Marco Tarini - Computer Graphics 2023/2024
Università degli Studi di Milano

3D Models: Polygonal Representations - Part 2

61

Attributi: il caso delle normali

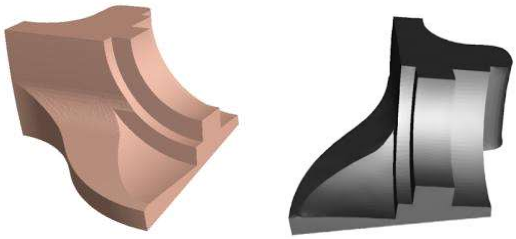
Nel caso in cui l'attributo è la normale
(cioè il *vettore unitario ortogonale alla superficie*)

- ✓ se memorizzato per faccia:
  - ⇒ la normale costante su ciascuna faccia
  - ⇒ facce (dall'aspetto) piatto
  - ⇒ discontinuità C0 sugli edge:
    edge sono «spigoli taglienti»,
    detti edge di crease, o «hard» edges
- ✓ se memorizzato per vertice
  - ⇒ normale che varia sulla faccia
  - ⇒ facce (dall'aspetto) curvo, in generale
  - ⇒ continuità C0: → aspetto «smooth» (liscio),
  - ⇒ (ma non continuità C1)

62

Normali di una superficie: osservazioni

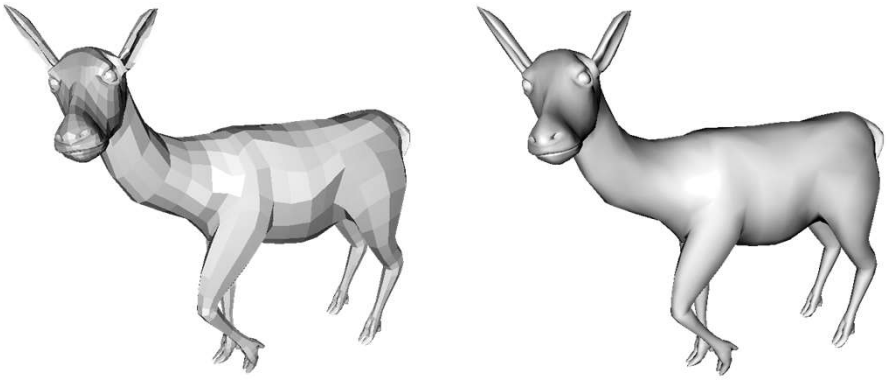
- ✓ Normali costanti => superficie piatta
- ✓ Normali che variano con continuità => superficie curva
- ✓ Normali che variano con discontinuità
  => superficie con un *crease*
  - ⇒ un *crease* è uno spigolo, punti dove la normale è discontinua
  - ⇒ anche detto *hard-edge,* o *line feature*



63

Attributi per faccia: normali

Normali per faccia | Normali per vertice

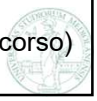


Nota: le normali sono rivelate all'occhio dall'*illuminazione*
(interazione con la luce, computata durante del rendering)

64

Normali come attributi per vertice di una mesh

- ✓ Normali definite per faccia:
  - ⇒ Le normali sono costanti sulle facce:
  - ⇒ Le facce appaiono piatte
  - ⇒ coerentemente col modello digitale, ma a differenza (spesso) dell'oggetto 3D che si intendeva rappresentare!
  - ⇒ Appaiono piccoli crease su ogni edge della mesh
- ✓ Normali definite per vertice:
  - ⇒ Le normali variano sulle facce (vengono interpolate dentro le facce, come qualsiasi altro attributo)
  - ⇒ Le facce appaiono in generale curve
  - ⇒ Come ottengo una faccia che appaia piatta? Uso una stessa normale come attributo dei tre vertici di un triangolo
  - ⇒ Come ottengo un crease (una discontinuità di normale)? (vedi next)
- ✓ Nota: le normali risultano visibili all'osservatore attraverso l'illuminazione del modello digitale
  - ⇒ Il calcolo dell'illuninazione è un task del rendering (2nda metà del corso)



65

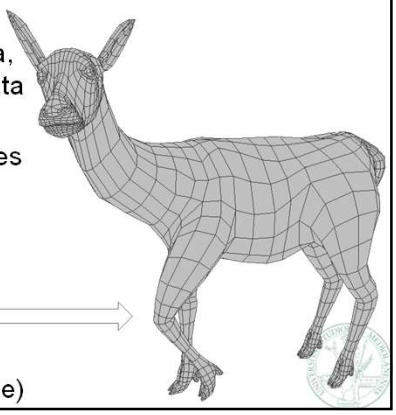
Interpolare l'attributo per vertice "vettore normale"

- ✓ Come abbiamo visto, interpolando vettori unitari si ottengono vettori non unitari
  - ⇒ Che vanno quindi ri-normalizzati
- ✓ Questo significa che gli attributi di tipo *vettore unitario*, come le normali, devono essere rinormalizzati dopo ogni l'interpolazione
- ✓ Esercizio: determinare con una stima se questo comporta un onere di calcolo eccessivo, per una GPU, durante un rendering in real-time
- ✓ Traccia
  - ⇒ stimare quante operazioni in virgola mobile (Floating Point Operation) sono necessarie per normalizzare un vettore (soluzione: circa 10)
  - ⇒ stimare quante volte sarà necessario ri-normalizzare un attributo «normale» in un rendering (stimiamo, a braccio: 1 volta in ogni pixel, quindi circa 1000x1000 = 10^6)
  - ⇒ frame per secondo, in un rendering in tempo reale: stimiamo 60
  - ⇒ totale **Fl**oating Point **Op**eration al **S**econdo (FLOPS) necessari a questo passaggio: 10 x 10^6 x 60 = 6x10^8 = 0.6 Giga-FLOPS
  - ⇒ Una GPU in commercio è capace di erogare… Tera-FLOPS = migliaia di Giga-FLOPS

66

Note su rendering delle mesh

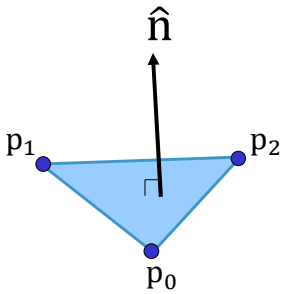
- ✓ L'uso di normali come attributo *per faccia* produce un tipo di rendering detto **flat shading**
  - ⇒ Perché le facce appaiono piatte (flat)
  - ⇒ Questo può essere utile per rivelare la forma poligonale delle mesh (ad esempio per poter giudicare la qualità della triangolazione)
- ✓ L'uso di normali come attributo *per vertice* produce un tipo di rendering detto **smooth shading**
  - ⇒ Perché la superficie appare liscia e curva, nascondendo la sua natura poligonalizzata
  - ⇒ Questo è di gran lunga in metodo più utilizzato, per esempio nei videogames
- ✓ Un tipo di rendering che rivela ancora più chiaramente la poligonalizzazione della mesh è detto **wireframe**
  - ⇒ Nel quale vengono disegnati esplicitamente anche gli edge (come linee)



67

Computo della normale di un triangolo

(Cioè del suo orientamento nello spazio)
(Vedi lezione nella marte matematica sul cross product)



due cosiddetti "edge vectors"

$$\vec{n} = (p_1 - p_0) \times (p_2 - p_0)$$

"area vector" definitp come un vettore ortogonale al triangolo, che è lungo quanto la (doppia) area del triangolo

normalizzazione

$$\hat{n} = \frac{\vec{n}}{\|\vec{n}\|}$$

"normale" del triangolo (un vett unitario)

E' anche la doppia area del triangolo

68

Calcolo delle normali per faccia di una mesh (note)

- ✓ La normale delle facce triangolari è data dal semplice computo visto sopra
  - ⇒ ripetuto su ciascuna faccia
- ✓ E' anche detta normale «geometrica» della faccia
  - ⇒ perché corrisponde effettivamente all'orientamento del piano su cui giace il triangolo (ed è costante)
  - ⇒ esiste sempre (nota: questo non vale per facce quads – a meno che non siano planari)
  - ⇒ un'eccezione: se ho triangoli «degeneri» (quelli con area 0) (con 3 vertici allineati o, 2 vertici coincidenti, etc) cosa succede se tento l'algoritmo qui sopra?
- ✓ La normale per faccia è orientata consistentemente (nello stesso verso) solo se la mesh è ben orientata
  - ⇒ (cosa succede altrimenti?)
  - ⇒ normale verso l'interno oppure l'esterno? Dipende della formula che uso per il suo computo (quali due edge-vector scelgo e in che ordine li moltiplico)

69

Calcolo delle normali per vertice di una mesh (note)

- ✓ La normale per vertice di una mesh non è definita in modo univoco per via geometrica
  - ⇒ quindi dobbiamo «inventarcene» una.
- ✓ La domanda che ci dobbiamo porre è:
  - ⇒ «immaginando che la mia mesh (che è composta da facce *piatte*) modelli invece una superficie reale *curva*, quale sarebbe la normale di questa superficie curva, su questo vertice?»
  - ⇒ Non esiste una risposta univoca!
  - ⇒ La risposta dipende da quale superficie si intende rappresentare.
- ✓ Strategia spesso utilizzata in pratica:
  - ⇒ usare una interpolazione delle facce adiacenti (per es, la media)
- ✓ Cioè: *le normali (per vertice) fanno parte del modello* (sono parte della descrizione della superficie) e spesso vengono costruite insieme al modello.
  - ⇒ Solo nei casi di una mesh sprovvista di normali, sarà necessario computarle dalla geometria (e dalla connettività)

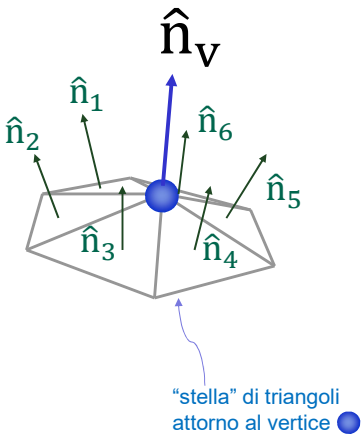
70

Normali come attributo per vertice

Un modo per calcolare di un vertice condiviso da k triangoli: la media delle k normali definite sui triangoli

k è detta la valenza del vertice, (nel disegno, k = 6)

$$\hat{n}_v = \frac{\hat{n}_1 + \cdots + \hat{n}_6}{\|\hat{n}_1 + \cdots + \hat{n}_6\|}$$



Nota: dato che si normalizza comunque il risultato della sommatoria, non c'è necessità di dividere per il numero di elementi sommati, come si fa di solito per la media

71

Algoritmo per computo di normale per vertice

Passo 1: computo delle normali per faccia triangolare

⇒ come prodotto cross dei due edge vectors della faccia, normalizzato

⇒ Adattare per mesh non triangolari

Passo 2: computo delle normali per vertice

✓ ∀ vertice **v**:
ciclo su tutte le facce adiacenti a **v**,
(cioè tutte le facce che annoverano **v** fra i propri vertici)
computo la somma di tutte le loro normali, e normalizzo il risultato
(per ottenere la normale di **v**)

Problema: come trovo «tutte le facce adiacenti ad un vertice dato»?

✓ Se scandisco l'intero vettore della «lista-facce» in cerca di facce che contengano **v**, il mio algoritmo diviene *quadratico*

⇒ se ho *k* vertici e 2*k* facce, richiede O(2*k*^2) operazioni!

⇒ nel mesh processing, gli algoritmi quadratici non sono accettabili (perché il numero *k* può essere molto grande)

Esiste invece un algoritmo efficiente (lineare) che usa solo la struttura «lista-facce» per la.

Sapresti identificare questo algoritmo?

72

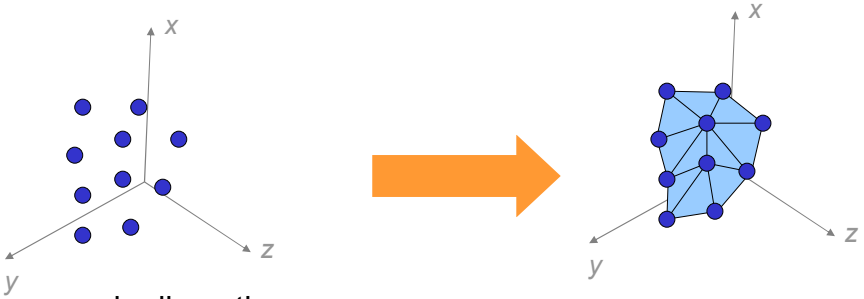
Algoritmo per computo di normale per vertice

- ✓ Passo 1 ∀ vertice: azzero il suo vettore normale
  - ⇒ Questo vettore servirà per mantenere le somme parziali
- ✓ Passo 2 ∀ faccia: calcolo la sua normale,
  e la sommo alla normale di ciascun vertice ai suoi corners
  - ⇒ Alla fine di questo ciclo, su ogni vertice avrò accumulato un vettori da ciascuna faccia adiacente a quel vertice
- ✓ Passo 3 ∀ vertice: normalizzo il risultato

- ✓ La strategia generale viene a volte detta scattering:
  - ⇒ Al passo 2, invece che raccogliere (*gather*) su ogni vertice le normali dalle facce adiacenti,
    distribuisco (*scatter*) da ogni faccia le normali ai vertici adiacenti
  - ⇒ La strategia dell'algoritmo precedente viene detta *gathering*
- ✓ Anche se il risultato è lo stesso del precedente, questo algoritmo è lineare:
  - ⇒ Se ho k vertici e 2k facce, mi ci vogliono solo ~ k + 2k + k ∈ O(n) operazioni
  - ⇒ Nel geometry processing, gli algoritmi lineari sono *feasible*: sono veloci anche per mesh a risoluzione molto grande (per es, k = 10^9)

73

Da Nuvola di punti a Triangle Mesh

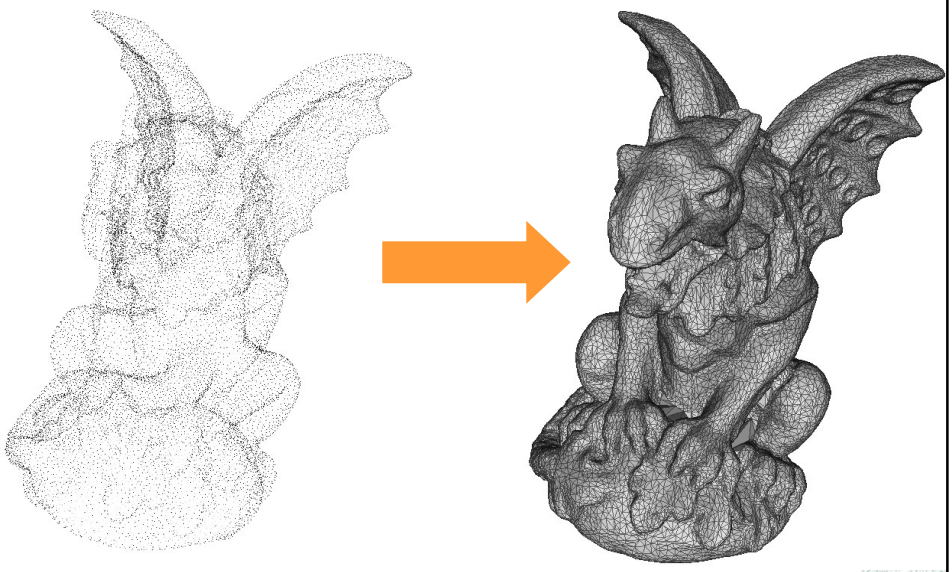


nuvola di punti

triangoli

74

Da Nuvola di punti a Triangle Mesh

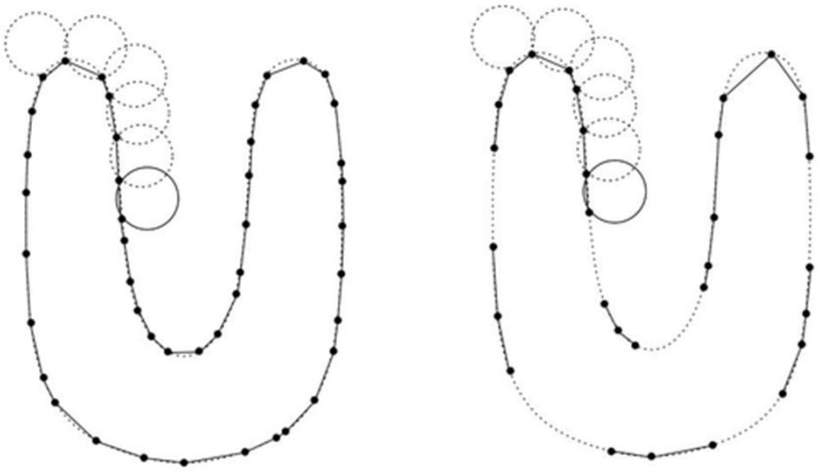


75

Da Nuvola di punti a Triangle Mesh

✓ Una classe di algoritmi: Front Advancing:

⇒Come per es «ball pivoting»



[Bernardini, Mittleman, Rushmeier, Silva TVCG 99]

76

Da Nuvola di punti a Triangle Mesh… in 2D

- ✓ In 2D: il problema ha un'ottima soluzione: la triangolazione di Delaunay
  - ⇒ Ben nota dagli anni '30 (in geom. computazionale)

Vertici (sul piano x,y)

77

Da Nuvola di punti a Triangle Mesh… in 2D

- ✓ In 2D: il problema ha un'ottima soluzione: la triangolazione di Delaunay
  - ⇒ Ben nota dagli anni '30 (in geom. computazionale)

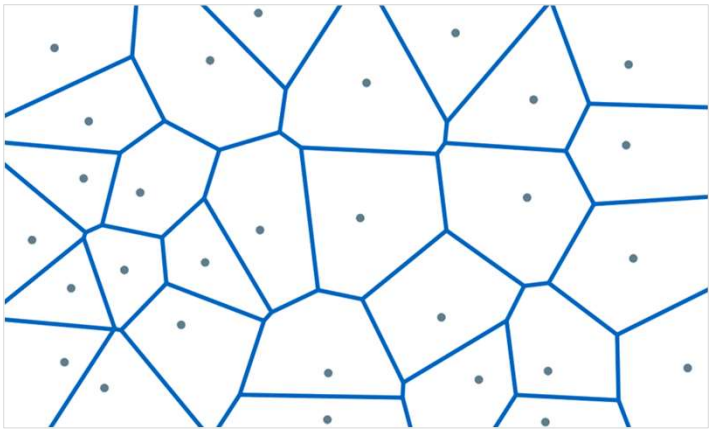
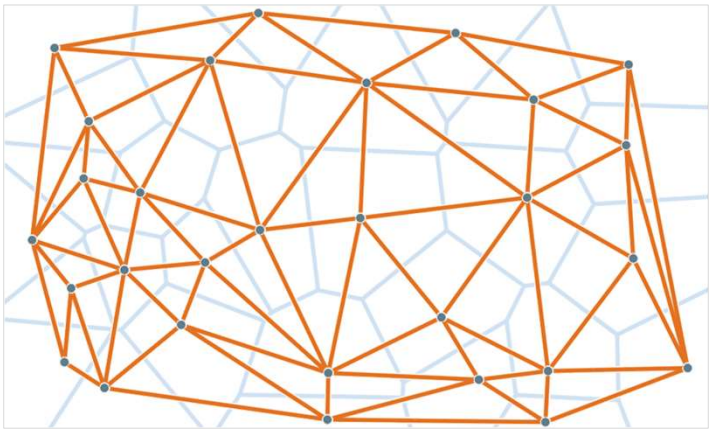


Diagramma di Voronoi

78

Da Nuvola di punti a Triangle Mesh… in 2D

- ✓ In 2D: il problema ha un'ottima soluzione: la triangolazione di Delaunay
  - ⇒ Ben nota dagli anni '30 (in geom. computazionale)

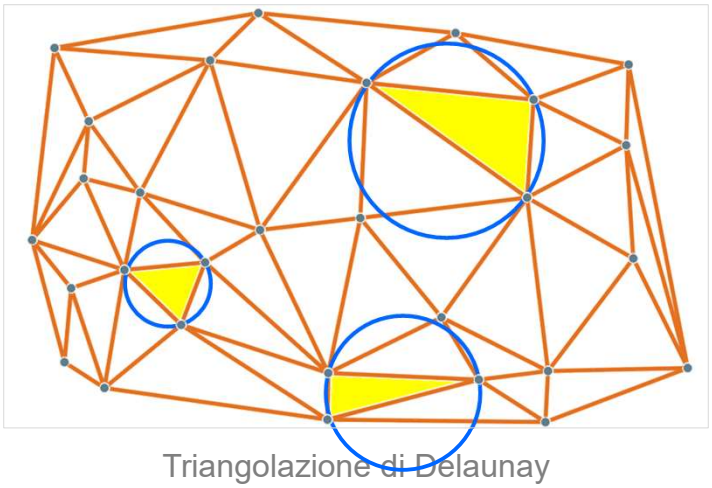


Triangolazione di Delaunay

79

Da Nuvola di punti a Triangle Mesh… in 2D

- ✓ E', una «buona» triangolazione
  - ⇒ Una buona proprietà è garantita:



Triangolazione di Delaunay

80

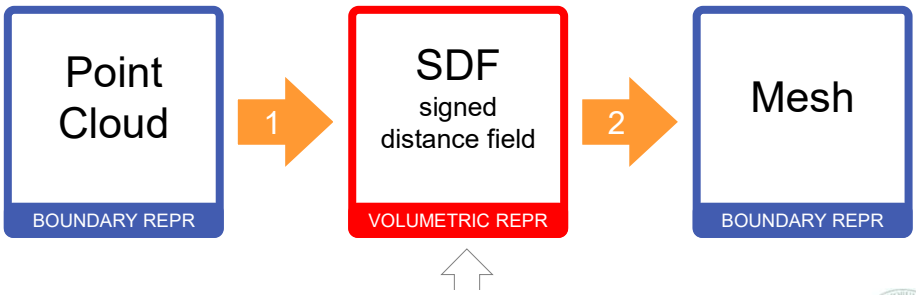
Triangolazione di Delaunay: note

- ✓ Diagramma di Voronoi (in 2D)
  - ⇒ Una scomposizione in regioni di piano indotta da n punti (detti "seed") $\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2,$
  - ⇒ Ogni seed produce una regione
  - ⇒ La regione di un seed $\mathbf{s}_i$ = l'insieme di punti $\mathbf{p}$ del piano che sono più vicini a $\mathbf{s}_i$ che ad ogni altro seed
- ✓ Triangolazione di Delaunay
  - ⇒ Il "duale" di un diagramma di Voronoi cioè:
  - ⇒ La connettività ottenuta connettendo con un edge ogni coppia di seed di regioni confinanti fra loro
  - ⇒ E' una triangolazione con ottime proprietà, come ad esempio: il circocentro che inscrive ogni triangolo non include nessun altro vertice

82

Da Nuvola di punti a Triangle Mesh

- ✓ Modo più diffuso (anticipazione): passa attraverso una *rappresentazione intermedia volumetrica*
  - ⇒ Vedi lezione su dati volumetrici

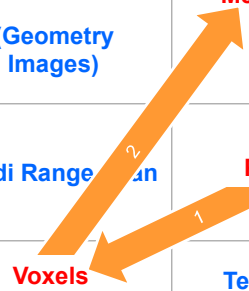


Vedremo nella lezione su dataset volumetrici.

83

Da Nuvola di punti a Triangle Mesh passando per…

		ELEMENTI DISCRETI			CONTINUI
		regolari *«a griglia»*	semi-regolari o irregolari: elementi simpliciali	semi-regolari o irregolari: elementi non simpliciali	
SUPERFICIALI	2-manifold *«rappresenta una vera superficie»*	**Height Field** **Range Scan** **(Geometry Images)**	**Triangle Mesh**	**Polygonal Mesh** **Quad-Mesh** **Quad dominant Mesh**	**Subdivision surface** **Parametric Surface** **(es. B-splines)**
	non-manifold *«non rappresenta una sup»*	**Set di Range…**	**Point Cloud**		
VOLUMETRICI	(3-manifold)	**Voxels** **Solid Textures**	**Tetra Mesh**	**Hexa Mesh**	**Implicit model** **(es. CSG)**



84

Da Nuvola di punti a Triangle Mesh: sommario

- ✓ Alcuni metodi aggiungono solo la connettività
  - ⇒ la nuvola di punti costituisce già la geometria della mesh (anche se è possibile scartarne alcuni)
  - ⇒ Front Advancing, come Ball Pivoting
  - ⇒ In 2D: Delaunay triangulation (da Voronoi diagram) (possiamo estendere a 3D se i punti vengono prima proiettati su un piano)
  - ⇒ Un problema comune è che i vertici della mesh mantengono il rumore della nuvola di punti
- ✓ Altri metodi ricampionano i vertici da zero
  - ⇒ Come "Poisson Reconstruction"
  - ⇒ Uso di rappresentazioni volumetriche intermedie
  - ⇒ Vedremo nella lez. su rappresentazioni volumetriche
  - ⇒ In pratica, questi metodi consentono di attenuare fortemente i problemi dovuti al noise e agli autliers della nuvola di punti

85