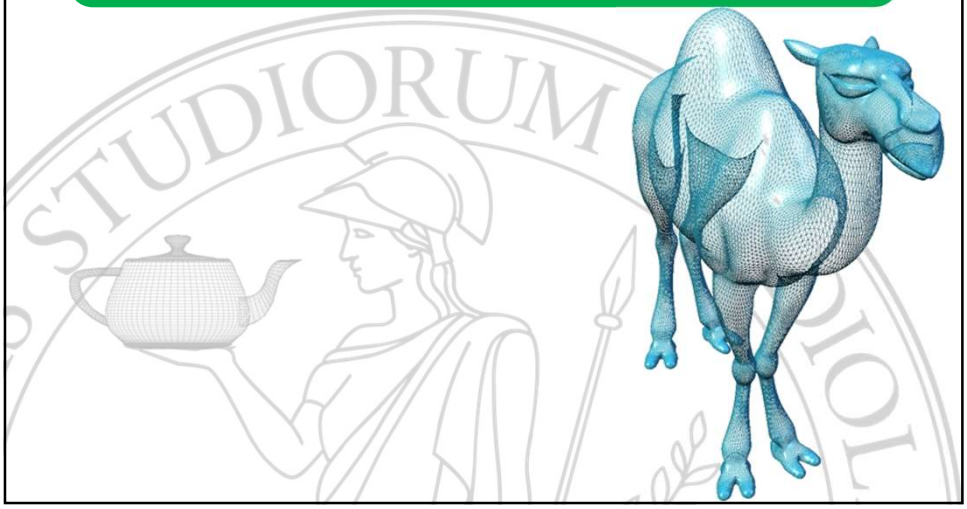
Marco Tarini - Computer Graphics 2022/2023
Università degli Studi di Milano

Polygonal Representations: cenni di Mesh Processing 2/2



145

Alcune librerie di mesh processing (C++, OpenSource)

VCG-Lib

vision and
computer graphic library

CNR ( )

CGAL

computational geometry
algorithms library

INRIA ( )

OpenMesh
+ OpenFlipper

RWTH ( )

libigl

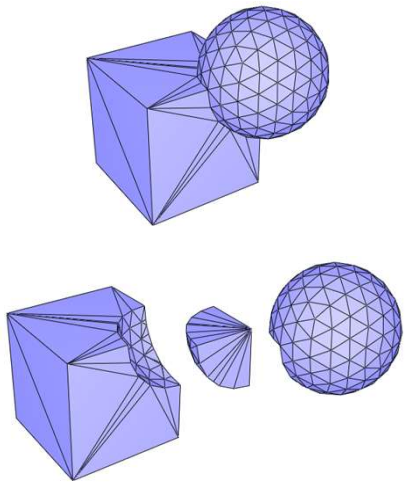
simple geometry
processing library

NYU ( )

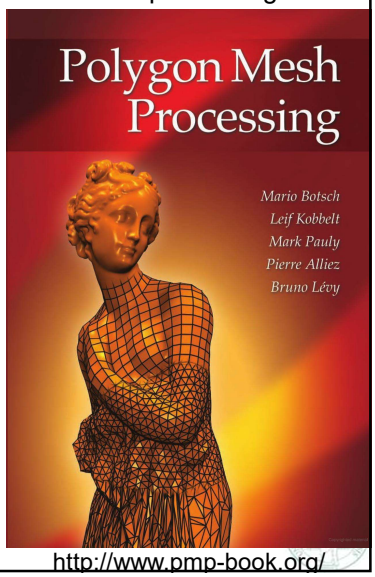
146

Mesh processing

Il **Geometry Processing** eseguito su **mesh poligonali**



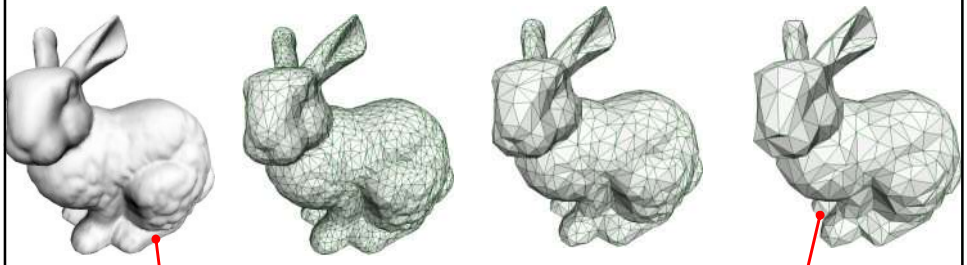
Un buon manuale per le basi al mesh processing:



http://www.pmp-book.org/

147

Struttura multires: “Level of Detail (LoD) Pyramid”



148

Piramide di livelli di dettaglio

- ✓ Idea: tenere in memoria tante versioni diverse di una mesh, a risoluzione decrescente
  - ⇒ LoD (Level of Detail) = Livello di dettaglio = una di queste versioni
  - ⇒ LoD-0 = la versione a piena risoluzione
  - ⇒ LoD-1 = versione a risoluzione ridotta
- ✓ L'insieme dei LoD è detto «piramide» di livello di dettaglio perché, al pari di una piramide, la base (LoD-0) è assai più grande (in RAM), della punta (LoD-*n*).
- ✓ In preprocessing, i livelli di dettaglio possono essere ottenuti uno dall'altro attraverso mesh-simplification
- ✓ In fase di rendering, scelgo il LoD in funzione di fattori come
  - ⇒ distanza dell'oggetto dalla telecamera, risorse di computazione a disposizione, etc
  - ⇒ le stesse considerazioni valgono per una simulazione fisica o altro
- ✓ Costo: duplicazione di dati in memoria ☹
- ✓ Beneficio: minor tempo di rendering o processing in generale

149

Piramide di livelli di dettaglio: stima del costo

- ✓ Mettiamo che ciascun LOD abbia ¼ della risoluzione del LOD precedente (scelta ragionevole e spesso adottata)
- ✓ Domanda: quanto costa in memoria l'intera piramide, rispetto al costo di memorizzare solo il modello originale (LOD-0)?
- ✓ Ricordare che:
  - ⇒ Il costo in RAM di una mesh è LINEARE con la sua risoluzione,
  - ⇒ $\sum_{i=0}^{\infty} k^i = \frac{1}{1-k}$ (con $k < 1$)
  - ⇒ che, con $k = 1/4$, fa $4/3$ (cioè $1 + 1/3$)
- ✓ Risposta: solo 1/3 in più!
- ✓ Conclusione: anche se la piramide LOD è una struttura multi-risoluzione triviale (semplice replicazione di dati a risoluzioni diverse), il suo costo in RAM non è eccessivo

150

Calcolo di distanza fra due mesh

- ✓ Task: date due mesh M_0 e M_1, quantificare quanto simili (o dissimili) siano le due superfici rappresentate
  - ⇒ Nota: mesh anche molto diverse internamente (diversi poligoni e vertici) possono rappresentare superfici molto simili, o anche *identiche*
  - ⇒ (sapresti dare un esempio?)
- ✓ E' utile a questo scopo l'uso della "distanza di Hausdorff" definita fra due superfici
  - ⇒ È zero se e solo se le due superfici coincidono (anche se le mesh che le rappresentano sono diverse)
  - ⇒ Vedi il prossimo lucido per la sua definizione esatta, su una mesh
- ✓ La misurazione di questa distanza richiede apposite algoritmi di geometry processing
  - ⇒ Meshlab include una implementazione di questi algortmi
- ✓ La distanza fra mesh è spesso usata in altri contesti di geom processing per valutare l'efficacia di un algoritmo
  - ⇒ Ad es: valutare l'errore introdotto da una semplificazione, misurando la distanza fra mesh originale e mesh semplificata

151

Definizione della distanza di Hausdorff fra 2 mesh

- ✓ Distanza punto → mesh:
  dato un punto $\mathbf{p}$*, quanto dista il punto* $\mathbf{q}$ *su* M_1 *più vicino possibile a* $\mathbf{p}$*?*

$$dist(\mathbf{p}, M_0) = \min_{\mathbf{q} \in M_1} (\|\mathbf{p} - \mathbf{q}\|)$$

  - ⇒ nota: $\mathbf{q}$ non è necessariamente un vertice di M_1 ; può anche essere un punto interno ad un triangolo.
- ✓ Distanza mesh → mesh:
  data una mesh M_0*, quanto dista al massimo un punto* $\mathbf{p}$ *scelto su* M_0 *da una seconda mesh* M_1?

$$dist(M_0, M_1) = \max_{\mathbf{p} \in M_0} \big(dist(\mathbf{p}, M_1) \big) = \max_{\mathbf{p} \in M_0} \left(\min_{\mathbf{q} \in M_1} (\|\mathbf{p} - \mathbf{q}\|) \right)$$

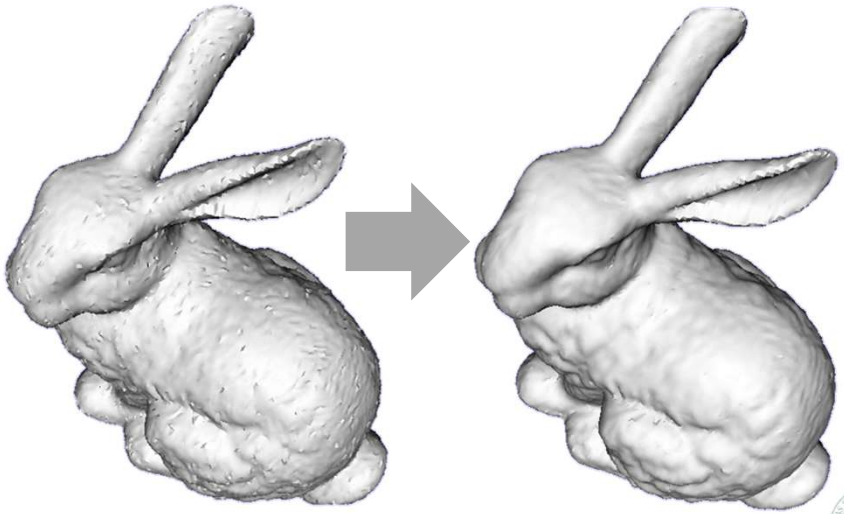
  - ⇒ nota: $\mathbf{p}$ non è necessariamente un vertice di M_0 ; può anche essere un punto interno ad un triangolo.
  - ⇒ Nota: $dist(M_0, M_1)$ non è necessariamente uguale a $dist(M_1, M_0)$
- ✓ Distanza reciproca mesh ↔ mesh

$$Hausdroff(M_0, M_1) = \max\big(dist(M_0, M_1), dist(M_0, M_1) \big)$$

152

Geometry Processing tasks: examples

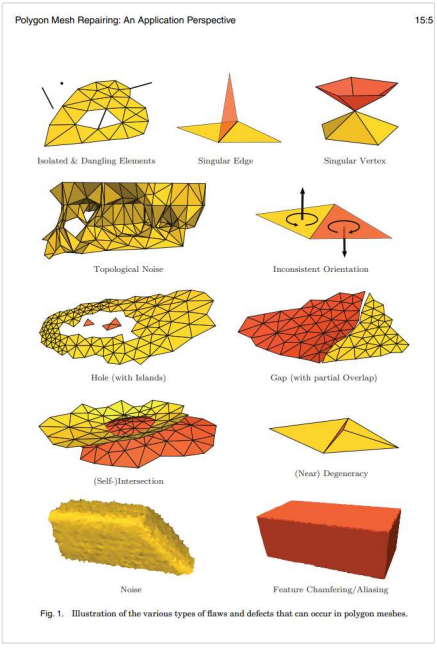
✓Mesh denoising



(img by Iman Sadeghi)

153

Geometry Processing: mesh cleaning / repearing



Pro-tip: collezione
di software a
http://meshrepair.org/

Una pagina dal survey:
"Polygon Mesh Repairing: An Application Perspective"
Attene, Campen, Kobbelt, 2012

154

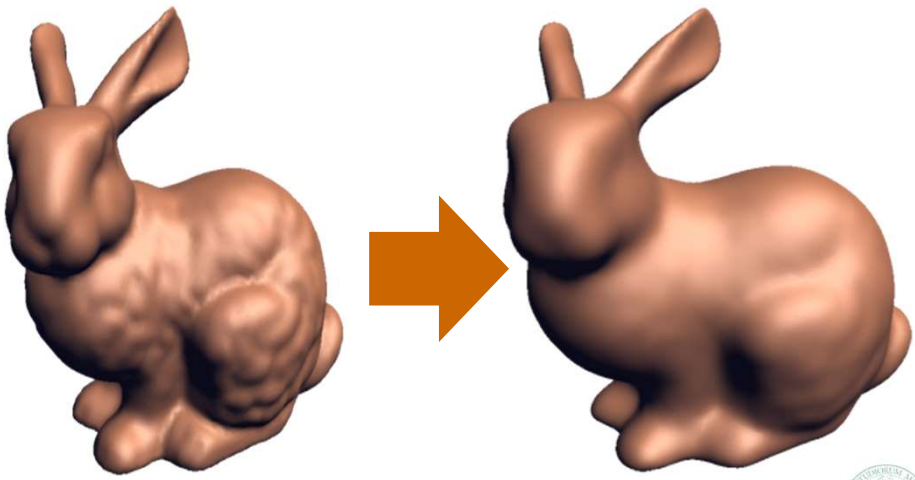
Geometry Processing: mesh cleaning / repearing

- ✓ Mesh cleaning (o repearing):
  l'insieme di task per la rimozione dei difetti della di una mesh, cioè
  - ⇒ le situazioni non two-manifold,
  - ⇒ buchi (per es un poligono mancante)
  - ⇒ Orientamento non consistente delle facce
    (se non è possibile farlo, ma mesh non è «ben orientabile»)
  - ⇒ Auto-intersezioni
  - ⇒ Replicazioni di vertici
  - ⇒ Facce degeneri
- ✓ Molte categorie di mesh, come le mesh scansionate (acquisizione 3D), presentano spesso molti di questi difetti
- ✓ Spesso, è un preprocessing necessario per consentire l'applicazione di vari altri procedimenti di geometry processing

155

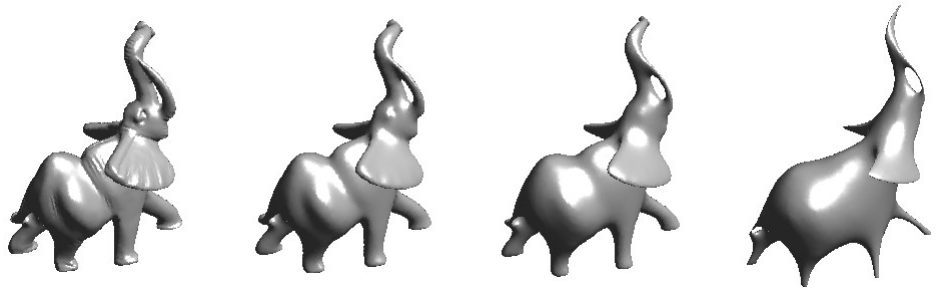
Geometry Processing tasks: examples

✓ Mesh smoothing



156

Mesh smoothing



157

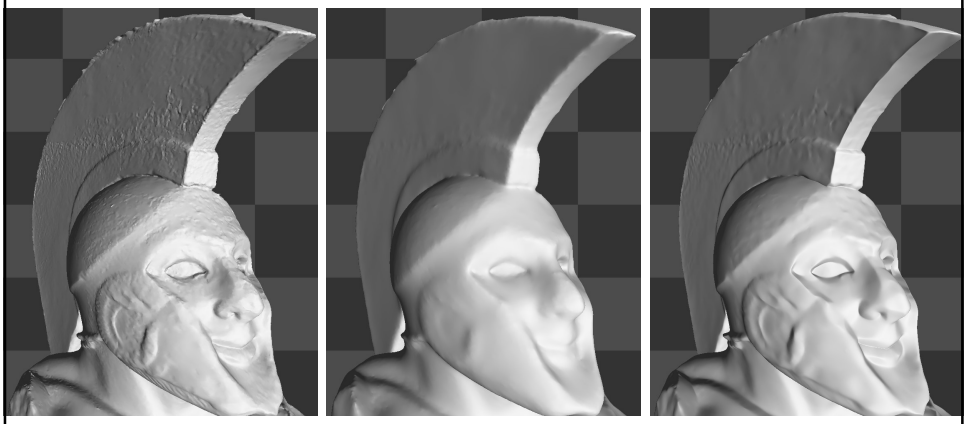
Mesh smoothing

- ✓ Obiettivo: rendere la mesh più tondeggiante e liscia
- ✓ How to (concettualmente, base):
  basta spostare ogni vertice nella posizione media dei vertici a lui adiacenti
  - ⇒ vertici adiacenti = vertici connessi da un edge
  - ⇒ Nota: la **connettività** della mesh rimane inalterata: si modifica solo la sua **geometria**
- ✓ Il processo si può ripetere molte volte, cumulandone gli effetti
- ✓ Analogo concettuale del processo di sfocatura (*blur*) di un'immagine
  - ⇒ se si sostituisce ad ogni pixel il valore medio dei pixel vicini, l'immagine diviene sfocata
- ✓ Concetto matematico connesso: filtro «Laplaciano»
  - ⇒ Minimizzazione del Laplaciano, cioè della distanza di ogni vertice dalla media dei suoi vicini
- ✓ L'operazione può essere fatta anche sugli attributi (colore, etc)
  - ⇒ oppure, *solo* sugli attributi
- ✓ Effetti desiderati:
  - ⇒ si abbatte il rumore
  - ⇒ si riducono le asperità della superficie
- ✓ Effetti collaterali (spesso indesiderati):
  - ⇒ si abbattono anche i dettagli di forma utili (ad alta frequenza)
  - ⇒ si perdono ad esempio gli spigoli vivi («crease angle» o «feature lines»)
  - ⇒ il volume della mesh tende a ridursi

158

Mesh feature preserving smoothing

- ✓ Feature preserving smoothing
  - ⇒ Ottenere uno smoothing che rimuove il rumore e le asperità a piccola scala ma evita gli «effetti collaterali» sopra descritti.
  - ⇒ Task studiato e più difficile



Originale (mesh scansionata) | Smoothing | Feature preserving smoothing

(images by Andrea Maggiordomo)

159

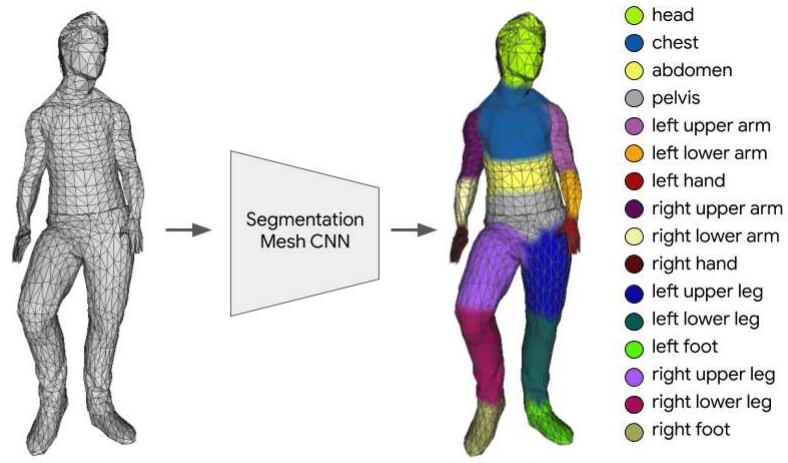
Stima di proprietà geometriche

Img by Karan Singh

- ✓ Qui: curvatura (intrinsica)
- ✓ Abbiamo visto il caso della stima delle normali
  - ⇒ Sono solo un caso di proprietà di “forma” definite sulla superfice
  - ⇒ Proprietà di cui si occupa la geometria differenziale

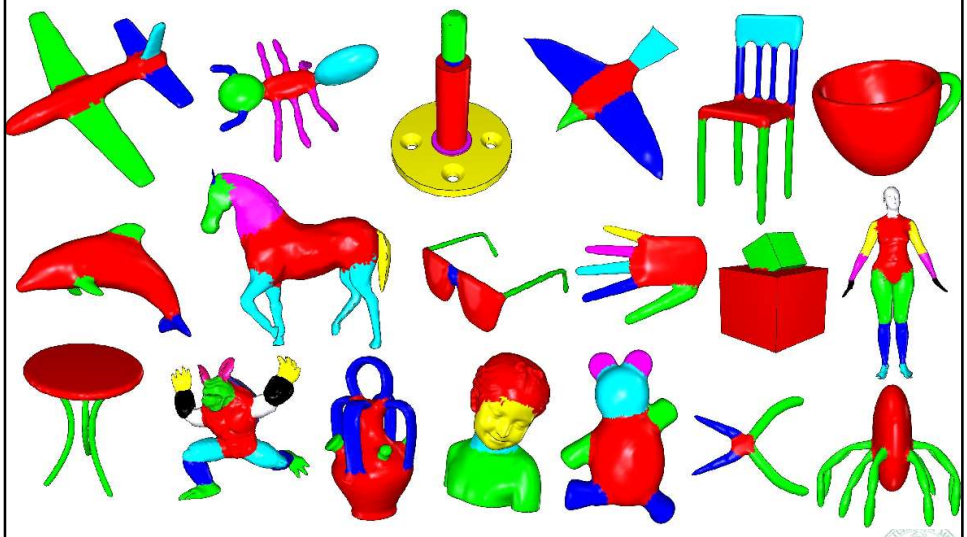
160

Automatic mesh segmentation / labelling



161

Automatic mesh segmentation / labelling



Img by Jiajun Lv, Xinlei Chen, Jin Huang, Hujun Bao

162

Automatic mesh segmentation / labelling

- ✓ Mesh segmentation (in generale):
  - ⇒ Data una mesh in input, identificare le zone semanticamente o strutturalmente distinte
  - ⇒ (tipicamente, come partizione delle facce o dei vertici in zone contigue)
  - ⇒ Guidati da un'analisi della geometria, oppure data driven
- ✓ Mesh labelling
  - ⇒ Assegnare un'etichetta semantica ad ogni partizione
- ✓ Utilizzato come punto di partenza di molti altri task

163

Shape Retrival

Img by Tamy Boubekeur, and Marc Alexa, SIGGR 2010

164

Shape Retrival

- ✓ Il task: data una grossa collezione di modelli 3D, individuare quelli simili ad una forma target richiesta
  - ⇒ Spesso, la forma target è specificata attraverso un disegno semplificato 2D
  - ⇒ Oppure: “trovare le mesh che rappresentano oggetti simili a quello rappresentato da una mesh target data”.

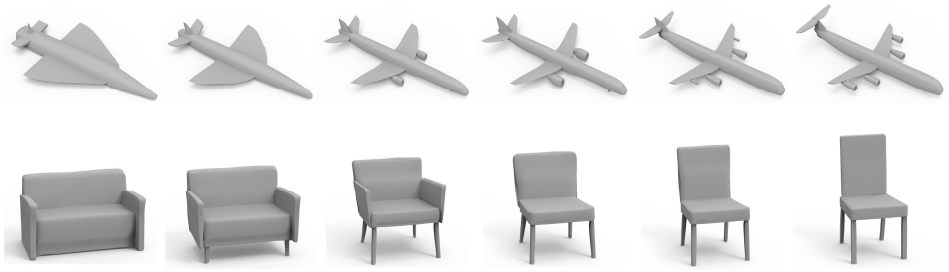
165

Procedural mesh generation

- ✓ Procedural mesh generation
  - ⇒ Nome generico per tutti quei procedimenti automatici che generano una mesh (geometria e connettività incluse)
  - ⇒ Utilizzato in videogames, VR, industria cinematografica, simulazioni…
  - ⇒ Molto studiati i casi di: piante, città, terreni, palazzi, manufatti di vario tipo, avatar umani, …
  - ⇒ Tipicamente, guidato da parametri controllati dall’utente che determinano le caratteristiche «ad alto livello» delle forme generate (ad esempio: nella generazione automatica di mesh che rappresentano ingranaggi, un parametro può identificare «il numero di denti») e / o da scelte pseudocasuali
- ✓ Geneative Networks:
  L’approccio Machine learning approach alla mesh generation
  - ⇒ Data driven
  - ⇒ Per es: data una collezione di mesh che rappresentano forme di una certa classe (sedie, aerei…), produrre un insieme di varianti

166

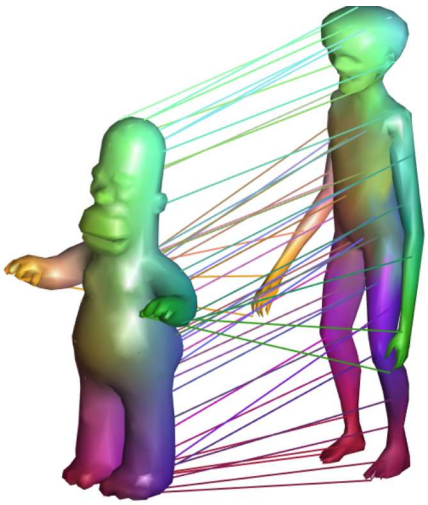
Generative Networks for Meshes



Img by Richard Zhang (2019)

167

Shape Correspondence



Img by Maks Ovsjanikov, Michael M. Bronstein, Emanuele Rodola

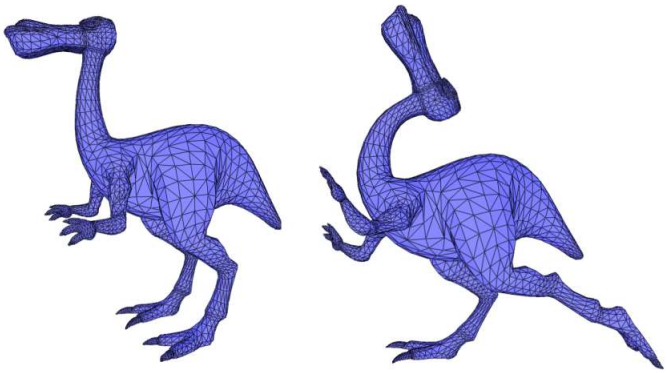
168

Shape correspondence

- ✓ Date due mesh che rappresentano oggetti di una stessa classe
  - ⇒ Per esempio, due forme umanoidi, o due quadrupedi
  - ⇒ Oppure, la scansione di uno stessa persona in due posizioni differenti…
- ✓ Identificare su ciascuna di essa il punto corrispondente sull'altra
  - ⇒ Ad esmepio, sotto forma di un mapping (non biunivoco) fra i vertici
- ✓ Basandosi su similitudine geometrica, topologica, e caratteristiche intrinseche della forma
  - ⇒ (ad esempio: la similarità di normale è poco significativa ma quella di curvatura intrinseca può esserlo di più)

169

Mesh Deformation



Img by Sébastien Loriot, Olga Sorkine-Hornung

170

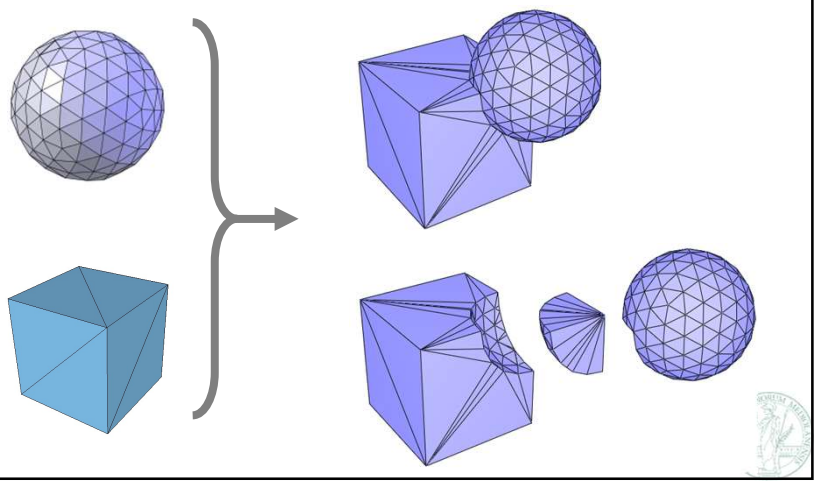
Mesh Deformation

- ✓ Data una mesh iniziale soggetta a degli stimoli o vincoli esterni, computare una sua deformazione spaziale
  - ⇒ cioè una nuova geometria, (la connettività della mesh è inalterata)
  - ⇒ La forma finale aderisce a prinicìpi fisici, geometrici (per es, conservazione dell'area, del volume, della forma dei triangoli, o semplicemente minimizzazione della distorsione subita dai singoli triangoli…)
  - ⇒ Esempio di vincolo esterno: una nuova la posizione xyz assegnata a solo alcuni vertici («questo vertice si sposta qui»)
- ✓ Utilizzata in animazione, design
  - ⇒ E' possibile effettuare questi computi in tempo reale

171

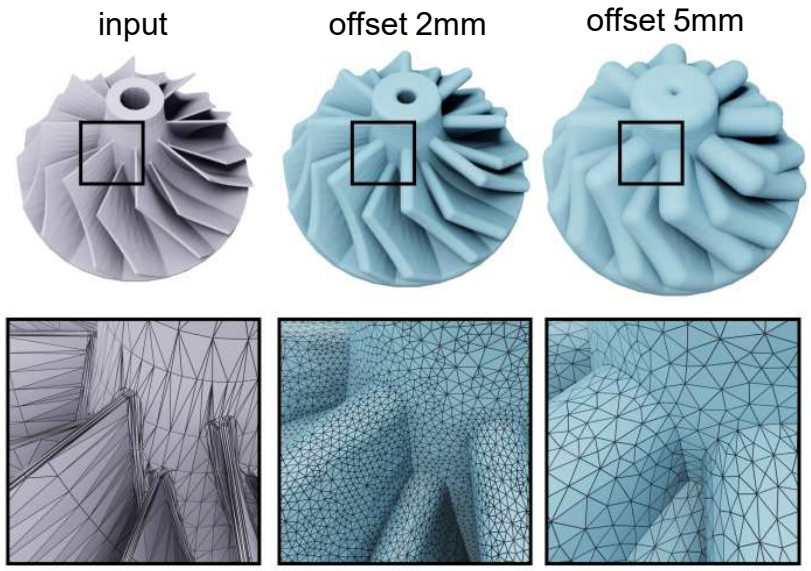
"Operazioni booleane" su mesh

Date due (o più) mesh **ben orientate** e **chiuse**,
(che quindi delimitano un volume interno ciascuna)
torvare la mesh che delimita l'intersezone ("AND"),
l'unione (lo "OR"), etc, fra i due volumi



172

Surface offsetting



[Image by P. Alliez et al, 2023]

(come abbiamo già visto nel caso delle nuvole di punti, in cui è più semplice)

174

Mesh processing: per saperne di più…

https://sgp2019.di.unimi.it/

Una conferenza internazionale su Geometry Processing che si è svolta nel nostro ateneo alcuni anni fa:

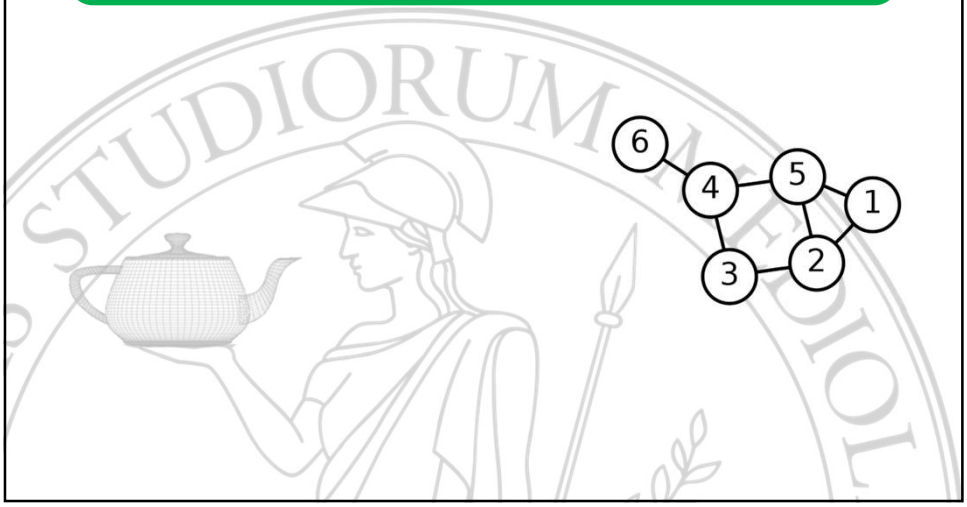
Graduate school: lezioni introduttive ad argoemnti avanzati di geometry processing

(i video sono ancora disponibili!)

175

Marco Tarini - Computer Graphics 2022/2023
Università degli Studi di Milano

Mesh Processing: algoritmi e strutture dati (cenni) 1/2



176

Mesh Processing: le basi algoritmche

- ✓ Gli esempi visti sono solo alcuni dei molti esempi di task affrontati nel contesto del mesh processing.
- ✓ Gli algoritmi che risolvono questi task necessitano (quasi tutti) operazioni base sulla connettività come per esempio:
  (operazioni di navigazione su mesh)
  - ⇒ Data una faccia, enumera tutte le facce adiacenti (separate da un edge)
  - ⇒ Dato una faccia ed un edge, trova la faccia (se esiste) che sta dall'altra faccia di quell'edge
  - ⇒ Dato un vertice, elenca tutte le facce che includono quel vertice (detta la «stella» o «stella-1» del vertice)
  - ⇒ Dato un vertice, enumera tutti i vertici che sono connessi a quel vertice da un edge
  - ⇒ Dato un edge, scopri se è un edge di bordo.
  - ⇒ Dato un edge di bordo, enumera tutti gli altri edge che fanno parte di quel bordo
  - ⇒ Dato una faccia, elenca tutti i vertici che fanno parte di quella faccia
- ✓ E' necessario che queste operazioni base siano effettuate efficientemente (idealmente, in tempo costante)

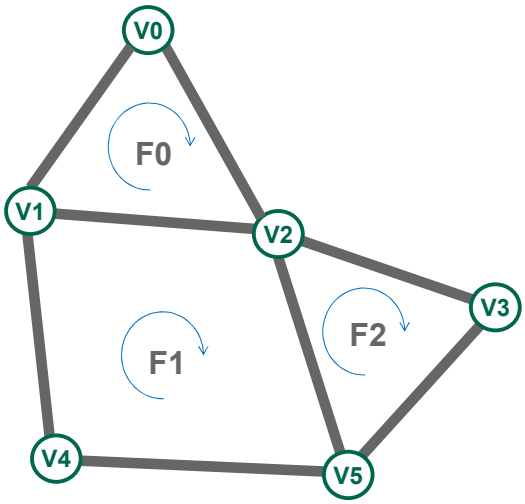
178

Strutture dati per Mesh Processing

- ✓ La struttura «lista facce» della mesh indexed, usata per memorizzare la connettività, non consente di effettuare queste operazioni
  - ⇒ (se non attraverso una scansione dell'intero vettore delle facce, che richiede ovviamente un tempo lineare col numero di facce)
  - ⇒ (eccetto una: «data una faccia, elenca tutti i vertici che fanno parte di quella faccia»)
- ✓ Per effettuare mesh processing, dobbiamo dotarci di strutture più adatte per memorizzare la connettività di una mesh
  - ⇒ svantaggio: più prolisse, onerose da mantenere durante le modifiche
  - ⇒ ma consentono di navigare sulla mesh molto più agevolmente
- ✓ Vediamo una di queste strutture

179

Struttura dati: connettività di una indexed mesh

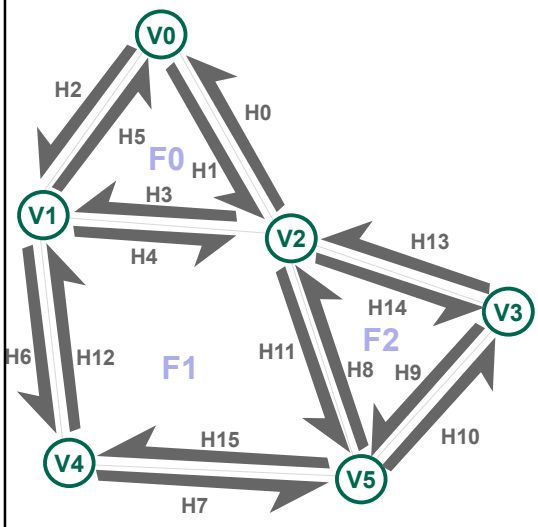


$F_0 \rightarrow$ 0 , 2 , 1
$F_1 \rightarrow$ 1 , 2 , 5 , 4
$F_2 \rightarrow$ 2 , 3 , 5

CONNETTIVITA'
(come lista facce)

180

Struttura dati: half edges



	V	F	Next	Opp
H_0 →	2	–	2	1
H_1 →	0	0	3	0
H_2 →	0	–	6	5
H_3 →	2	0	5	4
H_4 →	1	1	11	3
H_5 →	1	0	1	2
H_6 →	1	–	7	12
H_7 →	4	–	10	15
H_8 →	5	2	14	11
H_9 →	3	2	8	10
H_{10} →	5	–	13	9
H_{11} →	2	1	15	8
H_{12} →	4	1	4	6
H_{13} →	3	–	0	14
H_{14} →	2	2	9	13
H_{15} →	5	1	12	7

181

Lista di half edge

- ✓ La connettività è rappresentata da un vettore di Half Edge
- ✓ Per ogni half edge memorizzo i campi (sono tutti indici):
  - ⇒ Indice di Vertice: da quale vertice parte
  - ⇒ Indice di Faccia: di quale faccia è un bordo
  - ⇒ Next: l'indice dell'half-edge che incontro proseguendo nella direzione dell'half edge (senza cambiare faccia o bordo della mesh)
  - ⇒ Opposite: indice all'altro half-hege che condivide lo stesso edge
- ✓ Strutture a contorno:
  - ⇒ In ogni vertice, posso memorizzare l'indice di un Half-Edge che parte da quell vertice (uno qualsiasi)
  - ⇒ Posso avere un vettore di facce, per ogni faccia l'indice di un half edge appartenente a quella faccia (uno qualsiasi)

187

HalfEdge: pseudocodice Java

```
class HalfEdge {
    int vi;   // indice di vertice
    int fi;   // indice di faccia (o -1)
    int next; // indice di halfHedge
    int opp;  // indice di halfHedge
}

// la tabella
HalfEdge[] he = new HalfEdge( .... );
```

ed es, il valore *opposite* del halfedge di indice 12 è…

```
he[12].opp;
```

ed es, l'half edge di indice 7 fa parte della faccia…

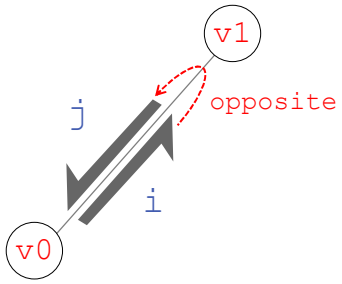
```
he[7].fi;
```

188

Esempio di uso base

- ✓ dato un halfedge di indice `i`, quali sono i due vertici `v0` e `v1` che delimitano l'edge corrispondente?

```
int v0 = he[i].vi;
int j = he[i].opp;
int v1 = he[j].vi;
```



189

Strutture dati per connettività a confronto

- ✓ Lista facce:
  - ⇒ Compatta
    (quindi, adatta a storing, ad esempio su disco o streaming)
  - ⇒ Sufficiente per ad alcuni task di processing
    (e in questo caso, preferibile)
  - ⇒ Il redering classico eseguito dalle GPU
    è pensato per questa struttura dati
  - ⇒ E' generale: non richiede ad esempio two-manifoldness o orientamento consistente delle faccie
    (quindi: capace di rappresentare strutture inconsistenti – è un vantaggio e uno svantaggio)
  - ⇒ Lista di elementi non omogenea, (alcune facce hanno un numero di vertici diverso da altre).
    eccetto che per tri-mesh o pure quad meshes: per loro, la lista facce è una matrice 3xN o 4xN (comodo)

190

Strutture dati per connettività a confronto

- ✓ Lista di half hedge:
  - ⇒ Prolissa
    (calcola: quanti interi è necessario memorizzare in media, rispetto ad una lista facce?
    Ipotesi: in una tri mesh, ho vertici , facce, edge tipicamente in proporzione 1x, 2x, 3x. In una quad mesh: 1x, 1x, 2x)
  - ⇒ Più complicata da mantenere coerente durante le operazioni di modifica della connettività
  - ⇒ Non adatta per il rendering su GPU
  - ⇒ Vantaggio: lista di elementi sempre omogenea: 4 elementi per half-edge (nella variante che abbiamo visto), anche su mesh poligonali miste
  - ⇒ Consente di «navigare sulla mesh», con salti all'elemento adiacente in tempo costante (consentendo algoritmi di mesh processing in tempo lineare o pseudolineare piuttosto che quadratico)
  - ⇒ Richiede adattamenti se la mesh non è two-manifold e ben orientata
- ✓ Nota: sono due rappresentazioni alternative di una stessa cosa (la connettività della mesh).
  - ⇒ Una si può costruire a partire dall'altra

191