# Esercitazione 3 - MeshLab

**Punto 2**

Nella seconda parte dell'esercitazione si richiedeva una sperimentazione nell'utilizzo di MeshLab, un tool che consente l'elaborazione di nuvole di punti e mesh poligonali.

<u>Parte a</u>

Obiettivo: ricostruzione di una mesh a partire da una nuvola di punti (e le relative normali). Innanzitutto si è generato un oggetto sotto forma di nuvola di punti attraverso l'utilizzo di Blender. Fatto ciò, si è importata la nuvola di punti su MeshLab.

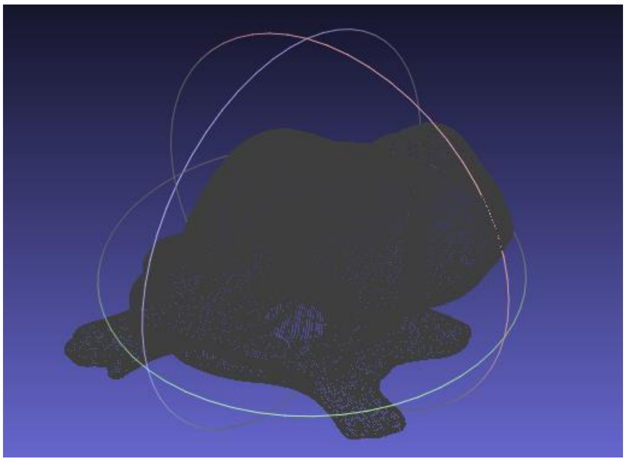


*Figura 1 - Nuvola di punti*

Per la ricostruzione della mesh si è utilizzato l'algoritmo **Ball Pivoting** (comando *Surface Reconstruction -> Ball Pivoting*). Il principio fondamentale alla base dell'algoritmo è molto semplice: tre punti formano un triangolo se una sfera, di raggio specificato dall'utente, li contiene senza includere altri punti. L'algoritmo parte da un triangolo (triangolo seme), la sfera si sposta lungo uno spigolo (ruota attorno al bordo mantenendo il contatto con i vertici del bordo) finché non tocca un altro punto, formando un triangolo. Il processo continua fino a quando non sono stati provati tutti i bordi raggiungibili. A questo punto si considera un altro triangolo iniziale; l'algoritmo termina quando sono stati considerati tutti i punti. Per ottenere una maggiore precisione nella ricostruzione si può eseguire diverse volte l'algoritmo con raggio della sfera crescente per gestire correttamente densità di campionamento irregolari.

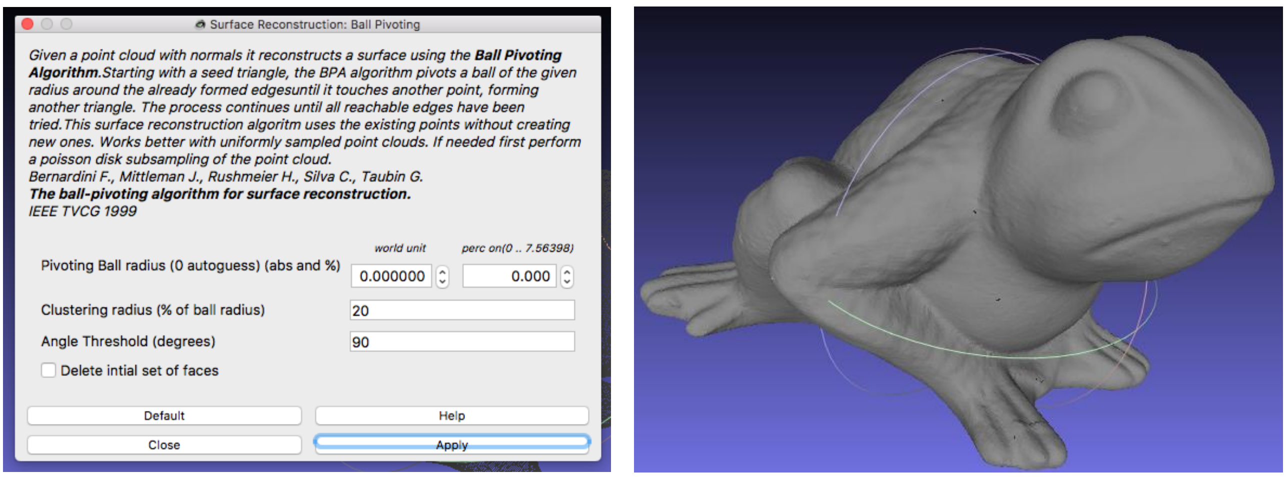



*Figura 2 - A sinistra il menù per l'applicazione dell'algoritmo, a destra la mesh ricostruita (raggio = 30)*

Parte b

Obiettivo: riparazione di una mesh parzialmente corrotta.
La mesh utilizzata è una di quelle fornite (file *palmetta.obj*).

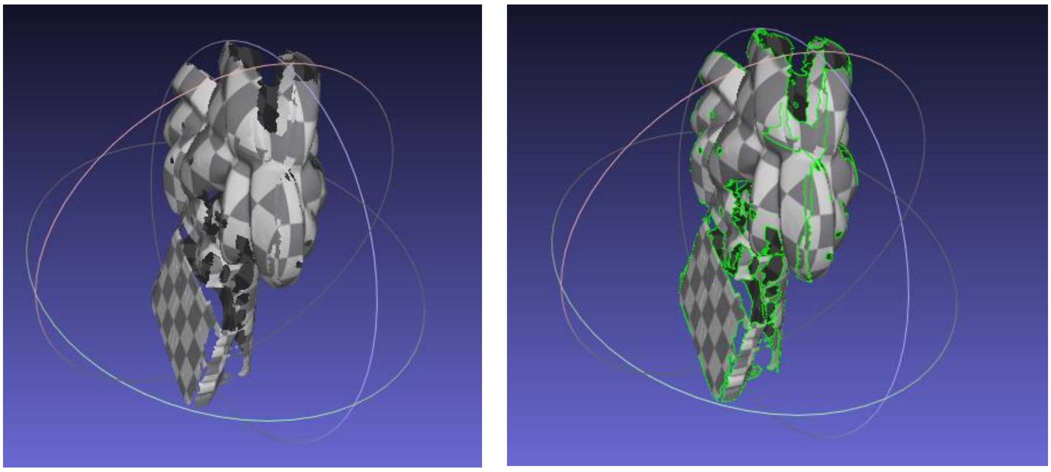


*Figura 3 - Mesh palmetta.obj*

Per la riparazione si sono utilizzate due tecniche:

- algoritmo *Close Holes*
- algoritmo *Screened Poisson Surface Reconstruction*

Il risultato dell'applicazione del primo algoritmo non è del tutto soddisfacente, come mostrato nelle seguenti immagini:

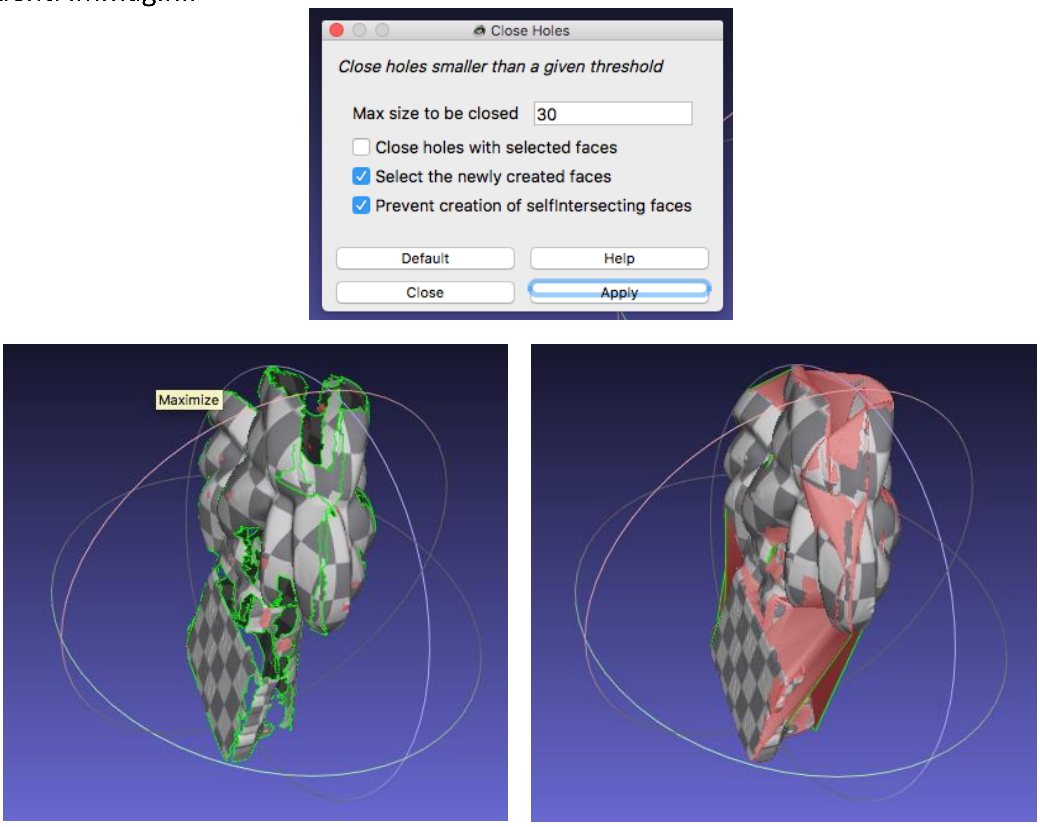



*Figura 4 - In alto il menù per l'applicazione dell'algoritmo, in basso a sinistra la mesh dopo l'applicazione dell'algoritmo con parametro size = 30 e in basso a destra con il parametro size = 1000*

L'applicazione del secondo algoritmo invece dà sicuramente dei risultati migliori.

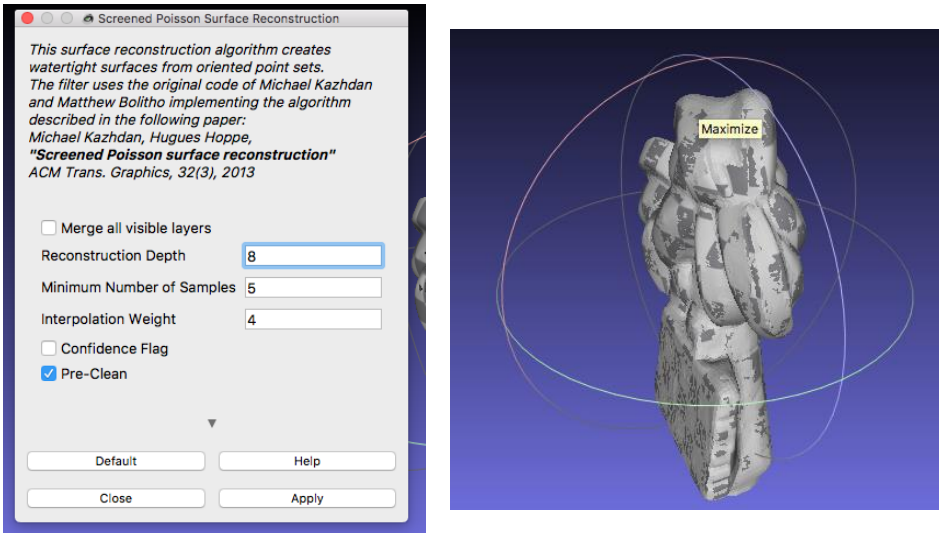



*Figura 5 - A sinistra il menù di applicazione dell'algoritmo, a destra la mesh dopo l'applicazione dell'algoritmo (parametro Minimum Number of Samples = 5)*

<u>Parte c</u>

Obiettivo: applicazione di un filtro di denoising ad una mesh perturbata.
La mesh utilizzata è una di quelle fornite (file *stell2perturb.obj*).

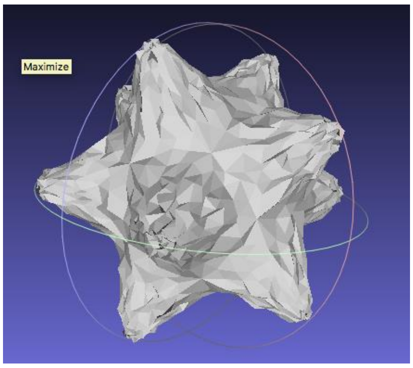



*Figura 6 - Mesh stell2perturb.obj*

Per ogni vertice della mesh l'algoritmo calcola la media delle posizioni dei vertici adiacenti e lo riposiziona di conseguenza. In questo modo, la topologia dell'oggetto non cambia ma si ha un effetto *smooth*.

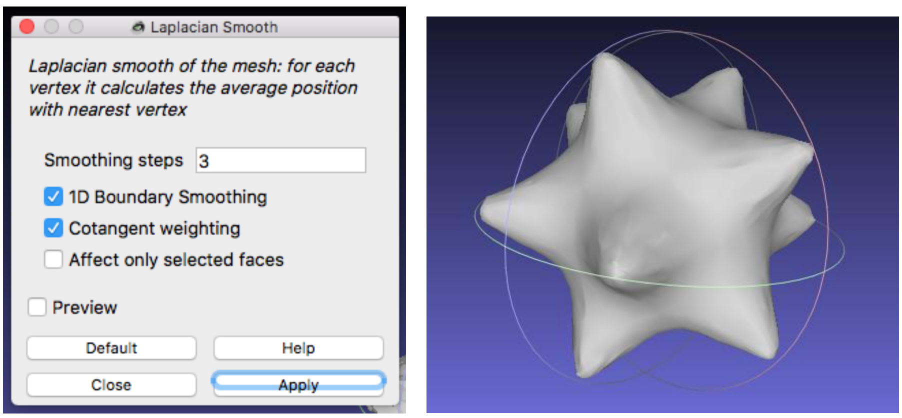



*Figura 7 - A sinistra il menù per l'applicazione dell'algoritmo, a destra la mesh dopo l'applicazione dell'algoritmo con parametro Smoothing steps = 3*

Parte d
Obiettivo: semplificazione a più livelli di una mesh con un numero elevato di elementi.
Anche in questo caso si sono utilizzate due tecniche:

- algoritmo MC Edge Collaption
- algoritmo Edge Collapse Decimation

Sono algoritmi che trasformano una mesh poligonale in un'altra con meno facce, spigoli e vertici (minore complessità di gestione, di memorizzazione, ecc…). Le operazioni compiute possono essere di due tipi:

- rimozione di un vertice
    - rimozione del vertice
    - rimozione degli spigoli e delle facce adiacenti al vertice
    - triangolazione del buco che si è formato (*k* - 2 triangoli, dove *k* è la valenza del vertice rimosso)
- rimozione di uno spigolo
    - dato uno spigolo di vertici *p* e *q*, l'algoritmo sostituisce i vertici con un unico vertice collocandolo in un determinato punto sul segmento che unisce i vertici *p* e *q*

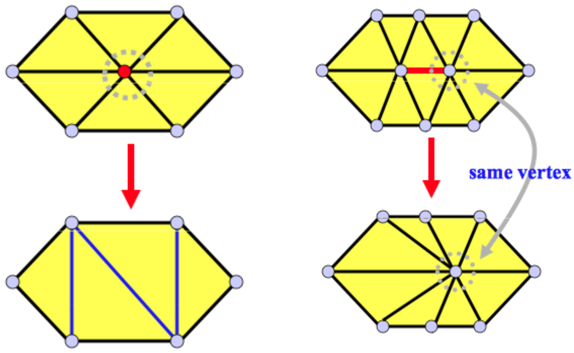



*Figura 8 - Operazioni di semplificazione*

Algoritmo Edge Collapse
Il principio dell'algoritmo è molto semplice:

- per ogni spigolo (di vertici *p* e *q*), viene calcolato il punto che minimizza l'errore dovuto alla contrazione dello spigolo stesso (N.B.: il punto si trova sul segmento che congiunge i punti *p* e *q*)
- l'insieme di spigoli viene quindi ordinato per costo crescente
- si esegue la seguente serie di passaggi fino a quando non si è raggiunto il numero di facce desiderato o non sono possibili ulteriori semplificazioni
    - si applica la contrazione dello spigolo di costo minimo - *p* e *q* vengono collassati in un unico vertice *v* (giace sul segmento che congiunge *p* e *q*)
    - si aggiorna il costo di contrazione degli spigoli adiacenti al vertice *v*
    - si riordinano gli spigoli se necessario

L'algoritmo non modifica la topologia della mesh.

L’algoritmo MC Edge Collaption non permette di specificare il numero desiderato di facce che deve avere la mesh al termine della computazione. Pertanto, il risultato non è del tutto soddisfacente.

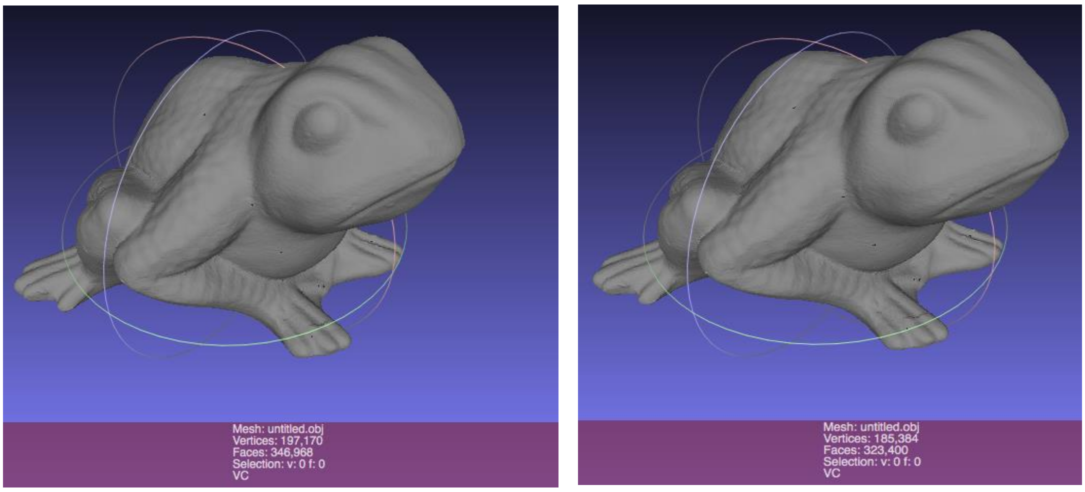
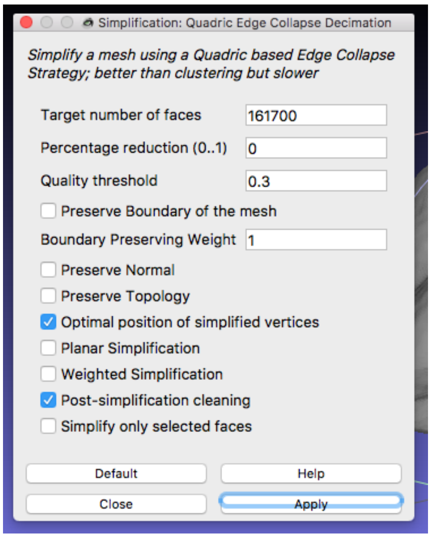



*Figura 9 - Applicazione dell'algoritmo MC Edge Collaption*

L’algoritmo Edge Collapse Decimation permette di specificare il numero desiderato di facce (*Target number of faces*). Si ha pertanto maggiore flessibilità.



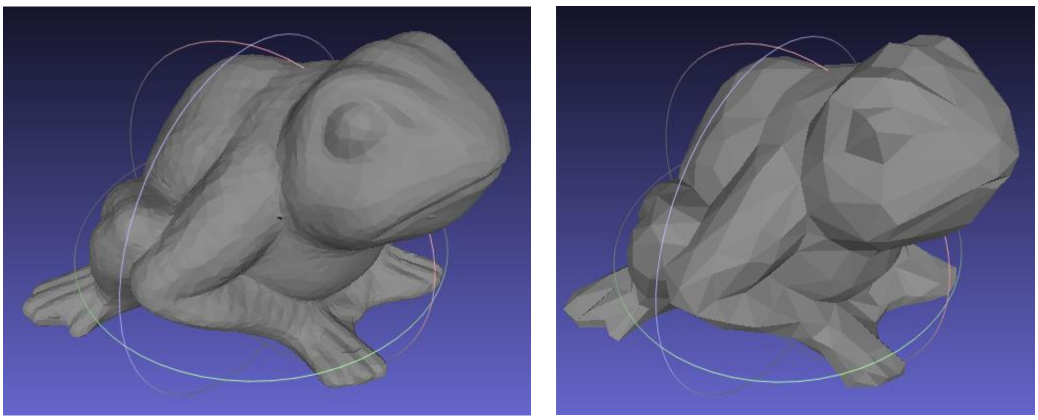



*Figura 10 - Applicazione dell'algoritmo Edge Collapse Decimation (faces = 10000, faces = 1000)*

Parte e

Obiettivo: sperimentazione degli algoritmi di valutazione della qualità di superficie.
La mesh utilizzata è una di quelle fornite (file *mannequin.obj*).

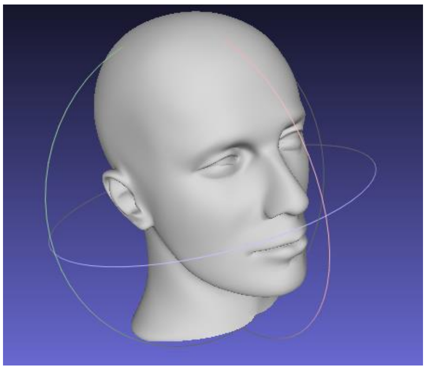


*Figura 11 - Mesh mannequin.obj*

In ogni punto P di una superficie σ, la curva Γ di intersezione tra σ ed un qualsiasi punto π che contiene la normale n alla superficie P ha una determinata curvatura ρ ed un raggio di curvatura definito dall'equazione:

$$R = \frac{1}{\rho}$$

Naturalmente, per P passa un intero fascio di piani contenenti la normale n. Tuttavia, esistono e sono uniche una direzione per la quale la curvatura è massima ed una direzione per la quale la curvatura è minima. Le curvature in tali direzioni sono dette curvature principali ($R_1$ e $R_2$).
Due combinazioni delle curvature principali sono di particolare interesse:

- curvatura media $\frac{2}{R_1+R_2}$
- curvatura gaussiana $\frac{1}{R_1 R_2}$

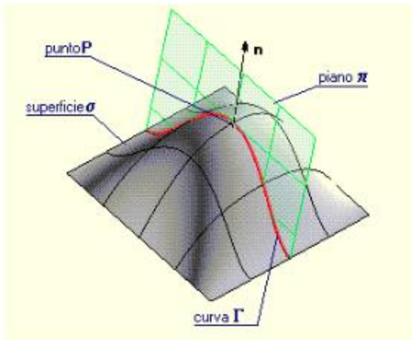



*Figura 12 - Curvatura*

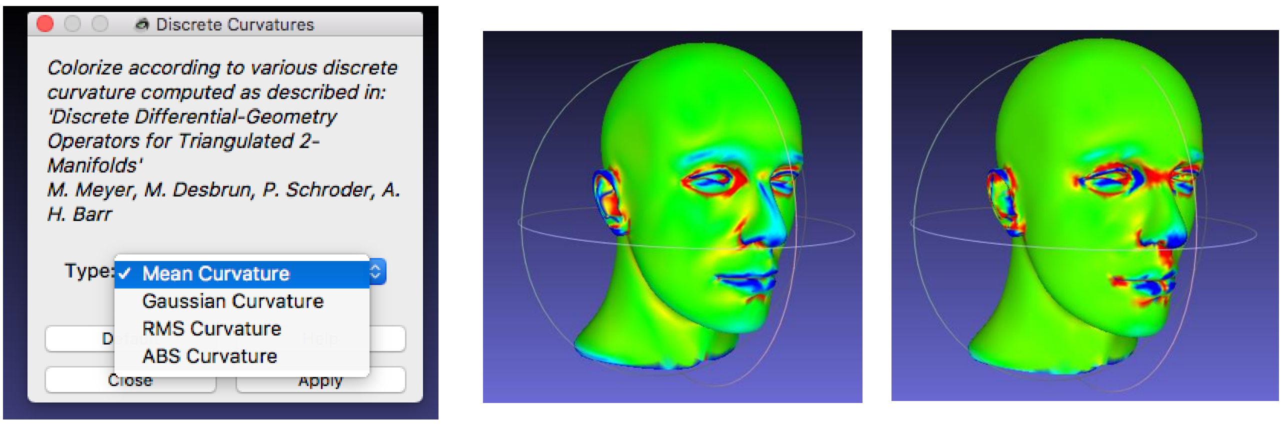



*Figura 13 - A sinistra la Mean Curvature, a destra la Gaussian Curvature*