. Si lavora sempre con mesh chiuse, in cui e già stato fatto il merge delle scansioni. Questa operazione creerà una mesh non uniforme, come è evidente nelle figure sottostanti. Anche in questo caso è possibile specificare un grado di tolleranza: da un minimo di 0.00254 millimetri (0 nessuna semplificazione) ad un massimo di 5.08 millimetri.

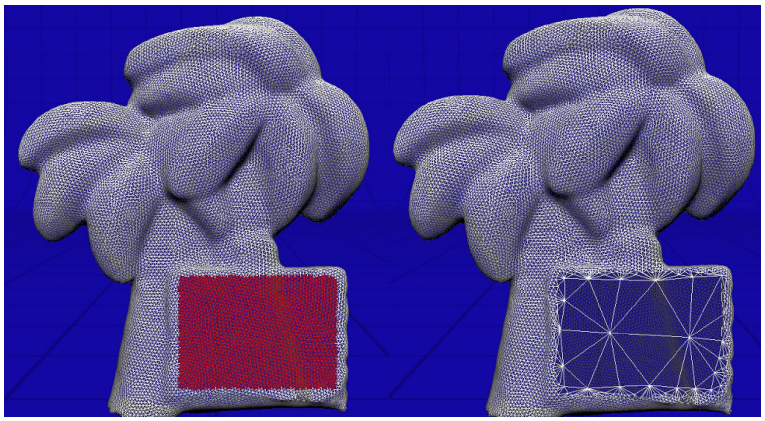

Figura 2.39: A destra Palma con selezione in rosso della parte da semplificare, a destra palma con semplificazione

	Totali	Fuse	Semplify 1
Punti	55936	35973	32910
Triangoli	104009	57242	54094

Figura 2.40: Tabella riassuntiva dei valori trovati

In tabella 2.40 sono mostrati i risultati dell'applicazione dell'operazione di simplify mostrata in fig. 2.39 sulla mesh *fused* ottenuta precedentemente.

In figura 2.39 in rosso compare la selezione della parte da semplificare, e a destra il risultato della semplificazione con tolleranza massima (cioè semplificando il più possibile). Essendo una superficie piana, dove quindi non c'è bisogno di un gran dettaglio, una semplificazione tanto significativa ha senso. Nella tabella 2.40 possiamo notare la

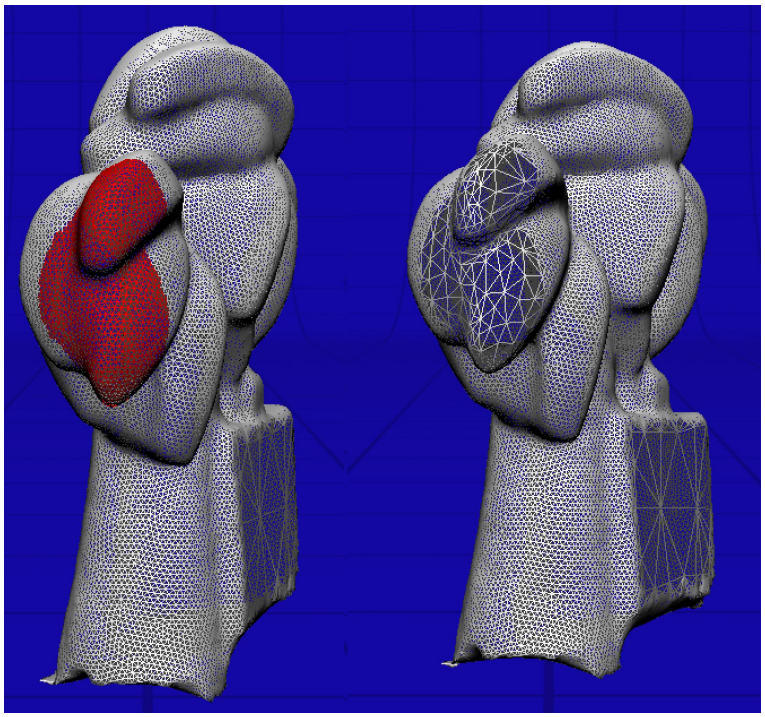

Figura 2.41: A sinistra selezione della parte da semplificare, a destra palma semplificata

	Totali	Fuse	Semplify 1	Semplify 2
Punti	55936	35973	32910	30360
Triangoli	104009	57242	54094	50926

Figura 2.42: Tabella riassuntiva con i valori trovati

diminuzione dei punti e dei triangoli generati. Utilizzando sempre la massima semplificazione selezionabile notiamo come per superfici curve (figura 2.42) si perdano informazioni significative, rendendo la geometria più spigolosa, anche se il software in maniera "intelligente" ha semplificato di meno nelle parti più ricche di dettagli. In vista puntiforme saltano all'occhio le semplificazioni effettuate.

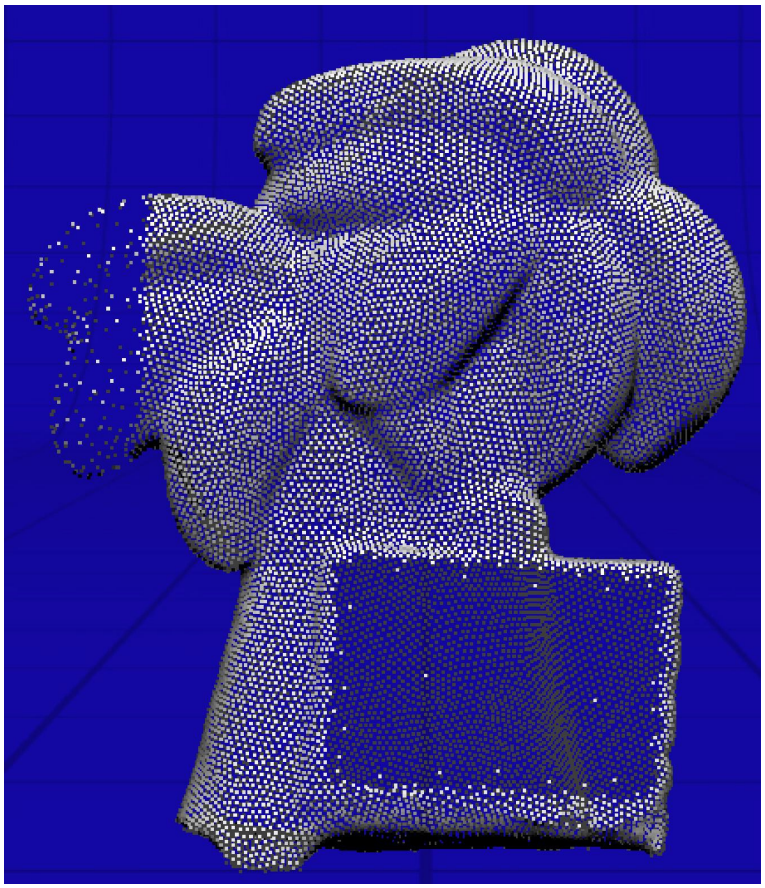

Figura 2.43: Palma in vista puntiforme dopo semplificazioni

2.3.5 Output

E possibile esportare le acquisizioni fatte e i modelli ricostruiti (sia come nuvole di punti sia in formato mesh) nei seguenti formati: h,ply,obj,stl,vrml,xyz. Si possono selezionare quali scansioni salvare, se una famiglia di scansioni o solo scansioni singole; salvare come un singolo file o in file separati in cui ad ogni scansione corrisponde un file di salvataggio. A seconda del formato può essere salvata anche l'informazione del colore: per esempio nel formato *.xyz (utilizzato per salvare le nuvole di punti) l'informazione sul colore non è mantenuta, mentre negli altri formati (quelli per salvare le mesh) sì. Inoltre l'informazione può essere salvata o all'interno del file (caso per esempio

del formato *.ply) oppure in un file esterno (caso *.obj oppure *.vrl) in cui c'è l'associazione delle immagini alla mesh.

2.4 Conclusione delle acquisizioni

L'algoritmo proposto da questa tesi lavora solo su mesh chiuse e *2-manifold*. In termini di post processing questo ha significato l'introduzione di una fase intermedia per fare dell'*hole filling*, ovvero riempire tutti quei buchi che si formano quasi inevitabilmente durante delle acquisizioni di questo tipo. Parte del problema deriva dal fatto che gli oggetti devono essere posizionati su un piedistallo che inevitabilmente viene scansionato insieme all'oggetto divenendo quindi parte della figura finale.
Questa parte, che chiaramente non interessa, va quindi rimossa con l'aiuto di software che permettano di fare delle operazioni di *trimming*, come descritto in precedenza. Ciò è stato fatto in parte con il software dello scanner NextEngine e in parte con **Meshlab**.
Anche quest'ultima operazione può creare degli hole; è quindi necessaria una fase ulteriore in cui si vanno a riempire questi buchi.
Inizialmente si è provato ad utilizzare anche in questo caso il software dello scanner e Meshlab, ma si sono dimostrati entrambi incapaci di chiudere gli hole più complicati. È stato quindi utilizzato il software **3Dreshaper** (`http://www.3dreshaper.com`) che però necessita di licenza.
Con l'aiuto di quest'ultimo software le mesh acquisite sono state chiuse incappando però in alcuni casi in errori di *non-manifoldness* su alcuni edge della mesh. L'unica soluzione in questi casi è stata intervenire direttamente *a mano* sulla mesh utilizzando il tool di modellazione 3D Blender (`http://www.blender.org`), che è in grado di evidenziare sulla mesh gli edge non-manifold e permette quindi di risolvere il problema.

Capitolo 3

Algoritmo proposto di semplificazione multilivello

L'algoritmo proposto in questo lavoro di tesi è composto in realtà da due passi fondamentali: uno di smoothing, l'altro di decimazione.
L'obiettivo è realizzare delle progressive mesh che mantengano, grazie alla fase di smoothing, un'apparenza più naturale e meno sfaccettata rispetto a come apparirebbero se fosse applicata la sola parte di decimazione.
Grazie a questo approccio questo algoritmo è in grado di offrire delle mesh semplificate di qualità migliore in un tempo di computazione relativamente breve. L'algoritmo è stato implementato all'interno del framework Meshviz, sviluppato all'interno dell'Università di Bologna per poter sviluppare software di modellazione grafica e trattamento di mesh poligonali. In termini di implementazione questo ha significato potersi avvalere di una infrastruttura per la gestione di mesh già pronta che utilizza al suo interno un sistema di riferimenti stile winged-edge, ottimale per navigare la mesh.
La prima parte dell'algoritmo, quella di smoothing, è stata sviluppata ed implementata dal co-relatore di questa tesi, l'Ing. Marco Rucci.
L'implementazione della seconda fase, quella di decimazione, è stato invece uno degli obiettivi di questa tesi.

3.1 Algoritmo

Le *multiresolution mesh* sono un ottimo metodo per rappresentare modelli grandi e complessi a differenti livelli di raffinamento. In un ambiente

di risoluzione multiresolution si ha la necessità di avere sia dei tool per rendere la mesh un po' più *grezza* sia di altri per raffinarla. L'approccio multilivello presentato in questa tesi semplifica mesh poligonali basandosi su un'*evoluzione delle superfici* ottenuta con una regolarizzazione p-Laplaciana.
Questa evoluzione può essere vista come un l'applicazione di un filtro geometrico ad una mesh iniziale ad alta risoluzione. Questo permette di rendere le superfici limite meno raffinate mantenendo al contempo i dettagli strutturali e permettendo così di raggiungere diversi livelli di semplificazione.
La semplificazione segue quindi un procedimento iterativo *a cascata* dove vengono applicati degli step di smoothing (basato su un'operatore p-Laplaciano pesato) e di decimazione.

ALGORITHM: MULTILEVEL MESH SEMPLIFICATION
INPUT: mesh iniziale X_0,% di edge da rimuovere PERC, # di iterazioni OUT_ITS
OUTPUT: mesh semplificata X1
E_0=lista edge della mesh
//Creazione istogramma per il calcolo della soglia di tolleranza TOLL sotto la quale applicare la decimazione
TOLL=Histogram(X_0,PERC)
//Calcolo curvature iniziali
H_0=computeCurvatures(X_0)
//Calcolo numero edge da rimuovere in base al totale di questi e alla percentuale in input
edgetocollapse=((PERC/100.0/3.0) * $|E_0|$)
$i = 0$
$X^{(i)} = X_0$
for(i=0;i<OUT_ITS;i++)
solve $X^{(i)}$ = Smoothing(X^{i-1},H_0)
solve $X^{(i)}$ = Decimation(X^i,E_0,H_0,TOLL,edgetocollapse)
OUT_ITS=OUT_ITS-1
endfor
$X_1 = X_{temp}$

L'algoritmo di semplificazione proposto è in grado di semplificare la mesh

della quantità voluta a partire da tre parametri in ingresso: la mesh originale, la percentuale di edge sul totale che si vogliono rimuovere e il numero di iterazioni, ovvero di cicli, da far effettuare all'algoritmo.
Nella fase preliminare, l'algoritmo è infatti in grado, a partire dalla percentuale data in input, di calcolarsi automaticamente che valore di tolleranza deve utilizzare durante la decimazione. Questo risultato viene raggiunto tramite il calcolo di un istogramma che raggruppa tutti gli edge facenti parte della mesh, ordinandoli in determinati *bin* (insiemi) in base alla loro lunghezza normalizzata.

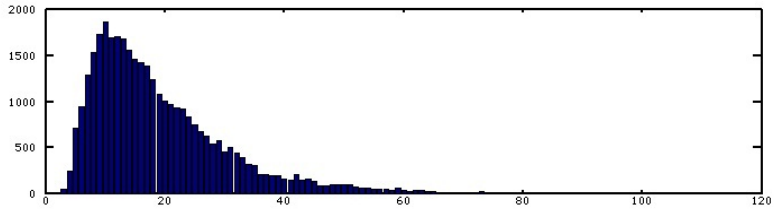

Figura 3.1: Istogramma creato per la mesh Mucca, composta da 23216 facce e 34824 edge

In questo modo si è in grado di calcolare, a partire dalla percentuale specificata in input, quanti sono gli edge che andranno rimossi e di conseguenza qual è il valore di tolleranza (lunghezza degli edge) sotto il quale devo applicare la decimazione.
La difficoltà in questa parte dell'algoritmo è relativa al valore di *risoluzione* dell'istogramma, ovvero in quanti bin dividere l'istogramma. È immediatamente comprensibile infatti come questo valore sia importantissimo per effettuare misurazioni accurate ed evitare di semplificare troppo o troppo poco. Sperimentalmente si è valutato che per una mesh poligonale, irregolare e di media risoluzione 100 bin è un valore sufficiente.
Vengono inoltre calcolate le curvature iniziali dei singoli vertici (H_0) che verranno poi preservate dall'algoritmo ad ogni iterazione. In questo modo vengono risolti molti problemi numerici dovuti proprio al calcolo delle curvature (sensibili alla presenza di triangoli degeneri e "situazioni strane"), migliorando la qualità del risultato finale.

L'algoritmo esegue un numero di iterazioni richieste dall'utente, la mesh finale prodotta è una sola, le mesh intermedie prodotte alla fine di ciascun ciclo vengono salvate in file temporanei e non vengono fornite in output. È possibile modificare il codice affinchè anch'esse vengano prodotte alla fine dell'esecuzione dell'algoritmo.
Oltre ai tre parametri di input principali già spiegati ce ne sono altri *opzionali*, in quanto servono solo per impostare da riga di comando dei valori che normalmente sono già impostati di *default*.
Il seguente esempio mostra una possibile linea di comando per avviare l'applicazione:

./simplify –in *mesh.obj* –percentage *N* –iterations *K* –smoothing-iterations *SI* –lambda *FLOAT* –out *meshsimplified.obj*

Ecco una lista di tutti i possibili comandi:

```
"parameters:\n"
" --in mesh.obj\n"
"    Specify the input mesh in obj format.\n"
"\n"
" --percentage N\n"
"    The percentage of edges to remove.\n"
"\n"
" --iterations K\n"
"    The number of \"outer\" iterations of the simplification
     algorithms.\n"
"\n"
" --ple FLOAT\n"
"    The value of the p-laplacian exponent.\n"
"\n"
" --lambda FLOAT\n"
"    Defines the fidelity parameter lambda in the solution of
     the smoothing model.\n"
"\n"
" --no-adaptive\n"
"    Disable the curvature adaptivity of the fidelity parameter
     lambda in the solution of the smoothing model.\n"
"\n"
```

```
" --no-sort\n"
"    Disable sorting of edges in decimation.  Enable threshold-based
     decimation.\n"
"\n"
" --no-smoothing\n"
"    Do not perform the smoothing step in each simplification
     iteration.\n"
"\n"
" --no-decimation\n"
"    Do not perform the decimation step in each simplification
     iteration.\n"
"\n"
" --out FILE\n"
"    Secify the output mesh file.\n"
"\n"
"";
```

3.2 Smoothing

Durante la fase di smoothing viene applicato sulla mesh l'operatore di discretizzazione p-Laplaciano definito nell'articolo [9]. Viene sfruttato il fatto che questo operatore ha comportamenti diversi a seconda dell'esponente p fornito. Infatti se $p \geq 1$ allora si comporta come un normale operatore per lo smoothing (con $p = 2$ Δ_p è il classico Laplaciano già descritto in precedenza), se invece $p < 1$ tende ad accorpare i vertici verso le zone ad alta curvatura, ed è proprio questo il caso sfruttato in questa algoritmo. Dunque non parliamo di uno smoothing vero e proprio ma il meccanismo alla base è comunque lo stesso.
Per ogni iterazione dell'algoritmo di semplificazione, l'algoritmo di smoothing esegue ogni volta un proprio numero di iterazioni n (di default questo valore è 10 ma può essere specificato da riga di comando) applicando n volte l'operatore p-Laplaciano alla mesh. Lo smoothing si basa sulla risoluzione della equazione differenziale di diffusione alle derivate parziali:

$$X \quad : \quad \frac{\partial x}{\partial t} = \Delta_p X$$

Dove X rappresenta le coordinate dei vertici della mesh e, se X è una funzione continua $\in C^2$, allora l'operatore p-Laplaciano è definito come $\Delta_p X = \nabla(|\nabla X|\nabla X))$.
Discretizzando in modo esplicito l'equazione si ha:

$$X^{n+1} = X^n + \Delta_t(\Delta_p X^n)$$

Dove Δ_t è il passo temporale, Δ_p è discretizzato nello spazio tramite una matrice L_p, X^n è il vettore delle coordinate dei vertici della mesh.

SMOOTHING
INPUT: mesh X^n,
OUTPUT: mesh smoothed X_{tmp}
sm = iterazioni$_s$*moothing*
$for(i = 0, i < sm, i++)$
calcolo L_p
$X^{n+1} = X^n + \Delta_t(L_p X^n)$
endfor
$X_{tmp} = X^{n+sm}$

In figura 3.2 si possono constatare gli effetti sulla mesh della fase di smoothing. Nell'ingrandimento della zona evidenziata si nota come i vertici siano stati raggruppati nelle zone a più alta curvatura che appaiono infatti più dense che nell'originale.

3.3 Decimazione

Una volta calcolata la tolleranza ed effettuata la fase di smoothing la decimazione viene avviata. Questa fase può essere affrontata in due modi. L'approccio di *default* prevede che venga eseguito un *sorting* (ordinamento) degli edge per dare la priorità a quelli con minor lunghezza e minor curvatura. In questo modo la decimazione viene eseguita gradualmente basandosi sul numero di edge totali da rimuovere e sul numero di iterazioni richieste. È possibile disabilitare il sorting da riga di comando (basta aggiungere agli input il comando –no-sort), se ciò viene fatto ad ogni iterazione dell'algoritmo vengono presi in esame tutti gli edge della mesh controllando volta per

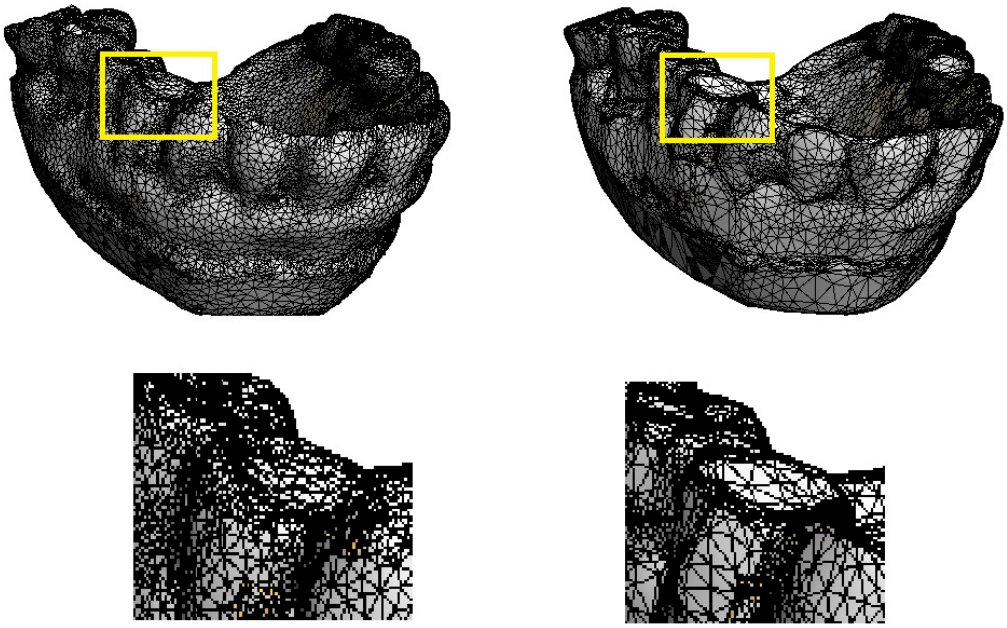

Figura 3.2: Effetti della fase di smoothing sulla mesh

volta la tolleranza.

DECIMATION
INPUT: mesh prodotta dalla fase di smoothing X_{tmp}, lista ordinata edges E_0, tolleranza TOLL
OUTPUT: mesh semplificata X_{tmp}
if (sorting è abilitato)
ne = edgetoCollapse / OUT_ITS
E_{tmp}=sorting(E_0)
for(i=0,i<ne,i++)
edgecollapse($E_{tmp}(i)$)
aggiornamento edges lenght
E_{tmp}=sorting(E_{tmp})
endfor
endif
else
for(i=0,i< $|E_0|$,i++)
if (lunghezza $E_0(i)$ < TOLL)
edgecollapse($E_0(i)$)
endif
endfor
endelse

L'operazione di edge collapse, come già spiegato in precedenza, collassa un lato in un unico vertice. Questo significa che nel collasso, oltre all'edge, vengono coinvolti i due vertici, le due facce incidenti e i restanti quattro edge che delimitano le due facce. L'ordine in cui si va ad agire su tali fattori è importantissimo in quanto la cosa più importante per evitare errori di topologia è aggiornare in modo corretto i riferimenti degli "attori" coivolti, specialmente i vertici e gli edge rimanenti.
Inoltre dato che l'edge collapse va anche a modificare le lunghezze di tutti gli edge collegati ai suoi due vertici estremi anche la lista degli edge va riordinata.
Un altro fattore importante riguardo la consistenza della mesh è l'attenzione per evitare situazioni potenzialmente non corrette. L'algoritmo non applica edge collapse ai seguenti due casi:

(a) uno o entrambi i vertici opposto all'edge da collassare hanno valenza

uguale a tre

(b) oltre alle due coppie di edge delimitanti le due facce incidenti, esiste anche un'altra coppia di edge che collegati sia tra loro che ai due vertici rispettivamente

Nel caso (a), che possiamo vedere in figura 3.3, vediamo che a seguito del collasso dell'edge in questione, se il vertice opposto ha valenza uguale a tre, si viene a creare una situazione topologica non corretta.

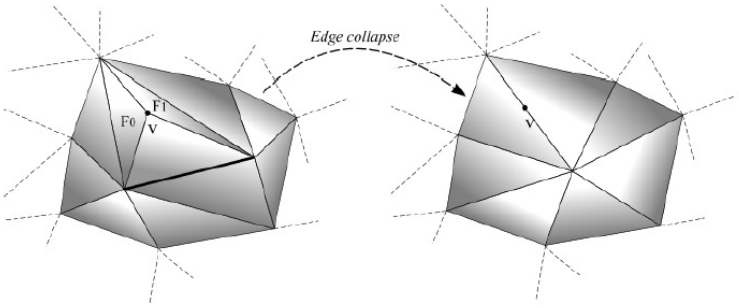

Figura 3.3: Caso **(a)** di situazione non corretta riscontrato

In questo caso non potendo eseguire l'edge collapse, l'algoritmo applica altre tecniche per eliminare il problema andando ad agire sul vertice che crea problemi.
Nel caso (b) invece la situazione è assimilabile a quella illustrata in figura 3.4, se collassiamo uno qualsiasi dei lati (edge) della base andremmo a sovrapporre perlomeno entrambi gli altri due lati della base. Questa situazione oltre a non essere topologicamente corretta va a creare anche dei problemi in fase di salvataggio e riutilizzo della mesh. Infatti se si lavora, come in questo caso, su file di formato .obj, il salvataggio andrà a creare delle situazioni di *non-manifold*, in quanto i file .obj tengono conto solo dei riferimenti dei vertici e delle facce della mesh. In questo modo, quando la mesh verrà nuovamente caricata si andranno a creare delle situazioni in cui esistono dei lati/edge in comune tra più di due facce, come nel caso illustrato dalla figura 3.5.

Quando l'algoritmo di edge collapse viene messo in esecuzione controlla subito di non essere in una situazione potenzialmente non corretta come quelle descritte sopra, e se non ci sono problemi, procede con il collasso.

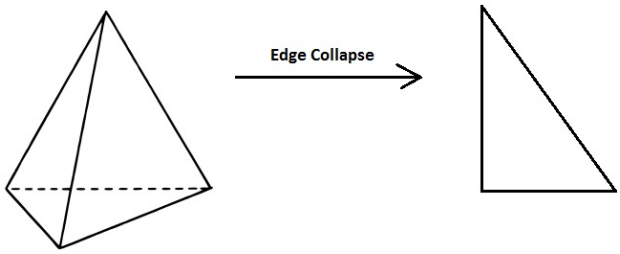

Figura 3.4: Caso **(b)** di situazione non corretta riscontrato

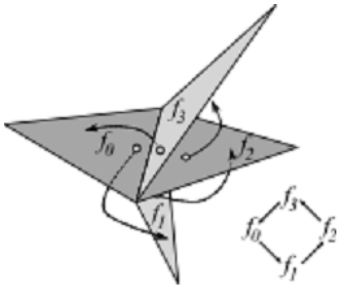

Figura 3.5: Caso di lato non manifold

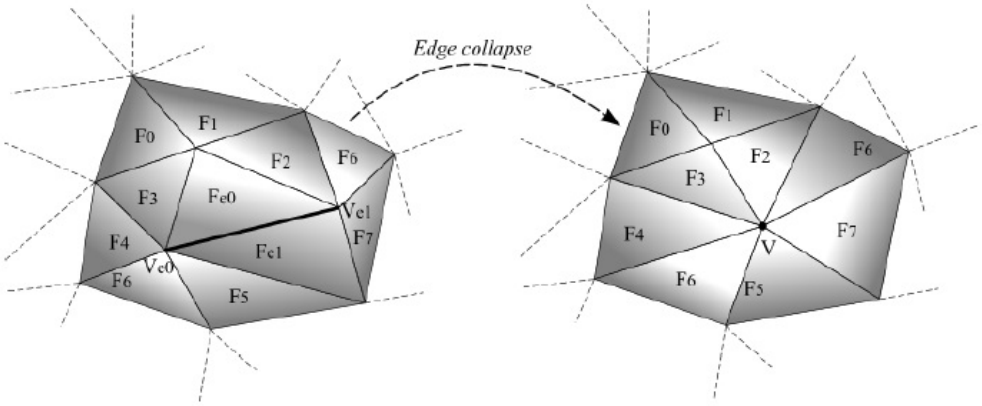

Figura 3.6: Edge Collapse

```
EDGE COLLAPSE
INPUT: edge da collassare EDGE
if (possibile situazione non corretta)
  return
endif
else
  EDGE vertices = ve0,ve1
  EDGE faces = fe0,fe1
  EDGE1,EDGE2 = edges collegati a EDGE tramite ve0
  EDGE3,EDGE4 = edges collegati a EDGE tramite ve1
  cv0=computeCurvature(ve0)
  cv1=computeCurvature(ve1)

//cancello EDGE
  edgeDelete(EDGE)

//aggiusto i riferimenti degli elementi coinvolti

//cancello le facce
faceDelete(fe0);
faceDelete(fe1);

//cancello i due vertici che erano uniti a EDGE tramite ve1
edgeDelete(EDGE3);
edgeDelete(EDGE4);

//aggiusto i riferimenti degli elementi coinvolti

//trovo il giusto valore di interpolazione tenendo conto delle due cu
//calcolate inizialmente, e trovo la nuova posizione interpolando
t = ((cv1-cv0)*0.5)+0.5;
nuovo_vertice=lerp(ve0,ve1,t)

//cancello ve1
vertexDelete(ve1)
endelse
return
```

I due vertici collegati dall'edge in questione sono stati nominati ve0 e ve1, mentre le facce incidenti sono fe0 e fe1. Per comodità e anche per evitare eventuali errori nel trasferimento dei riferimenti è stata fatta la scelta implementativa di mantenere nella struttura della mesh il vertice ve0 e i due lati relativi a fe0 e fe1 a lui collegati, anche dopo l'esecuzione dell'edge collapse. Aggiustando infatti i riferimenti di questi elementi per adeguarli alla nuova struttura della mesh, non vi è stato bisogno di creare nuovi elementi che sostituissero quelli eliminati durante il collasso. L'algoritmo fa in modo che i riferimenti delle facce incidenti relativi ai due edge collegati a ve1 (quindi da cancellare) vengano passati ai due collegati a ve0.
Si agisce anche sui riferimenti relativi al vertice ve1 che vengono tutti passati al vertice ve0. Inoltre a ve0 vengono assegnate delle nuove coordinate calcolate basandosi sulla curvatura normalizzata dei due vertici. In questo modo ve0 viene spostato in base all'attrazione esercitata dai valori di curvatura dei due edge (in questo modo il nuovo vertice giace sul segmento tra i due vecchi vertici). Possono essere usati due tipi di curvatura, la curvatura media e la curvatura Gaussiana. L'algoritmo funziona bene con entrambe, ma è consigliato l'uso della prima in quanto esistono dei casi in cui utilizzando la Gaussiana il calcola del valore di interpolazione t non porta ad una situazione ottimale.
Sono comunque dei casi limite, nei quali la mancanza di continuità in una delle direzioni tangenti fa sì che la curvatura Gaussiana sia nulla.

Capitolo 4

Sperimentazione

Una volta completato l'algoritmo di semplificazione sono stati effettuati molteplici test su mesh di varia natura. L'obiettivo è stato comprendere quali combinazioni dei parametri in ingresso permettono di avere la migliore qualità nei risultati. Per la sperimentazione sono state utilizzate anche mesh molto irregolari e ad alta risoluzione per testare l'algoritmo nei casi più complessi. Le mesh utilizzate nei test sono mostrate in figura 4.1, hanno tutte più di 20000 facce (in alcuni casi anche più di 100000).

4.1 Misure di Valutazione

I risultati dell'esecuzione dell'algoritmo sulle mesh sono stati valutati in due modi:

- **Qualitativamente**: valutazione visiva della bontà dei modelli ottenuti in base a come l'algoritmo si è comportato in casi in cui il mantenimento di certe caratteristiche del modello era obbligatorio.

- **Quantitativamente**: valutazione della soluzione ottenuta tramite l'algoritmo, utilizzando come parametro di valutazione la distanza di Hausdorff.

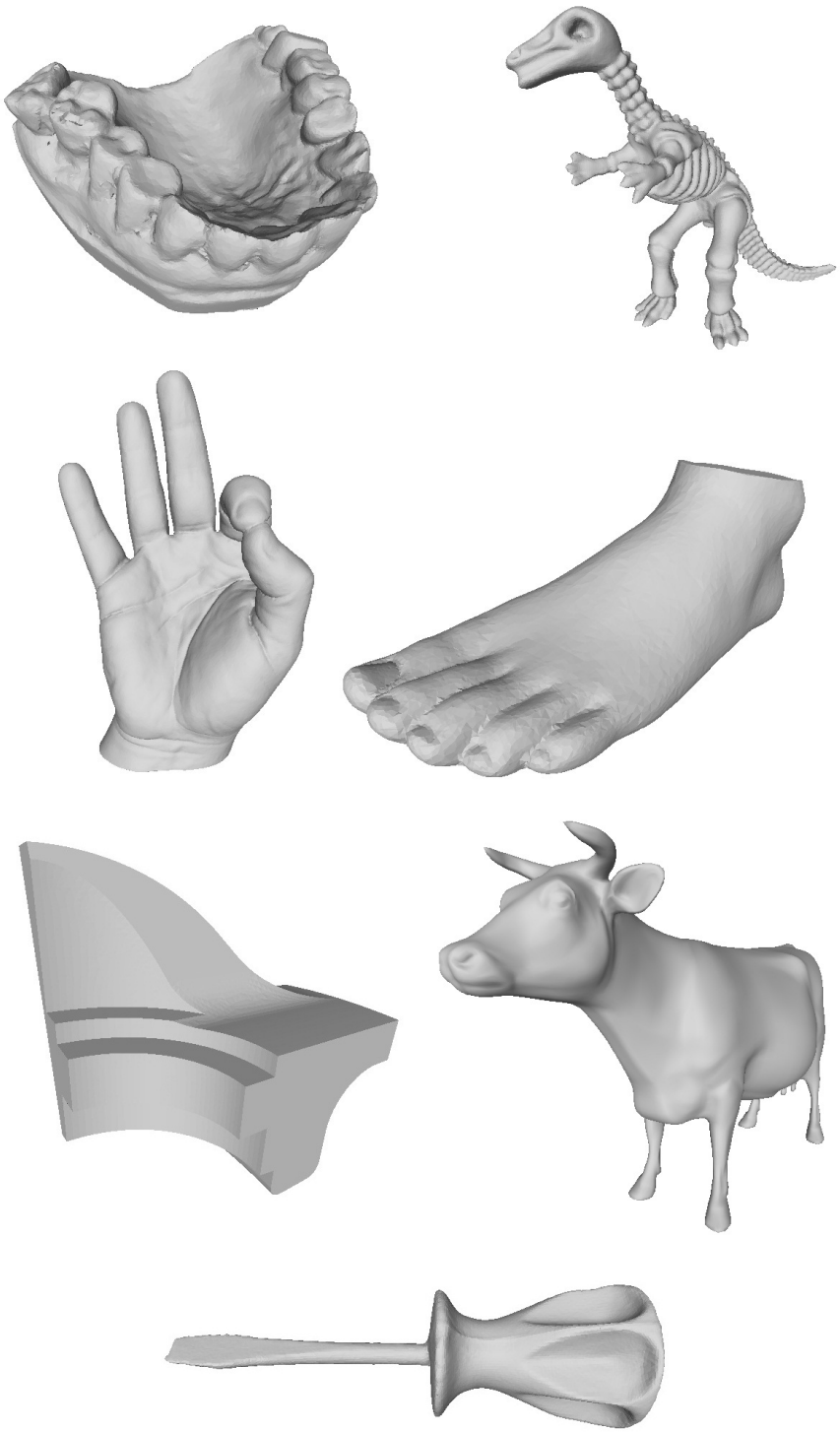

Figura 4.1: Mesh utilizzate durante la fase di sperimentazione

4.2 Esempio 1

In questo esempio vengono mostrati i risultati dell'esecuzione dell'algoritmo sulle mesh visualizzate in fig. 4.1. Ecco il risultato dell'esecuzione dell'algoritmo sulle mesh utilizzate per la sperimentazione.
Tutte le semplificazioni sono state eseguite con i seguenti parametri (eccetto dove indicato diversamente):

- Numero di iterazioni totali 4 (nell'algoritmo Simplify, capitolo 3, sono indicate con OUT_ITS)
- Numero di iterazioni di smoothing 3 (nell'algoritmo Smoothing, capitolo 3, sono indicate con sm)
- Parametro λ 10 (nell'algoritmo Simplify, capitolo 3, indicata come lambda)

I parametri non indicati sono stati utilizzati con il valore di default.

Questi parametri si sono rivelati sperimentalmente i migliori nella quasi totalità dei casi.
È importante notare la diretta correlazione che c'è tra il numero degli edge totali che compongono la mesh e la percentuale di semplificazione massima raggiunta prima che la qualità del modello si deteriorasse troppo. Questo dipende dal fatto che in mesh con 100000 edge o superiori il numero di edge che non superano il valore di tolleranza e la cui eliminazione non comporta un deterioramento della qualità saranno molti di più rispetto ad una mesh con valori inferiori. Minore è il numero di edge dell'oggetto, più piccola sarà la percentuale massima di semplificazione che potremo utilizzare senza deteriorare troppo la mesh iniziale.

4.2.1 Dentiera

L'oggetto Dentiera è composto da 52122 facce per un totale di 78183 edge. In figura 4.2 l'oggetto viene mostrato a diversi livelli di semplificazione:

a Originale

b Semplificazione al 50%: 26048 facce, 39072 edge.

c Semplificazione al 70%: 15628 facce, 23442 edge.

d Semplificazione al 90%: 5204 facce, 7806 edge.

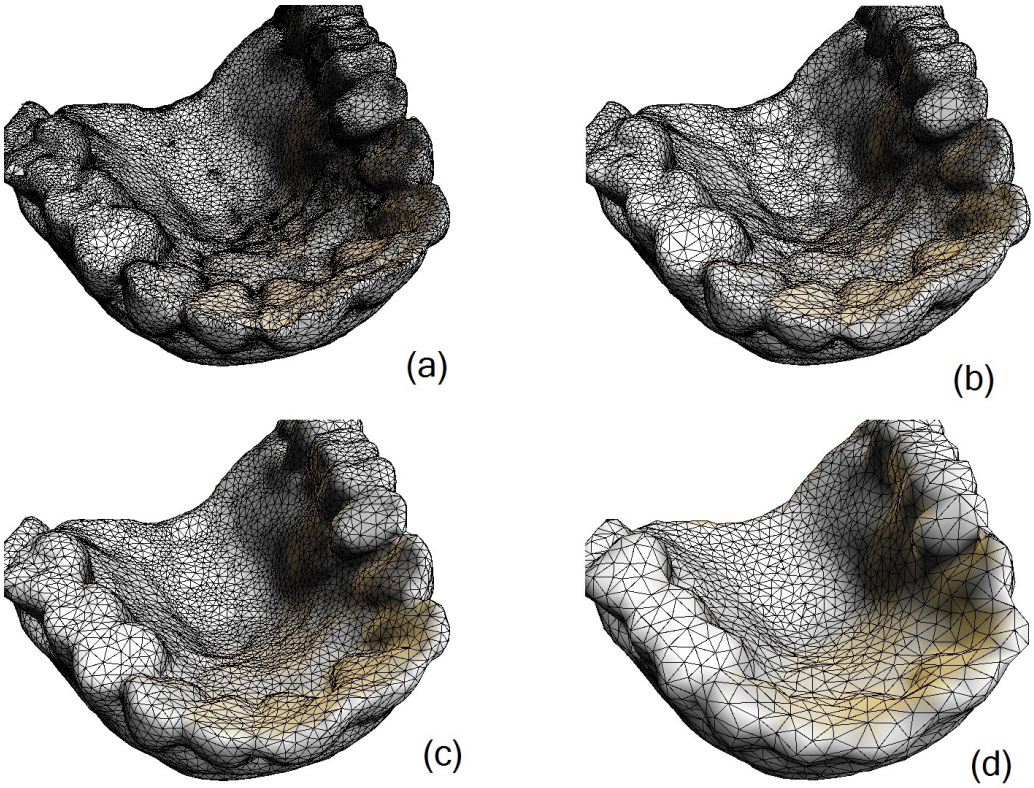

Figura 4.2: Effetti dell'applicazione dell'algoritmo alla mesh dentiera con diverse percentuali di semplificazione

In figura 4.3 particolare degli incisivi con le stesse percentuali di semplificazione.

4.2.2 Piede

L'oggetto Piede è l'oggetto più piccolo tra quelli testati, è infatti composto da 20476 facce e 30714 edge.
Anche in questo caso l'oggetto viene mostrato a diverse percentuali di semplificazione. In figura 4.4abbiamo:

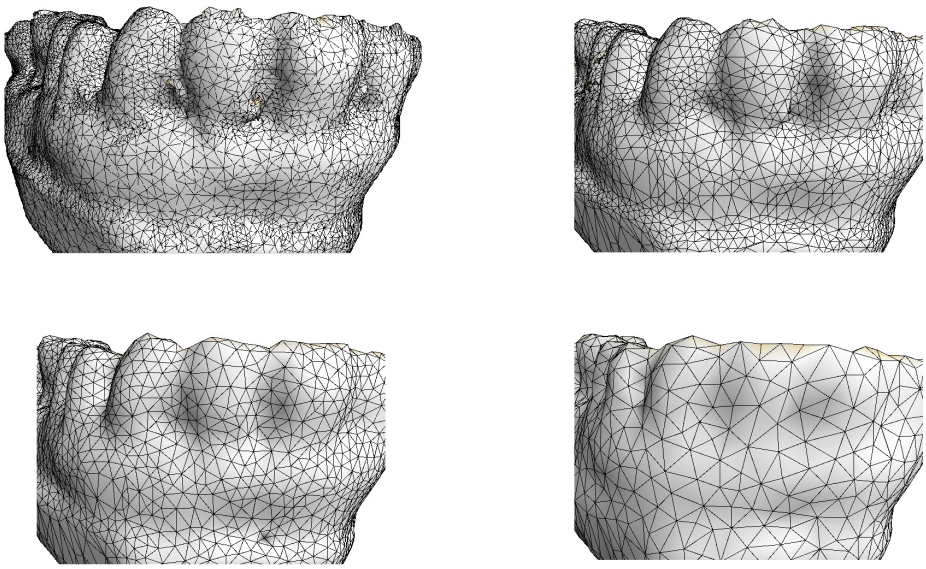

Figura 4.3: Effetti dell'applicazione dell'algoritmo alla mesh dentiera da una diversa prospettiva

a Originale

b Semplificazione al 30%: 14332 facce, 21498 edge.

c Semplificazione al 50%: 10236 facce, 15354 edge.

d Semplificazione al 70%: 6140 facce, 9210 edge.

In questo caso le semplificazioni a percentuali maggiori del 70% non sono state in grado di mantenere una qualità accettabile nell'oggetto.

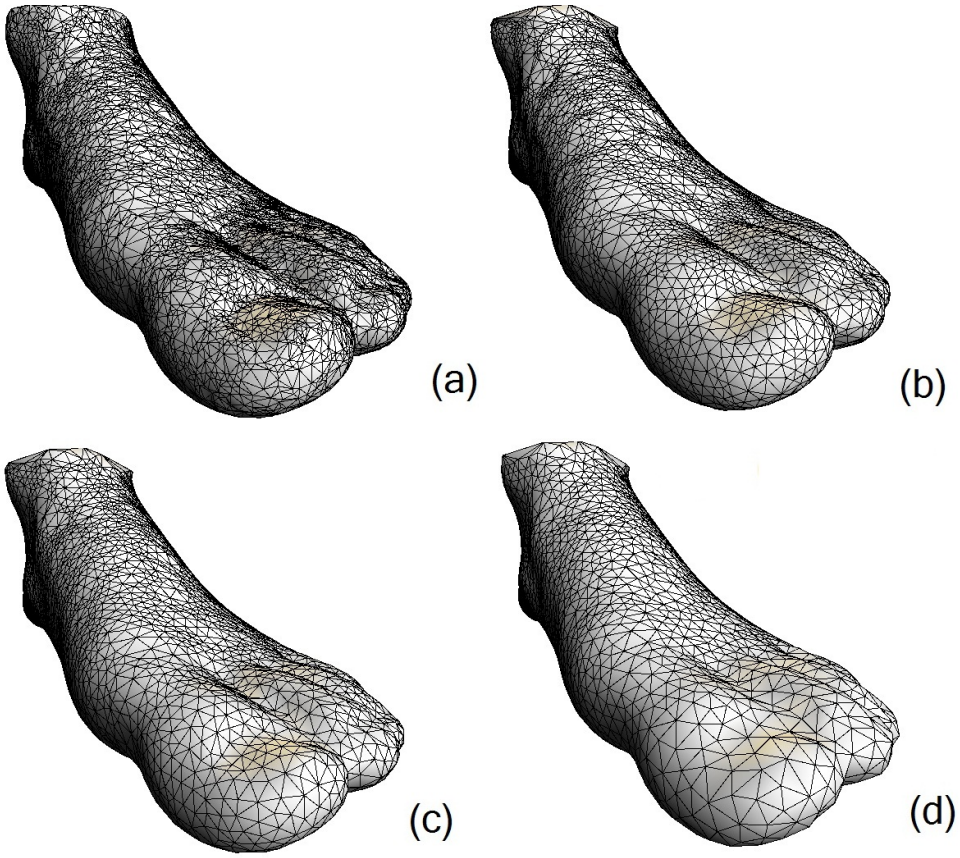

Figura 4.4: Effetti dell'applicazione dell'algoritmo alla mesh Piede con diverse percentuali di semplificazione

4.2.3 Cacciavite

L'oggetto Cacciavite è costituito da 54300 facce e 81450 edge.
La particolarità di questo oggetto è l'impugnatura che presenta zone a curvatura abbastanza elevata. In figura 4.5 sono mostrate le varie semplificazioni applicate all'oggetto:

a Originale

b Semplificazione al 50%: 27148 facce, 40722 edge.

c Semplificazione al 70%: 16284 facce, 24426 edge.

d Semplificazione al 90%: 5428 facce, 8142 edge.

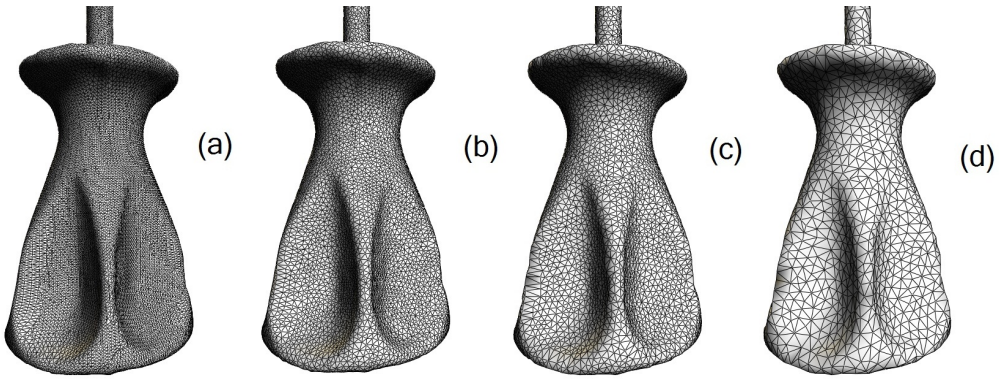

Figura 4.5: Effetti dell'applicazione dell'algoritmo alla mesh Cacciavite con diverse percentuali di semplificazione

4.2.4 Mano

L'oggetto Mano è uno dei più voluminosi, ha infatti 101446 facce e 152169 edge.
Anche questo presenta molte zone ad alta curvatura, specialmente nel palmo è importante riuscire a mantenere le *pieghe*. In figura 4.6 sono mostrate le varie semplificazioni applicate all'oggetto:

a Originale

b Semplificazione al 50%: 50718 facce, 76077 edge.

c Semplificazione al 70%: 30430 facce, 45645 edge.

d Semplificazione al 90%: 10142 facce, 15213 edge.

e Semplificazione al 95%: 5070 facce, 7605 edge.

In questo caso, dato l'alto numero di edge, siamo stati in grado di effettuare semplificazioni fino al 95% mantenendo un ottima qualità, come mostrato anche in figura 4.7.

In figura 4.8 viene mostrata la mesh Mano dopo una semplificazione del 99.5% (502 facce,251 vertici e 753 edge).

4.2.5 Fandisk

L'oggetto Fandisk è molto interessante in quanto presenta delle caratteristiche strutturali molto interessanti per testare l'algoritmo. È necessario infatti preservare gli spigoli netti che l'oggetti presenta. Questo modello è costituito da 51784 facce e 77676 edge.
In figura 4.9 sono mostrate le varie semplificazioni applicate all'oggetto:

a Originale

b Semplificazione al 50%: 25888 facce, 38832 edge.

c Semplificazione al 70%: 15528 facce, 23292 edge.

d Semplificazione al 90%: 5176 facce, 7764 edge.

e Semplificazione al 95%: 2584 facce, 3876 edge.

f Semplificazione al 98%: 1036 facce, 1554 edge.

La semplificazione al 98% è stata eseguita cercando di *stressare* l'algoritmo ed è mostrata in fig. 4.10.

Come si può facilmente notare a 98% la qualità del modello è molto degradata.

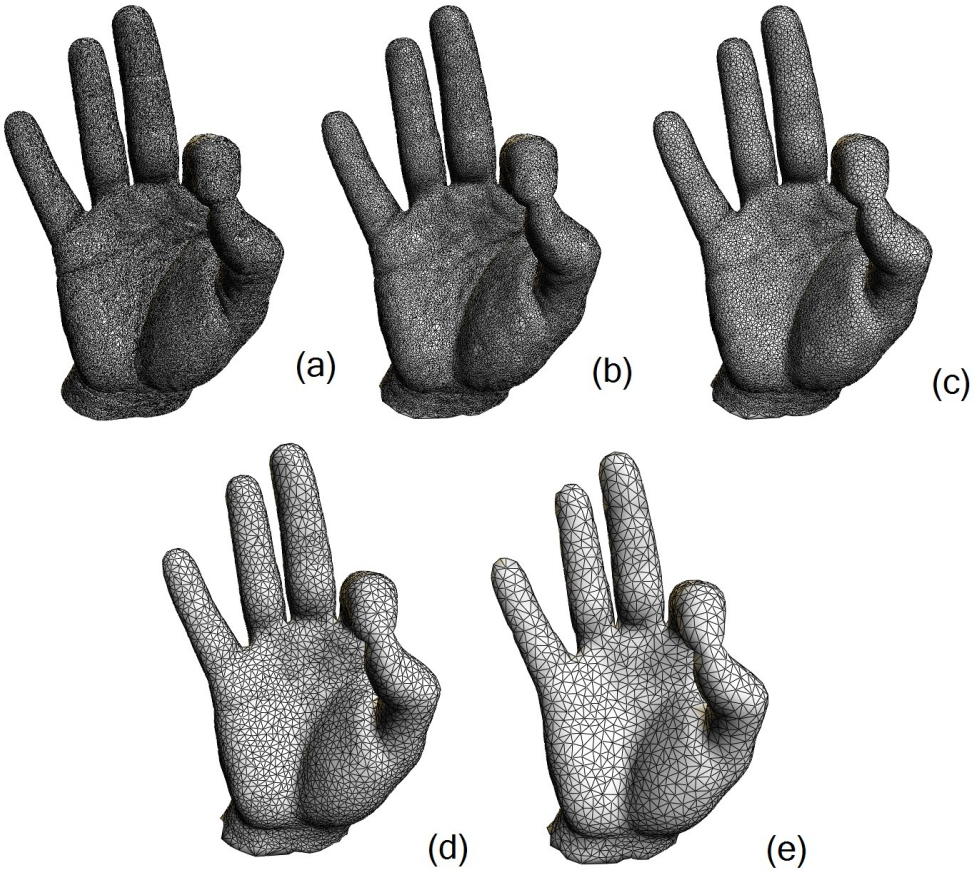

Figura 4.6: Effetti dell'applicazione dell'algoritmo alla mesh Mano con diverse percentuali di semplificazione

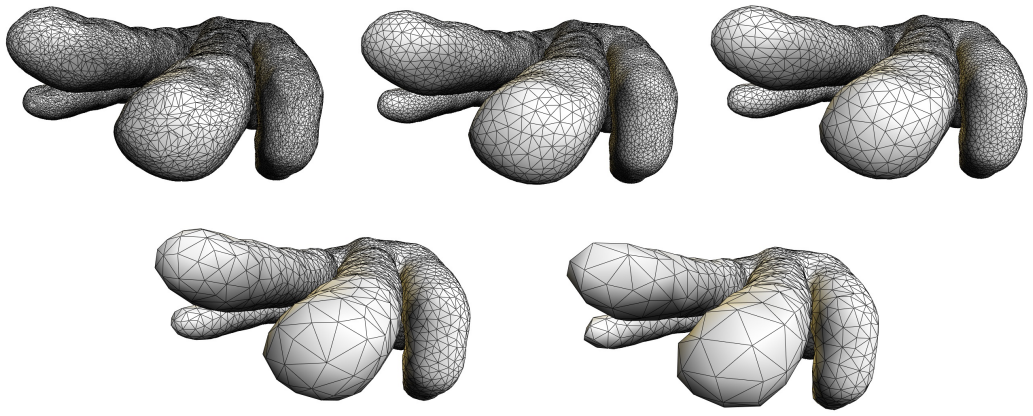

Figura 4.7: Effetti dell'applicazione dell'algoritmo alla mesh Mano (dettaglio delle dita)

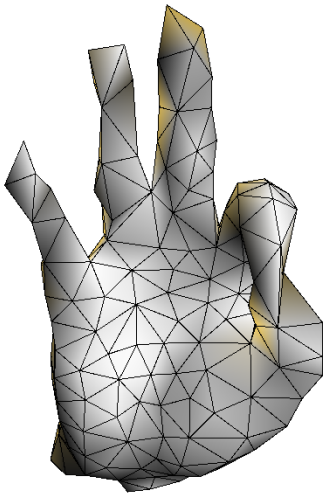

Figura 4.8: Effetti dell'applicazione dell'algoritmo alla mesh Mano con percentuale di semplificazione del 99.5%

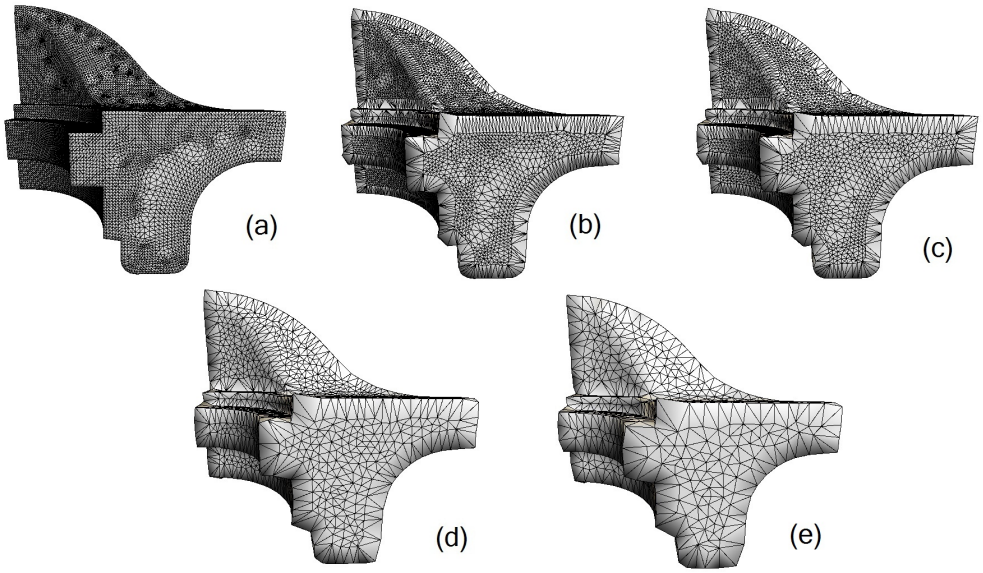

Figura 4.9: Effetti dell'applicazione dell'algoritmo alla mesh Fandisk con diverse percentuali di semplificazione

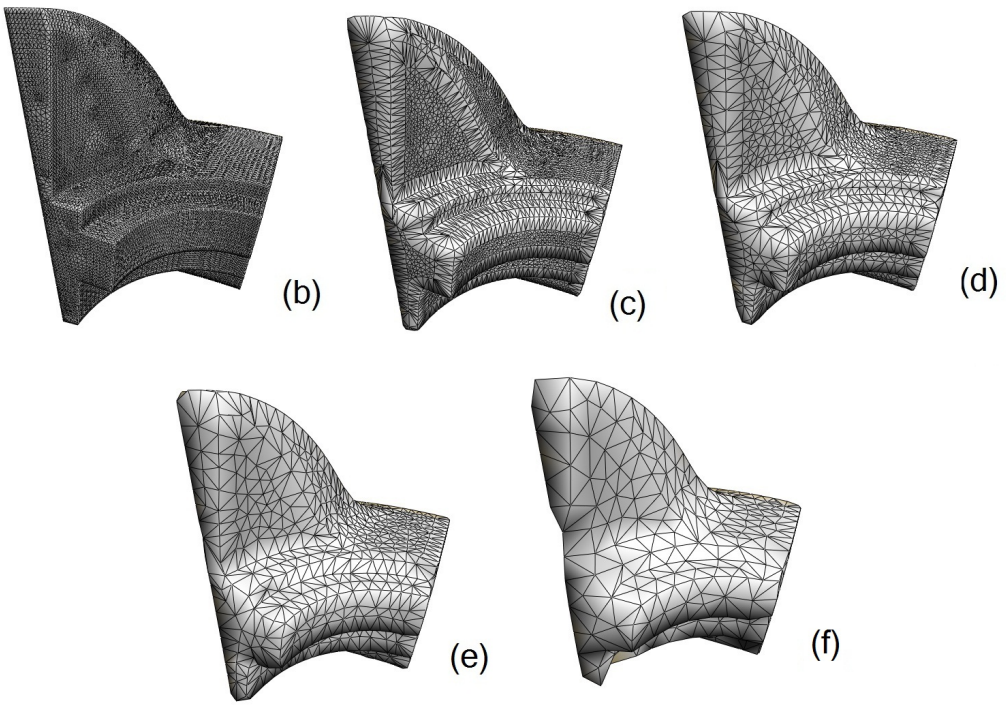

Figura 4.10: Effetti dell'applicazione dell'algoritmo alla mesh Fandisk con diverse percentuali di semplificazione e da un'altra visuale

4.2.6 Mucca

L'oggetto Mucca è costituito da 23216 facce e 34824 edge.
Il dettaglio più interessante sono senza dubbio le corna, aree ad altissima curvatura.
In figura 4.11 vengono mostrate le varie semplificazioni applicate all'oggetto:

a Originale

b Semplificazione al 50%: 11600 facce, 17400 edge.

c Semplificazione al 70%: 6960 facce, 10440 edge.

d Semplificazione al 90%: 2316 facce, 3474 edge.

4.2.7 Dinosauro

L'oggetto Dinosauro è l'oggetto più voluminoso sul quale l'algoritmo è stato testato; è costituito infatti da 112384 facce, 168576 edge.
Anche questo modello presenta dei dettagli molto interessanti come la colonna vertebrale e le zampe. In figura 4.12 vengono mostrate le varie semplificazioni applicate all'oggetto:

a Originale

b Semplificazione al 50%: 56184 facce, 84276 edge.

c Semplificazione al 70%: 33712 facce, 50568 edge.

d Semplificazione al 90%: 11232 facce, 16848 edge.

e Semplificazione al 95%: 5616 facce, 8424 edge.

Notare come anche a percentuali di semplificazione molto alte il modello appaia ancora molto buono.

In figura 4.13 viene mostrata la mesh Dinosauro dopo una semplificazione del 99.5% (556 facce,280 vertici e 834 edge).

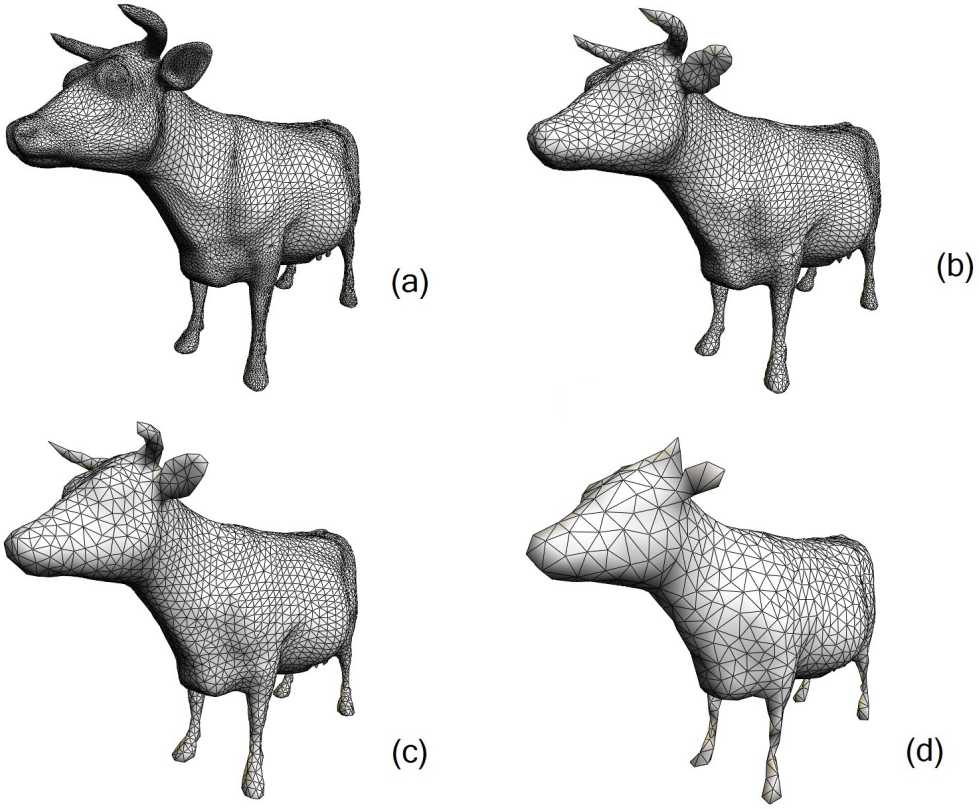

Figura 4.11: Effetti dell'applicazione dell'algoritmo alla mesh Mucca con diverse percentuali di semplificazione

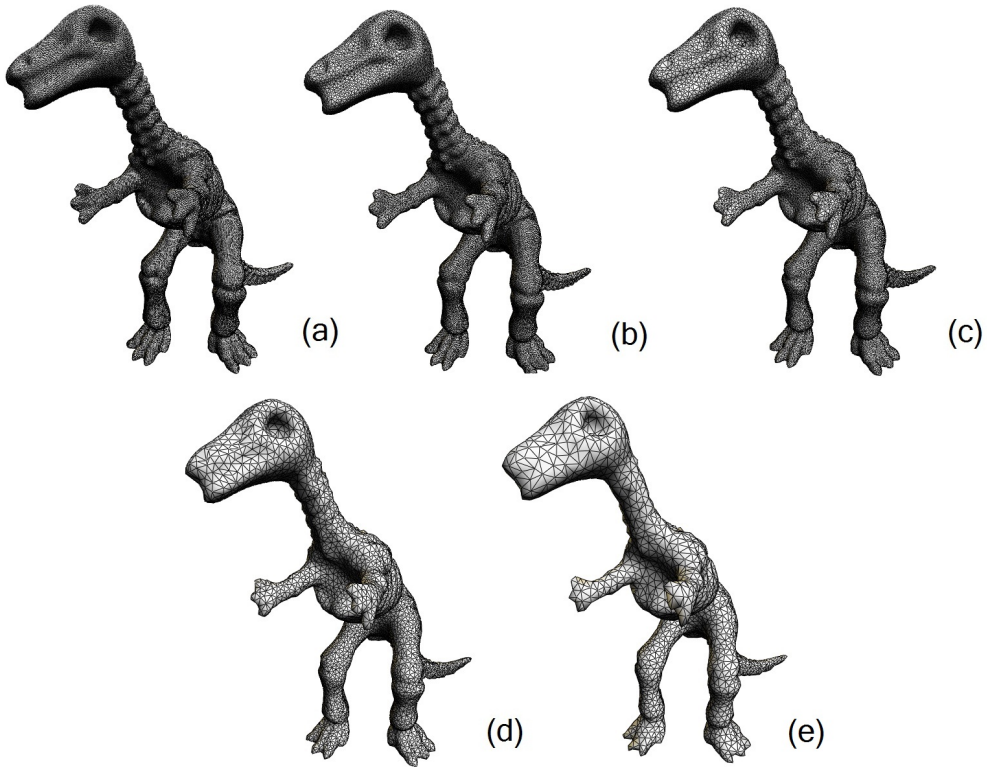

Figura 4.12: Effetti dell'applicazione dell'algoritmo alla mesh Dinosaur con diverse percentuali di semplificazione

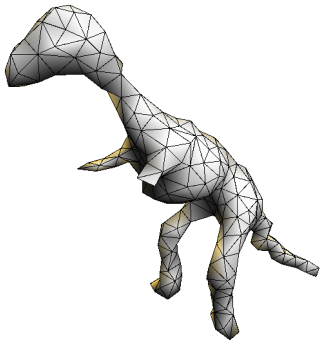

Figura 4.13: Effetti dell'applicazione dell'algoritmo alla mesh Dinosauro con percentuale di semplificazione del 99.5%

4.3 Esempio 2

In questo esempio viene mostrata l'evoluzione della mesh calco di una dentiera ad ogni iterazione dell'algoritmo, fino ad arrivare al risultato finale. La mesh iniziale è il calco della dentiera mostrato in figura 4.1, mesh **a**. In figura 4.14 possiamo notare come procede la semplificazione della mesh dopo ogni iterazione dell'algoritmo. In questo caso è stata richiesta una semplificazione dell'80% in 4 iterazioni dell'algoritmo.

In figura 4.15 applicando l'algoritmo di semplificazione alla mesh Mucca, notiamo come le aree a più alta curvatura vengano preservate anche dopo molte semplificazioni.

Si vuole osservare che le mesh intermedie prodotte risultano tutte di ottima qualità e possono quindi essere sfruttate in un algoritmo di LOD che utilizza semplificazioni progressive.

4.4 Esempio 3

In questo esempio vengono mostrati nel dettaglio gli effetti che l'applicazio delle operazioni di smoothing e decimation ha sulla mesh durante l'elaborazione. In figura 4.16 si può vedere l'effetto di un'iterazione e di tre iterazioni di smoothing sulla stessa zona della mesh Fandisk. Come si può notare l'algoritmo di smoothing accorpa i vertici verso le aree a maggior curvatura, ma allo stesso tempo preserva le caratteristiche delle zone dove la curvatura è

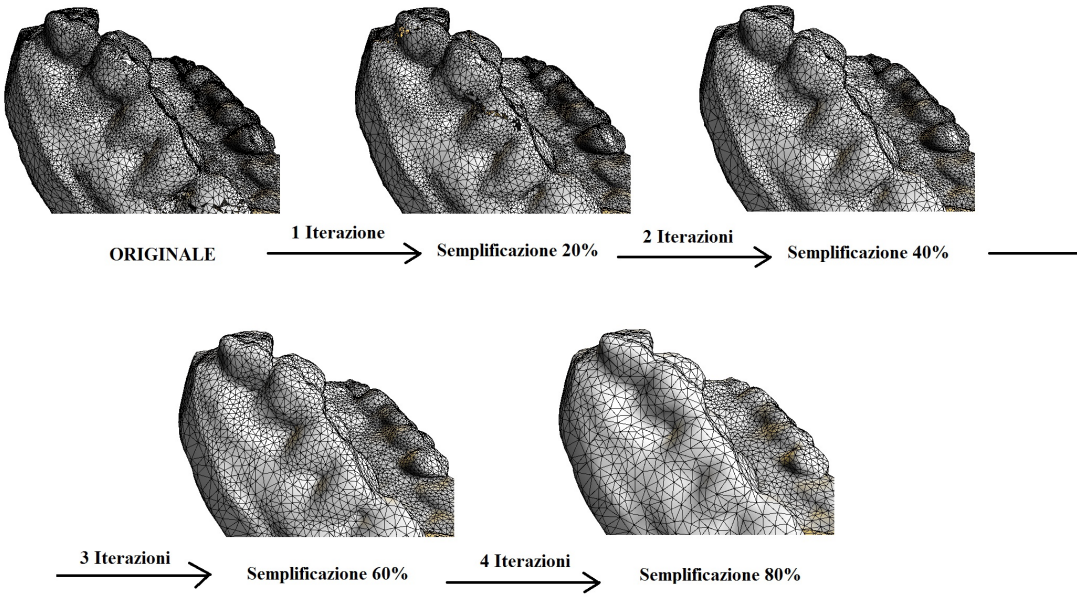

Figura 4.14: Particolare della superficie della mesh a diversi livelli di semplificazione dopo ogni iterazione

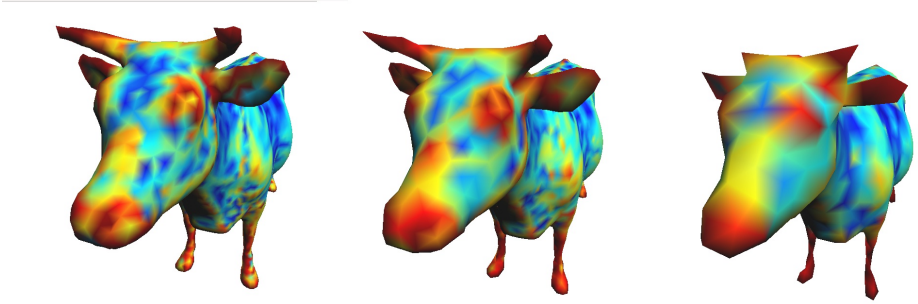

Figura 4.15: Curvatura evidenziata nella mesh Mucca, semplificate rispettivamente a (da sinistra verso destra) 30%,50%,70%

maggiore. Nello specifico, in questo esempio vediamo come i vertici vengano accorpati vicino ai bordi ma in modo da lasciare intatti questi bordi netti che sono caratteristici dell'oggetto in questione e vanno quindi preservati.

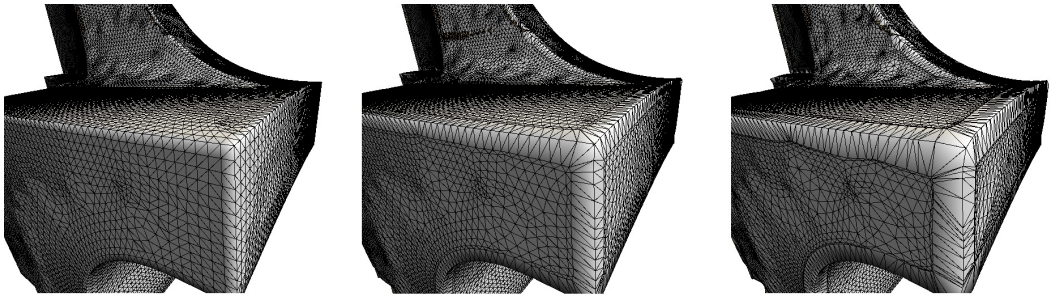

Figura 4.16: Effetti dell'algoritmo di smoothing dopo una e dopo tre iterazioni

Se la fase di smoothing non venisse eseguita, il solo algoritmo di decimazione avrebbe molti problemi a capire dove effettuare le semplificazioni, soprattutto in mesh regolari (edge tutti circa della stessa lunghezza). Per evidenziare l'importanza della fase di smoothing si mostra in fig. 4.17 il risultato di una semplificazione del 50% della mesh Fandisk in una sola iterazione senza la fase di smoothing. Si nota sono state danneggiate aree dove invece andrebbe mantenuto un certo livello di dettaglio. In figura 4.18 viene mostrato l'output della fase di decimazione per le corrispondenti mesh di figura 4.16.

Andando ad applicare invece i due algoritmi smoothing e decimation uno di seguito all'altro si nota come la qualità della semplificazione sia molto più alta.

Mettendo a confronto (figura 4.19) i modelli ottenuti dopo la fase di smoothing e dopo la decimazione si nota bene su quali aree vada ad agire la decimazione grazie alla fase di smoothing. Nella prima riga di fig. 4.19 è mostrato l'output della fase di smoothing (a sinistra) e l'output della successiva fase di decimation sulla destra applicando una sola iterazione di smoothing. Nella seconda riga di fig. 4.19 è mostrato l'output della fase di smoothing (a sinistra) e l'output della successiva fase di decimation sulla destra applicando tre iterazioni di smoothing.

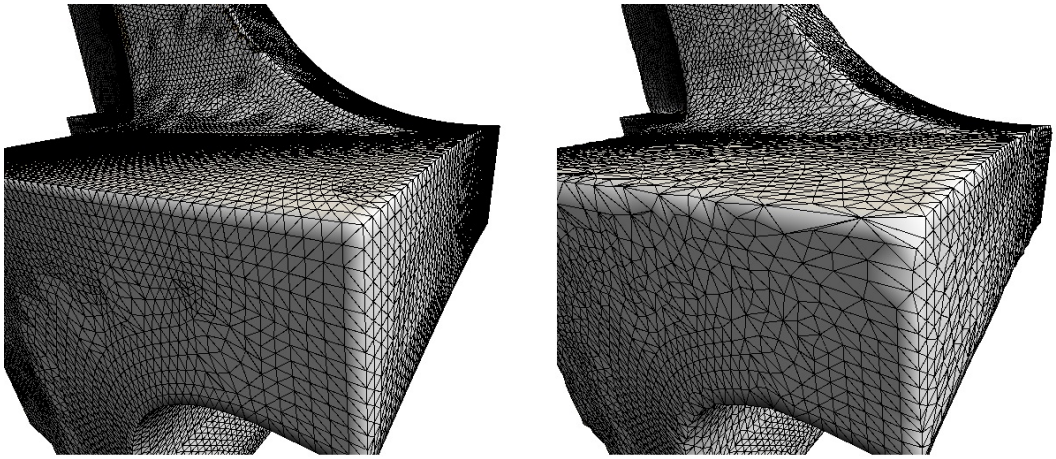

Figura 4.17: Effetti dell'algoritmo di decimazione con una percentuale di semplificazione richiesta del 50% e una sola iterazione (OUT_ITS)

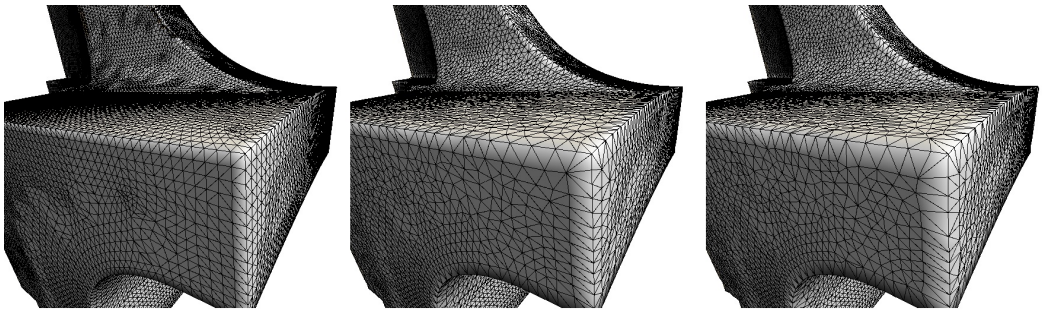

Figura 4.18: Effetti dell'applicazione dell'algoritmo di decimazione nei casi descritti precedentemente

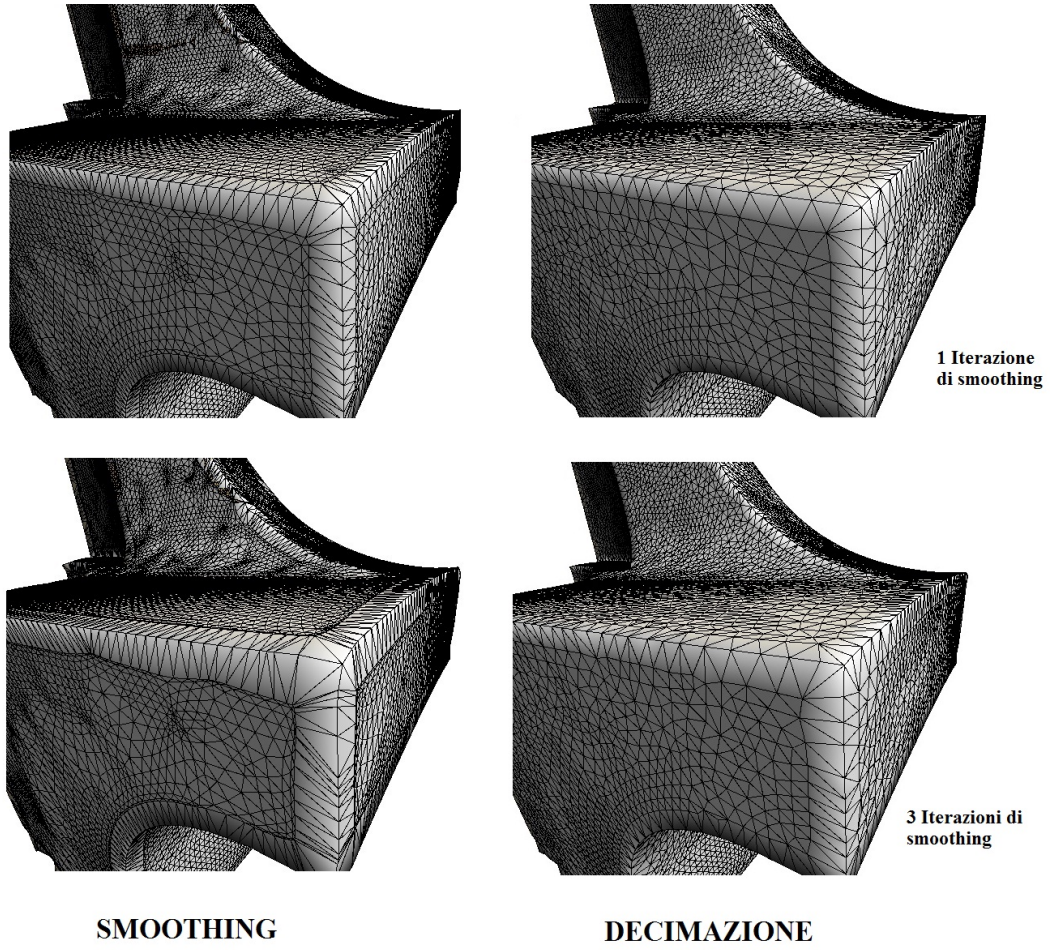

Figura 4.19: Effetti dello smoothing e della successiva decimazione

4.5 Esempio 4

Questo esempio mostra l'output della sola fase di smoothing con diversi valori dell'esponente p del p-Laplaciano. In figura 4.20 si può notare come con $p < 1$ l'effetto dell'operatore p-Laplaciano sia di accorpare i vertici nelle aree ad alta curvatura mentre con valori superiori effettua appunto uno smoothing vero e proprio della superficie.

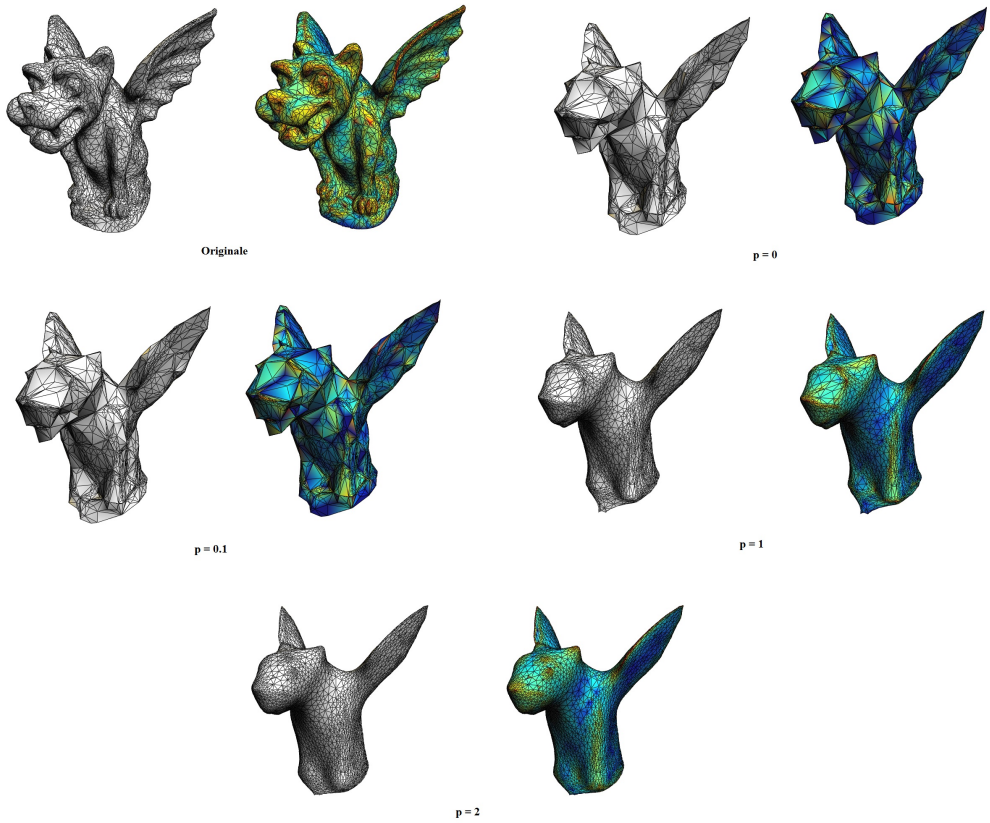

Figura 4.20: Output della fase di smoothing con diversi valori di p

4.6 Esempio 5

In questo esempio si vuole mostrare l'influenza del numero di iterazioni (OUT_ITS) sul risultato. In figura 4.21 viene mostrata la mesh Dinosauro,

dalla stessa angolazione dopo una semplificazione del 90%.
La stessa mesh semplificata è stata ottenuta prima con due, poi con quattro e infine con sei iterazioni.

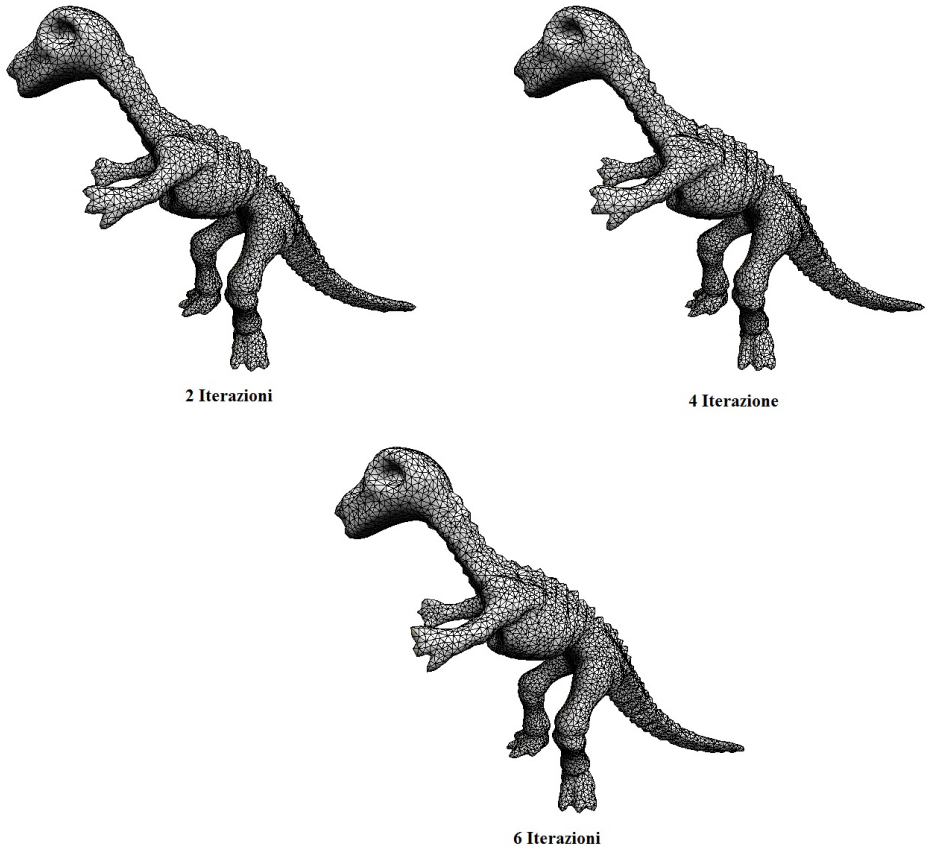

Figura 4.21: Effetti dell'applicazione dell'algoritmo alla mesh Dinosaur con percentuale a 90% e un diverso numero di iterazioni

Sperimentalmente si è osservato che quattro è un numero di iterazioni giusto in questo tipo di mesh per avere la migliore qualità di output.

4.7 Esempio 6

Confronto qualitativo e quantitativo effettuato mettendo a confronto i modelli ottenuti dal nostro algoritmo con quelli ottenuti sulle stesse mesh dall'algoritmo di Garland (Quadric Edge Collapse Decimation) implementato dal software Meshlab.
La valutazione in questo caso è stata anche di tipo quantitativo. Come valore discriminante è stata utilizzata la distanza di Hausdorff che fornisce la distanza media e massima dei punti appartenenti alla mesh semplificata rispetto a quelli appartenenti alla mesh originale.
Le misurazioni sono state effettuate sulle mesh di figura 4.22.

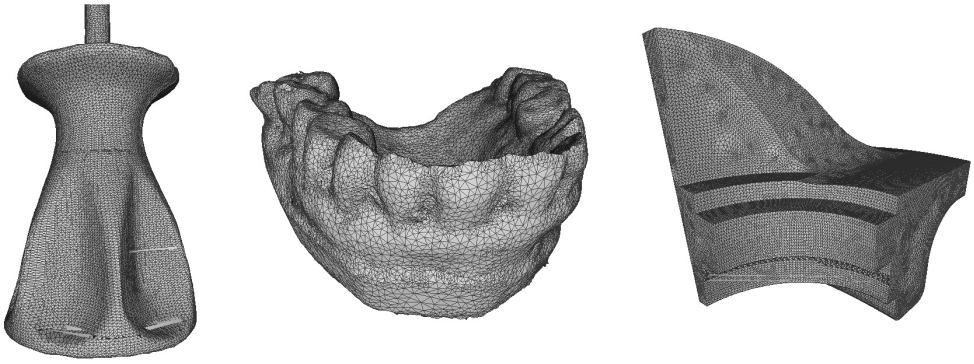

Figura 4.22: Mesh utilizzate per i test di confronto

In figura 4.23 sono mostrate le mesh create con Meshlab e quelle create con il nostro algoritmo. Qualitativamente sono paragonabili ma i valori delle distanze di Hausdorff mostrano che a livello quantitativo le mesh ottenute con Meshlab sono leggermente più fedeli ai modelli originali.

Anche nei confronti mostrati in figura 4.24 e 4.25 si vede come i risultati visivamente siano assolutamente comparabili, specie a percentuali di semplificazione non troppo alte, mentre le distanze di Hausdorff mostrano che anche in questi casi le mesh create con Meshlab sono leggermente migliori.

I valori mostrati sono la distanza di Hausdorff massima e media, come si può notare in tutti i casi i valori ottenuti con Meshlab sono lievemente migliori. In termini di qualità però le mesh prodotte con Simplify sono assolutamente paragonabili. I risultati sono particolarmente apprezzabili

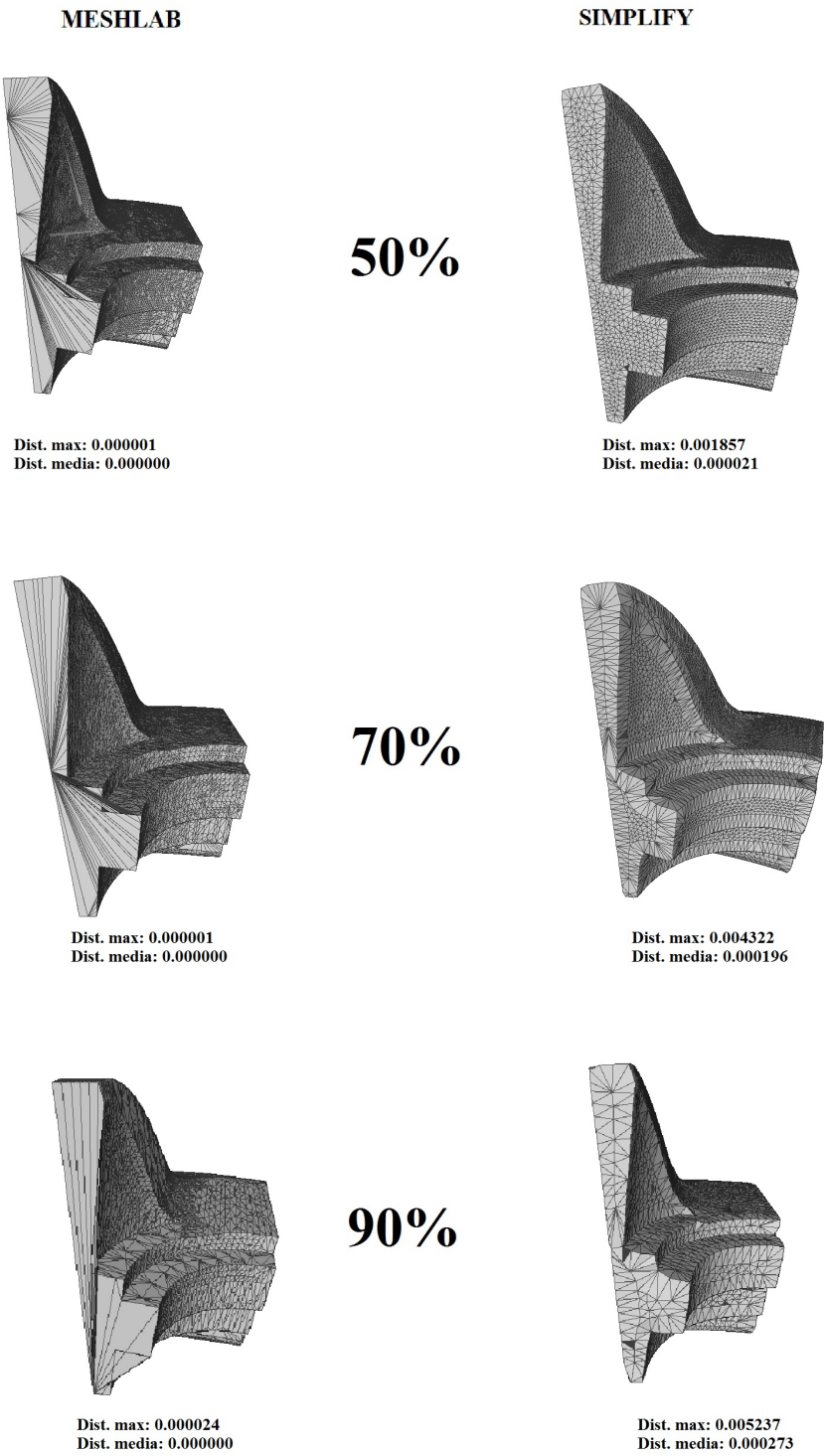

Figura 4.23: Semplificazioni della mesh Fandisk ottenute con entrambi i metodi e relative distanze di Hausdorff

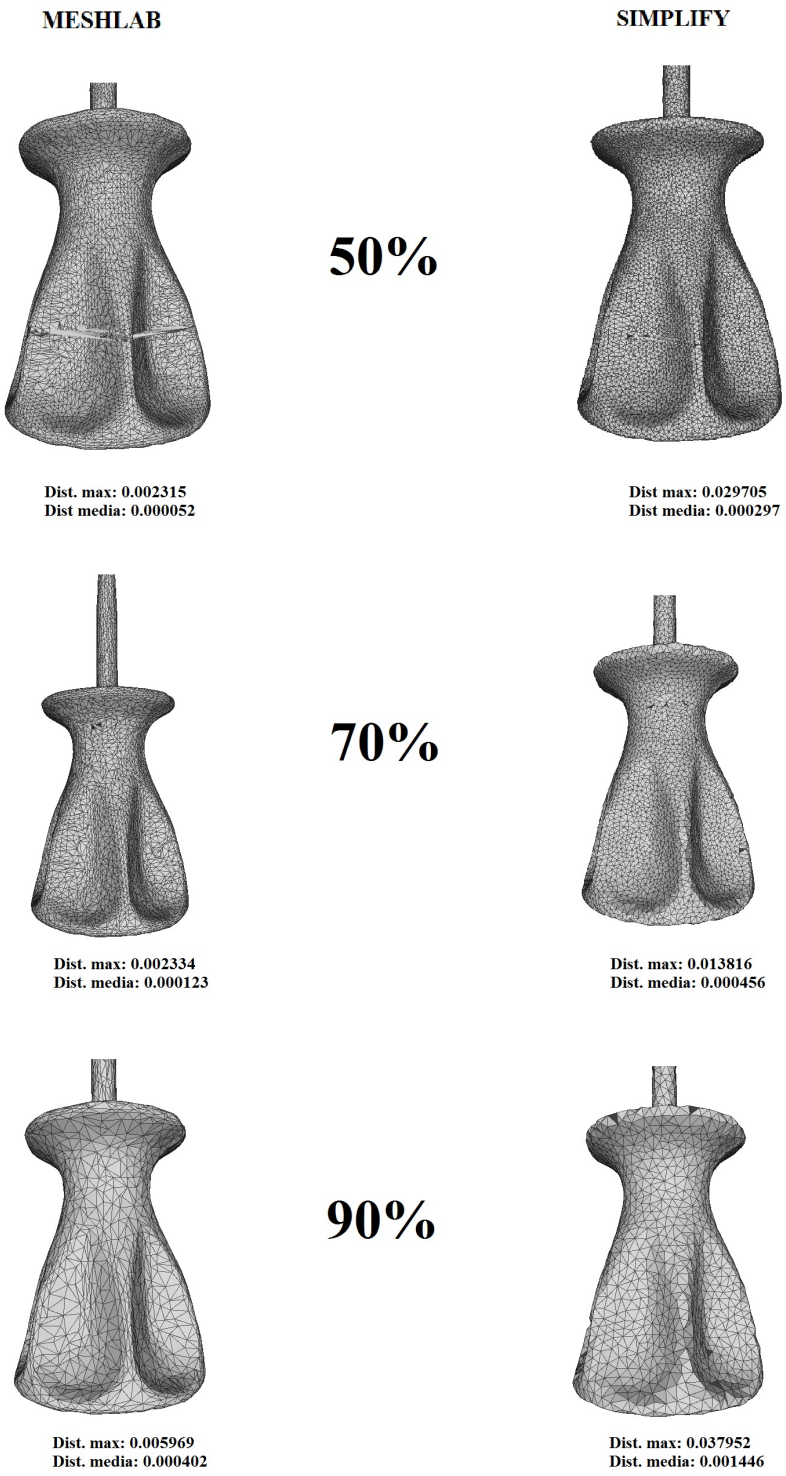

Figura 4.24: Semplificazioni della mesh Cacciavite ottenute con entrambi i metodi e relative distanze di Hausdorff

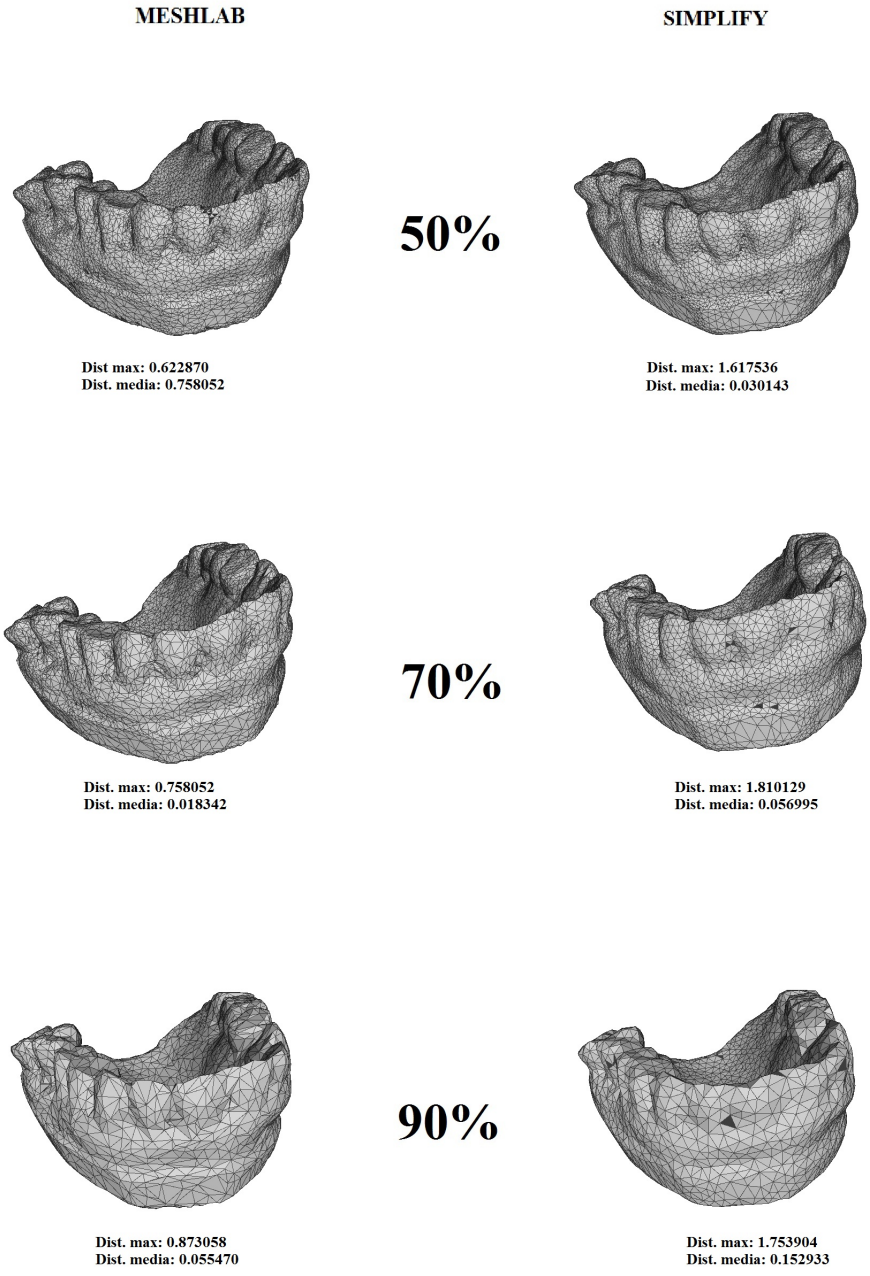

Figura 4.25: Semplificazioni della mesh Dentiera ottenute con entrambi i metodi e relative distanze di Hausdorff

nelle mesh di figura 4.23 dove si può vedere una distribuzione dei triangoli nettamente migliore nel caso di Simplify. Le mesh semplificate da Garland e mostrate nella colonna di sinistra in fig. 4.23 sono assolutamente non utilizzabili.

Capitolo 5

Conclusioni

L'obiettivo di questa tesi è stato fin da subito molto chiaro, realizzare un software che permettesse di generare dei modelli di mesh semplificati, a partire da una mesh originale. Oltre a questo era necessario che il software fosse in grado di confrontarsi come risultati con altri software che rappresentano lo stato dell'arte in libera distribuzione.
Nel primo capitolo abbiamo fornito una panoramica sul mondo delle mesh poligonali, il loro utilizzo e come lavorare con esse. Nel secondo è stata descritta la mia esperienza di acquisizione con uno scanner laser 3D e sono state fornite tutte le informazione necessarie per intraprendere un'esperienza simile. Nel terzo abbiamo mostrato il funzionamento dell'algoritmo di semplificazione da noi proposto. Infine nell'ultimo capitolo abbiamo offerto una selezione delle varie sperimentazioni fatte sul software. In questo capitolo finale ho raccolto tutte le mie impressioni sullo sviluppo di questa applicazione e sui successivi miglioramenti che si possono apportare.
L'esperienza fatta è stata senza dubbio molto positiva, non solo per l'utilizzo che ho potuto fare di tecnologie a me prima sconosciute, ma anche perché l'algoritmo prodotto alla fine del lavoro è senz'altro un ottimo prototipo. Ci sono molti margini di miglioramento per una futura ottimizzazione ma la base è già decisamente solida.
Dai dati sperimentali mostrati nel capitolo finale sulle sperimentazioni è emerso chiaramente che già a questo livello la qualità dei modelli prodotti è paragonabile a quella ottenibile attraverso i software più utilizzati nel campo, come appunto Meshlab.
Purtroppo come è stato fatto notare, i risultati quantitativi non sono an-

cora paragonabili a quelli degli altri software ma continuando lo sviluppo dell'applicazione non escludo che in futuro si potranno superare.
Riguardo a questo punto è necessario tuttavia un commento sull'utilizzo di Meshlab per il confronto. Il problema è che la documentazione di Meshlab non è di facile accesso e quindi in realtà non possiamo sapere se oltre all'algoritmo di Garland si utilizzi, durante lo svolgimento della semplificazione, anche altre tecniche per migliorare il modello generato.
In conclusione il lavoro svolto è stato decisamente utile e mi ha portato ad avere una profonda conoscenza e comprensione delle mesh poligonali e del loro utilizzo.
Tutti gli obiettivi della tesi sono stati raggiunti e non potrei essere più soddisfatto del lavoro svolto.

Bibliografia

[1] J. Rossignac and P. Borrel, Multi-Resolution 3D Approximations for Rendering Complex Scenes, Geometric Modeling in Computer Graphics, Springer-Verlag, Berlin, 1993, pp. 455-465.

[2] K-L. Low and T.S Tan, Model Simplification Using Vertex Clustering, Proc. 1997 Symp. Interactive 3D Graphics, ACM Press, New York, 1997, pp. 75-82.

[3] P. Lindstrom, Out of core simplification of large polygonal models, Computer Graphics (Proc. SIGGRAPH 2000), vol. 34, ACM Press, New York, 2000, pp. 259-262.

[4] M. Garland and P. Heckbert, Simplification Using Quadric Error Metrics, Computer Graphics (Proc. SIGGRAPH 97), vol. 32, ACM Press, New York, 1997, pp. 209-216.

[5] H. Hoppe, Progressive Meshes, Computer Graphics (Procs. Of SIGGRAPH 96), vol. 30, ACM Press, New York, 1996, pp. 99-108.

[6] H. Hoppe, View-Dependent Refinement of Progressive Meshes, Computer Graphics (Procs. Of SIGGRAPH 97), vol. 31, ACM Press, New York, 1997, pp. 189-198.

[7] D. Lazzaro and L.B. Montefusco, Radial basis functions for the multivariate interpolation of large scattered data sets, Journal of Computational and Applied Mathematics 140, 2002, pp. 521536.

[8] G. Casciola, D.Lazzaro, L.B. Montefusco, S.Morigi, Fast surface reconstruction and hole filling using positive definite Radial Basis Functions, Department of Mathematics, University of Bologna, Bologna, 2004.

[9] A. Elmoataz, O. Lezoray, S. Bougleux, Nonlocal Discrete Regularization on Weighted Graphs: a framework for Image and Manifold Processing, University of Caen, Caen, 2007, pp. 5-8.

[10] M. Attene, Algoritmi per la semplificazione di griglie di triangoli, Università degli Studi di Genova.

[11] L. Kobbelt, M. Botsch, A survey of point-based techniques in computer graphics, Computers & Graphics, vol. 28, Elsevier Ltd., 2004.

[12] C. Gotsman, S. Gumhold, and L. Kobbelt. Simplification and compression of 3d meshes. In M. Floater A. Iske, E. Quak, editor, Tutorials on multiresolution in geometric modeling. Springer, 2002.

[13] O. Sorkine, Laplacian Mesh Processing, School of Computer Science, Tel Aviv University, Israel, 2005 pp. 2-3.

Elenco delle figure