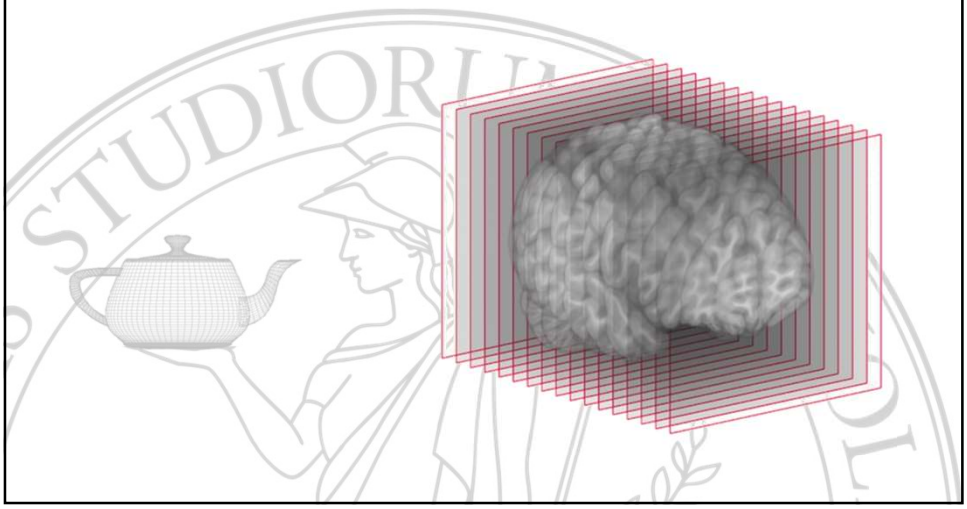
Marco Tarini - Computer Graphics 2022/2023
Università degli Studi di Milano

Rappresentazioni Volumetriche



1

Una (imperfetta) categorizzazione dei tipi di modelli digitali 3D

		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 surfaces** **Parametric Surfaces (es. B-splines)**
	non-manifold *«non rappresenta una sup»*	**Set di Range Scan**	**Point Cloud**		
VOLUMETRICI	(3-manifold)	**Voxelized Volume** **Volumetric Textures**	**Tetra Mesh**	**Hexa Mesh**	**Implicit models (es. CSG)**

2

Modelli 3D Volumetrici

1. Discreti & irregolari: mesh poliedrali
   ⇒analogo di una mesh poligonale (ma nel volume)
   ⇒insieme di poliedri adiacenti faccia a faccia

2. Discreti & regolari: dataset voxellizati
   ⇒analogo di un immagine rasterizzata, ma in 3D
   ⇒una griglia di voxel

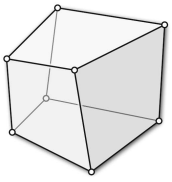
3. Continui: modelli impliciti
   ⇒rappresentazione basata su funzioni volumetriche
   ⇒superficie come luogo di zeri di una funzione

4

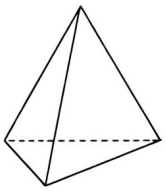
Mesh poliedrale

✓Corrispondente volumetrico delle mesh poligonali

✓Composta da poliedri quali…



esaedro
(o “hexa” per brevità)

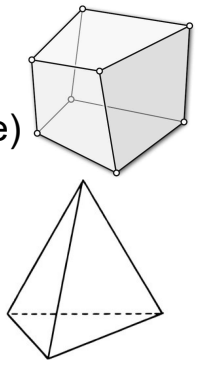


tetraedro
(o “tetra” per brevità)

5

Modelli 3D volumetrici ad **elementi finiti**

- ✓Tipo degli elementi:
  - ⇒hexahedra (anche detti “cuboidi”)
  - ⇒tetrahedra (piramidi a base triangolare)
  - ⇒poliedri generici (raro)



- ✓mesh poliedrale = mesh composta da elementi poliedrici adiacenti faccia a faccia
  - ⇒hexahedron mesh, o hexa-mesh
  - ⇒tetrahedron mesh tetra-mesh
- ✓esempio di visualizzatore: www.hexalab.net

6

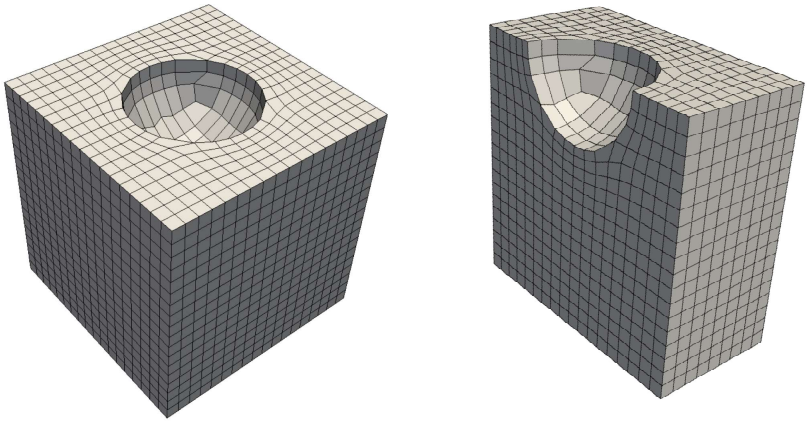
Mesh poliedrale

- ✓Composta da
  - ⇒geometria:
    - vertici (0D), con pos (x,y,z)
  - ⇒connettività:
    - poliedri (3D)
    - facce (2D)
    - edge (1D)

    che connettono i vertici
  - ⇒attributi
    - sui vertici,
    - Implicitamente inteprolati dentro gli elementi
- ✓Struttura dati: simile alla mesh poligonale
  - ⇒indicizzata:
    - lista vertici,
    - lista poliedri
  - ⇒Esistono anche varianti per mesh poliedrali di struttura basata su half-edge

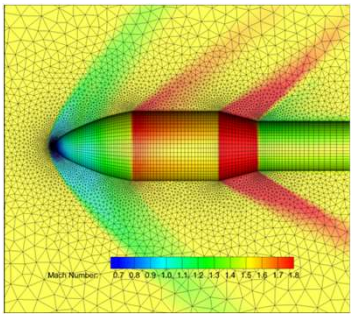
7

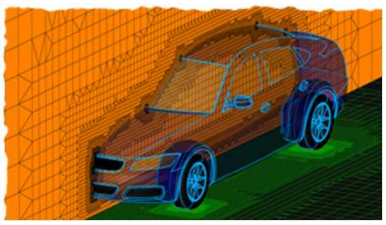
Hexa-mesh



8

context: FEA (Finite Element Analysis)

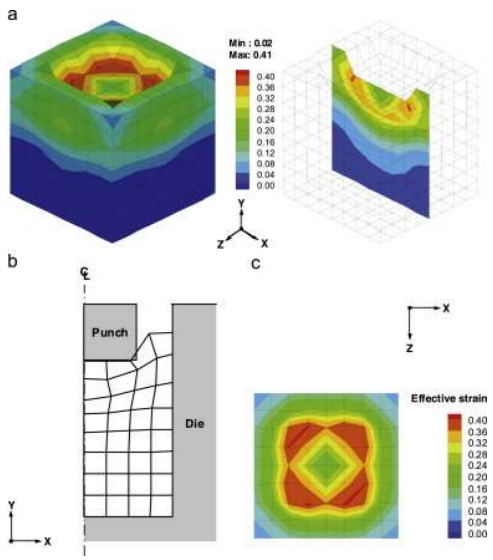




source: pointwise.com

9

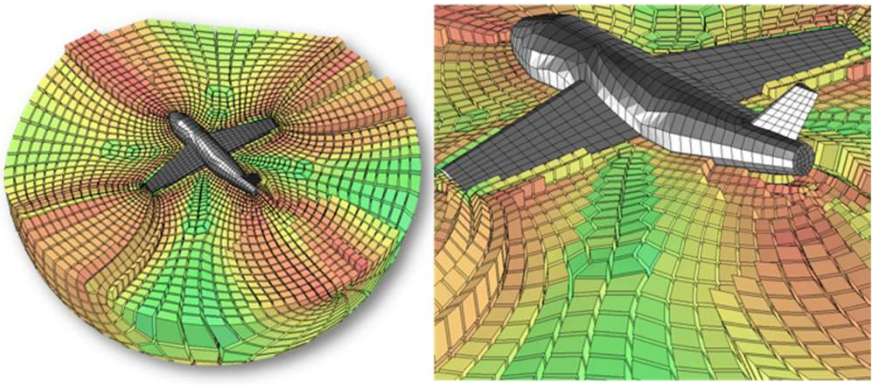
context: FEA (Finite Element Analysis) simulation



Soure: M.C.Lee et al: "3D simulation of forging"
in: Finite Elements in Analysis and Design journal

12

FEA simulation



Source: [Ruiz-Girones Roca Sarrate 2012]

13

Uso tipico: simulazioni fisiche

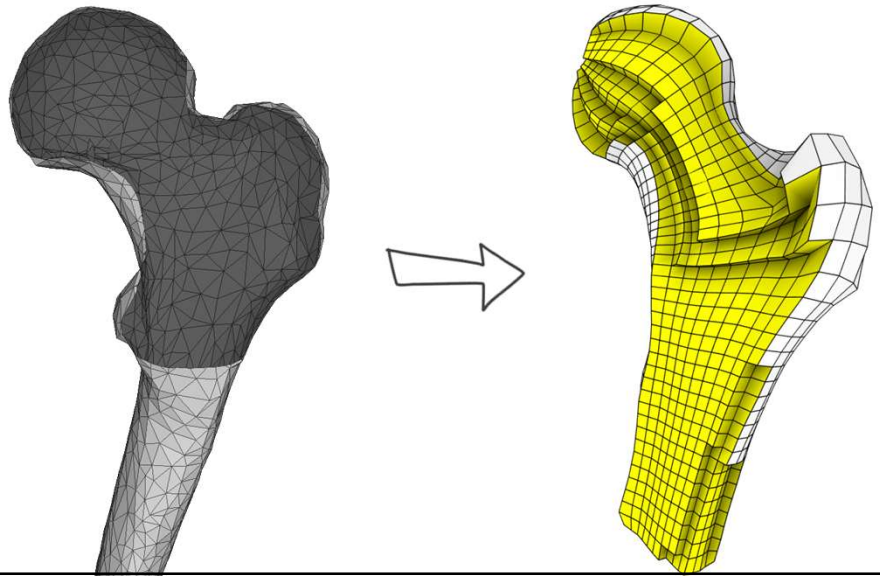
- ✓ FEM / FEA
  (Finite Element Method /
  Finite Element Analysis)
  - ⇒ Usata in ingegneria per verificare virtualmente le proprietà strutturali degli oggetti rappresentati
  - ⇒ Esempio: simulazione di carico:
    questo palazzo sostiene il suo peso?
    questo ponte sostiene il suo carico?
  - ⇒ Esempio: simulazione di termodinamica:
    come si diffonde il calore all'interno di questo oggetto?
  - ⇒ Simulazione dinamica o statica
- ✓ Le simulazioni sono il principale uso delle mesh poliedrali
- ✓ Problema (difficile): hexa-meshing o tetra-meshing:
  costuire una hexa-mesh o tetra-mesh
  a partire da una rappresentazione superficiale
  - ⇒ tipicamente, da una mesh poligonale (es: triangolare)

14

Polyhedral-Mesh construction

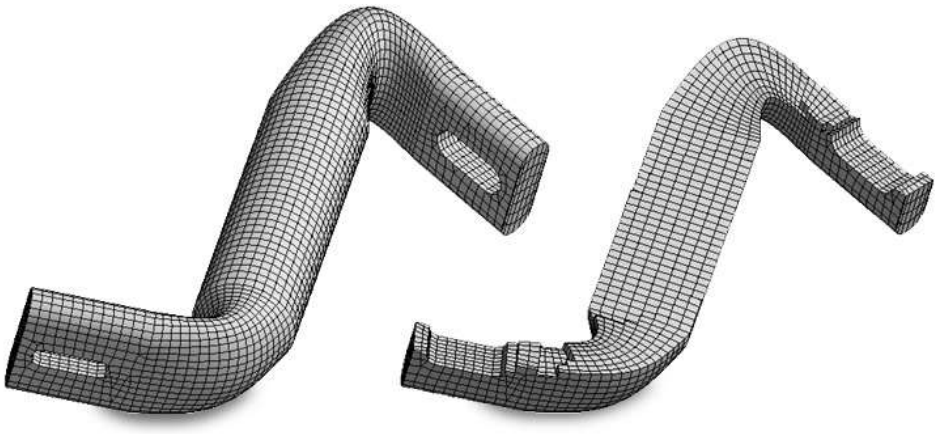
INPUT:
una mesh superficiale
(es: una tri mesh) M

OUTPUT:
una mesh poliedrale (es: una hexa-mesh)
il cui bordo approssima (o è) M



20

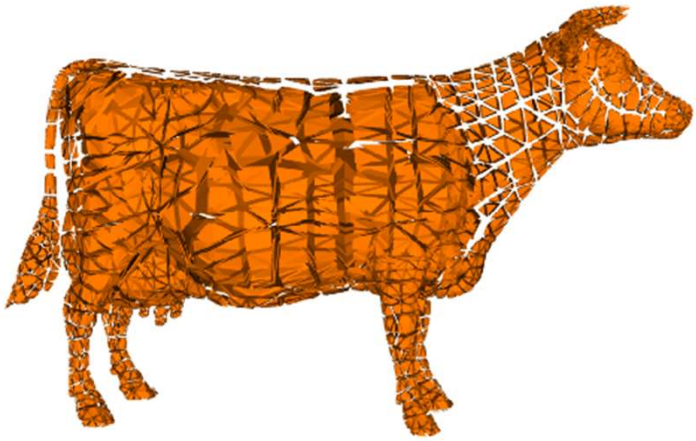
Hexahedron Mesh (Hex-mesh, hexa-mesh)



vedere esempi attraverso il visualizzatore
di hexa mesh online: www.hexalab.net

21

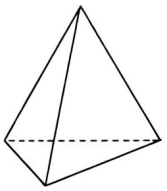
Tetra Mesh



22

Tetraedri

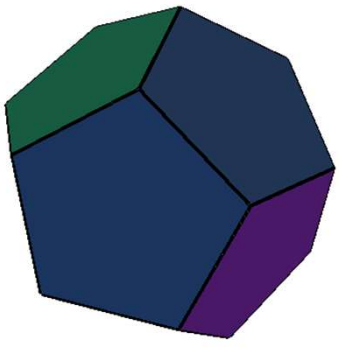
- ✓ Tetraedro: struttura *simpliciale* del volume
  - ⇒ come il triangolo è lo è della superficie.
  - ⇒ cioè: un tetraedro è il luogo di punti che sono l'interpolazione lineare fra i suoi quattro vertici
  - ⇒ cioè: ogni punto P dentro un tetraedro T (superficie compresa) è esprimibile come una (e una sola) combinazione lineare dei quattro vertici di T
  - ⇒ i 4 pesi di questa combinazione (quattro scalari) sono detti le coordinate baricentriche di P dentro T
  - ⇒ posso usare le coordinate baricentriche per interpolare fra gli attributi definiti sui vertici di T
- ✓ In tutto questo: tetra-mesh (volume) del tutto analoga alla tri-mesh (superficie)

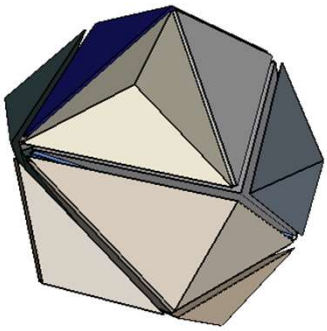


23

Tetrahedralization

- ✓ Ogni poliedro può essere scomposto in tetraedri
  - ⇒ Così come ogni poligono in triangoli

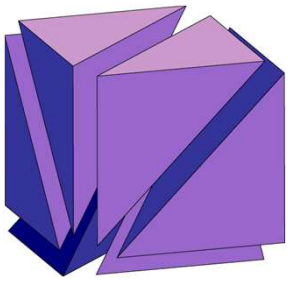




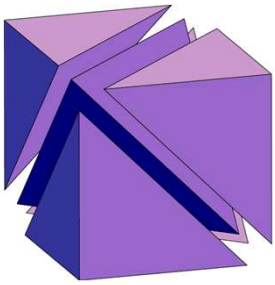
24

Tetrahedralization

- ✓ Ogni poliedro può essere scomposto in tetraedri
  - ⇒…in modi diversi
  - ⇒un hexa, per esempio, ammette almeno due scomposizioni in tetraedri…



12 tetra



5 tetra

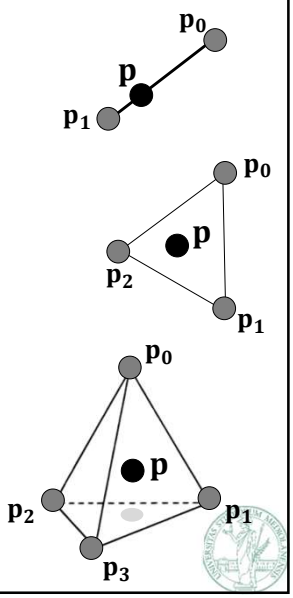
25

Computo delle «coordinate baricentriche» dentro un qualsiasi elemento *simpliciale*

- ✓ E' lo stesso problema in tutte le dimensioni con la stessa soluzione!
- ✓ Dato un tetraedro (o un triangolo, o un segmento) costituito dai suoi 4 (o 3, o 2) vertici $\mathbf{p}_0 \dots \mathbf{p}_n$ e un punto $\mathbf{p}$ al suo interno, trovare le 4 (o 3,o 2) «coordinate baricentriche» di $\mathbf{p}$ dentro a quel tetraedro (o tri, o sec)
- ✓ cioè i 4 (o 3, o 2) valori scalari $t_0 \dots t_n$ tali che

$$\mathbf{p} = \sum_i t_i \ \mathbf{p}_i \qquad \sum_i t_i = 1 \qquad \forall i : \ 0 \leq t_i \leq 1$$

- ✓ Fatto questo, il valore dell'attributo a in $\mathbf{p}$ sarà così dato dalla stessa combinazione lineare degli attributi $a_0 \dots a_n$ definiti sui vertici : $a = \sum_i t_i \ a_i$

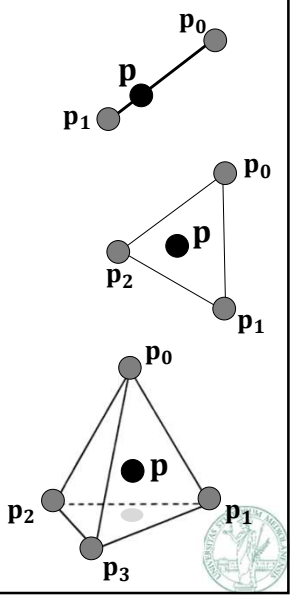


26

Computo delle «coordinate baricentriche» dentro un qualsiasi elemento *simpliciale*

- ✓ Schema generale della soluzione
1. Unire i 4 (3, 2) vertici $\mathbf{p}_0 \ldots \mathbf{p}_n$ al punto $\mathbf{p}$
2. Hai ottenuto 4 (3,2) nuovi sotto-tetraedri (-triangoli, -segmenti) che scompongono il tetraedro (triangolo, segmento) originale
3. Calcola l'estensione (cioè il volume, o l'area, o la lunghezza) di questi nuovi sotto-elementi
4. La coordinata baricentrica t_i è data dall' estensione dell'elemento opposto al vertice $\mathbf{p}_i$ diviso la somma delle estensioni (cioè diviso l'estensione dell'elemento originale)

- ✓ Esercizio: scrivere la formula nei tre casi, ipotizzando di avere una funzione volume(p0,p1,p2,p3) che restituisce il volume di un tetraedro di 4 vertici dati



27

Tetra meshes o Hexa Meshes?

- ✓ Risoluzione di una mesh poliedrale: n. di poliedri (o di vertici)
  - ⇒ maggiore risoluzione: simulazioni più accurate ma più lente
  - ⇒ nota: numero di elementi è CUBICO con 1/dimensione lineare
- ✓ Può essere adattiva (e spesso lo è)
- ✓ Multi-risoluzione: piramidi di livello di dettaglio sono possibili
  - ⇒ similmente alle mesh poligonali
- ✓ Categorie, simili a mesh poligonali:
  - ⇒ «Pure» hexa-mesh: solo elementi hexa
  - ⇒ «Hexa-dominant» mesh: grande maggioranza di elementi Hexa
- ✓ Esiste un concetto di regolarità locale anche per le hexa meshes / tri meshes
  - ⇒ analogo a quello delle alle mesh, ma definito sugli edge: un edge di una hexa mesh è regolare sse è condiviso da 4 hexa un edge di una tetra mesh è regolare sse è condiviso da 6 tetra
  - ⇒ mesh semiregolare: maggioranza di edge regolari.

29

Tetra meshes o Hexa Meshes

- ✓ Per poter essere utilizzata in una simulazione, una hexa mesh / tetra mesh deve avere elementi di buona «qualità», cioè (semplificando), la loro forma deve essere lontana dall'essere degenere (cioè piatta o, peggio, concava)
  - ⇒ la generazione automatica di mesh poliedrali con questa caratteristica è un problema aperto e difficile
  - ⇒ la generazione è fatta a partire da una struttura superficiale, es una mesh polignonale (deve essere two-manifold, chiusa, ben orientata)
- ✓ Hexa meshes:
  - ⇒ più difficile da costruire
  - ⇒ ma le simulazioni su hexa mesh sono più efficienti (a parità di risoluzione) o più accurate (a parità di tempo di esecuzione)
  - ⇒ recentemente, questo assunto è stato messo in discussione da alcuni risultati teorici di ricerca, che sembrano negare i vantaggi delle hexa-mesh

30

Una (imperfetta) categorizzazione dei tipi di modelli digitali 3D

		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 surfaces**, **Parametric Surfaces** (es. B-splines)
	non-manifold *«non rappresenta una sup»*	**Set di Range Scan**	**Point Cloud**		
VOLUMETRICI	(3-manifold)	**Voxelized Volume**, **Volumetric Textures**	**Tetra Mesh**	**Hexa Mesh**	**Implicit models** (es. CSG)

31

Modelli 3D Volumetrici

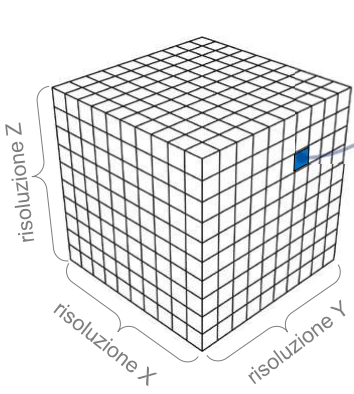
1. Discreti & irregolari: mesh poliedrali
   ⇒Tetra-mesh, hexa-mesh
   ⇒insieme di poliedri adiacenti faccia a faccia

2. Discreti & regolari: dataset voxellizati
   ⇒analogo di un immagine rasterizzata, ma in 3D
   ⇒una griglia 3D regolare di voxel

3. Continui: modelli impliciti
   ⇒rappresentazione basata su funzioni volumetriche
   ⇒superficie: luogo di zeri di questa funzione

32

Modello 3D a voxel (o voxellizato)



Griglia regolare 3D
(o lattice)

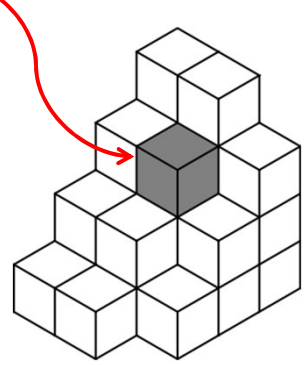
consumo memoria *cubico* con la risoluzione
(diventa facilmente ingestibile)

```
array [RES_X] [RES_Y] [RES_Z] of Voxels
```

33

Modelli Voxellizati

- ✓"Voxel" = Volume element
  - ⇒Così come…
    - "Pixel" = Picture Element
    - "Texel" = Texture Element
- ✓Elemento di una griglia regolare 3D
  - ⇒che è anche detta un lattice



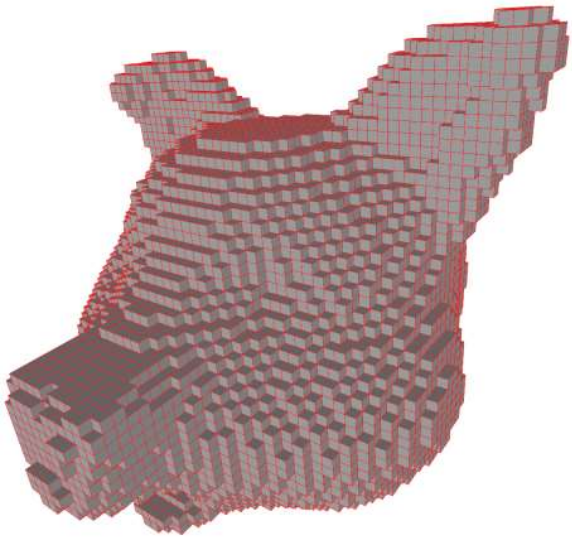
- ✓In codice:

```
Voxel[][][] data = new Voxel[resX][resY][resZ];
```

Esempio in Java

34

In questo caso, 1 Voxel = 1 Boolean (1 bit)

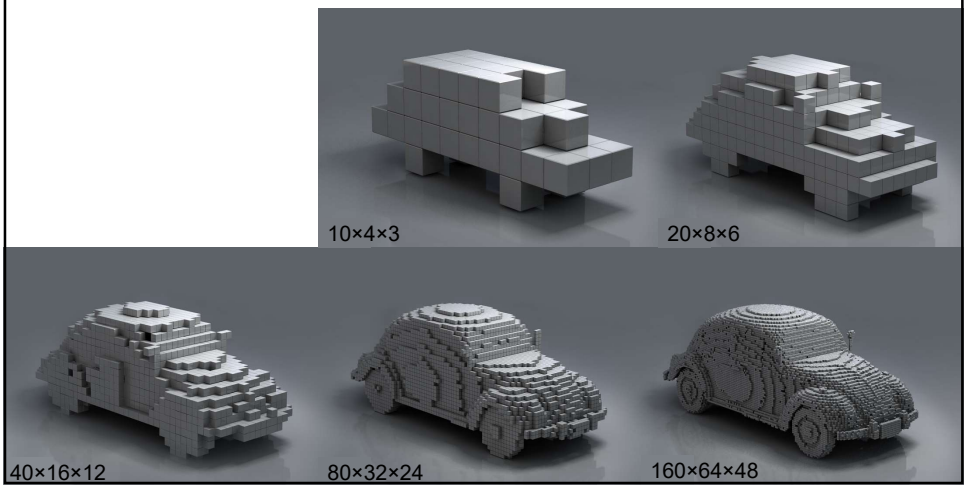


ogni *voxel* è pieno (1) o vuoto (0)

35

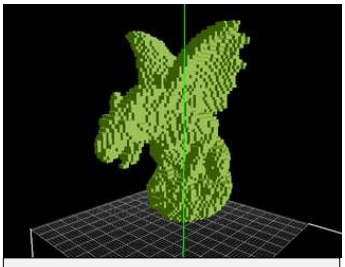
Risoluzione e costo in memoria

✓risoluzione: un intero per lato res = (X,Y,Z)



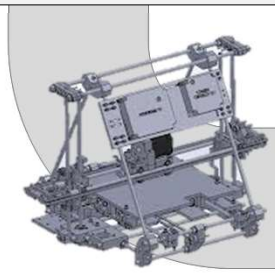
36

Voxel = 1 bit (pieno / vuoto)



Voxelized dataset
Voxel: pieno/vuoto

✓Input naturale di (alcuni)
3D printing devices
⇒Rapid prototyping devices per stampa “additiva”



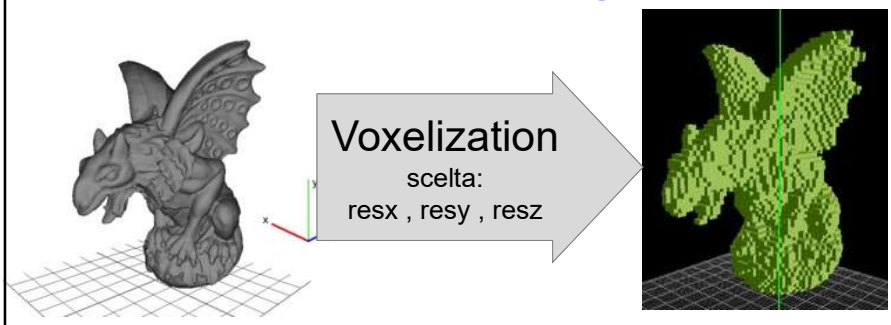
3D printer

oggetti stampati

37

Voxelized Volumes:

- ✓Input naturale dei **3D printing devices**



Tri Mesh

ben orientata
e chiusa!

Voxelized volume
– 1bit × voxel

to printer

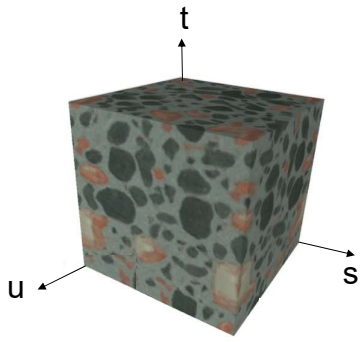
38

Occupazione spaziale dei dataset voxelizzati

- ✓Lo spazio è cubico con la risoluzione (linare)
- ✓E' di solito un prezzo troppo alto
  - ⇒Es: 1024^3 voxel = 1 gigavoxel
  - ⇒Molto oneroso, persino nel caso, come abbiamo visto fin'ora, di 1 solo bit per voxel (1 = pieno / 0 = vuoto)
  - ⇒Quando si memorizza 1 byte, 1 float, 1 double, 1 colore… etc, la situazione peggiora
- ✓Detta la «curse of dimensionality»

40

Se Voxel = 1 colore RGB: “texture volumetriche”



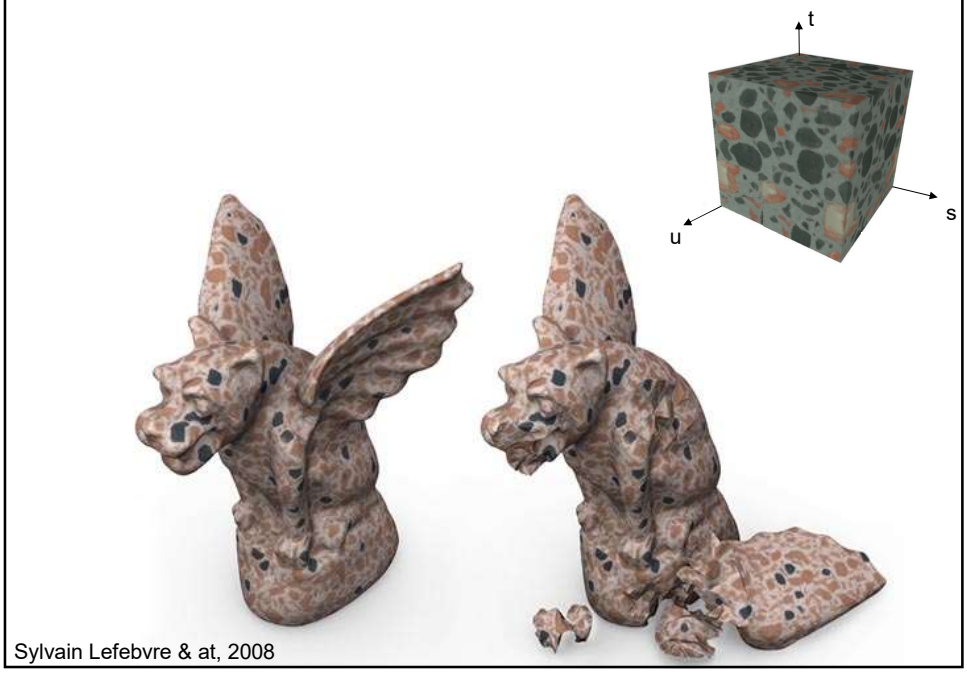
43

Volumetric Textures (o “solid Textures”)

- ✓ 1 texel = 1 voxel
  - ⇒esempio, solid RGB textures: 1 texel = 1 RGB color
- ✓ E’ supportata dall’Hardware, come ogni altra tessitura:
  - ⇒occupa la RAM della scheda video
  - ⇒accesso HW accelerato durante il rendering
  - ⇒interpolazione **tri**-lineare durante l’accesso …
- ✓ Modella il segnale (es. il colore) *dentro* al volume
  - ⇒per es: come gli oggetti sono colorati all’interno
  - ⇒utile per modelli che si possono rompere
  - ⇒utile per pattern come legno, marmo…
- ✓ Non richiede alcuna parametrizzazione della superficie!
  - ⇒La tessitura viene indicizzata dalle posizioni dei vertici
- ⚠ Solito problema, occupazione di memoria
  - ⇒es: quanto per 1 tessitura 1024^3 8-bits-per-channel RGBA?
  - ⇒es: quanto per 1 tessitura 265^3 8-bits-per-channel RGBA?

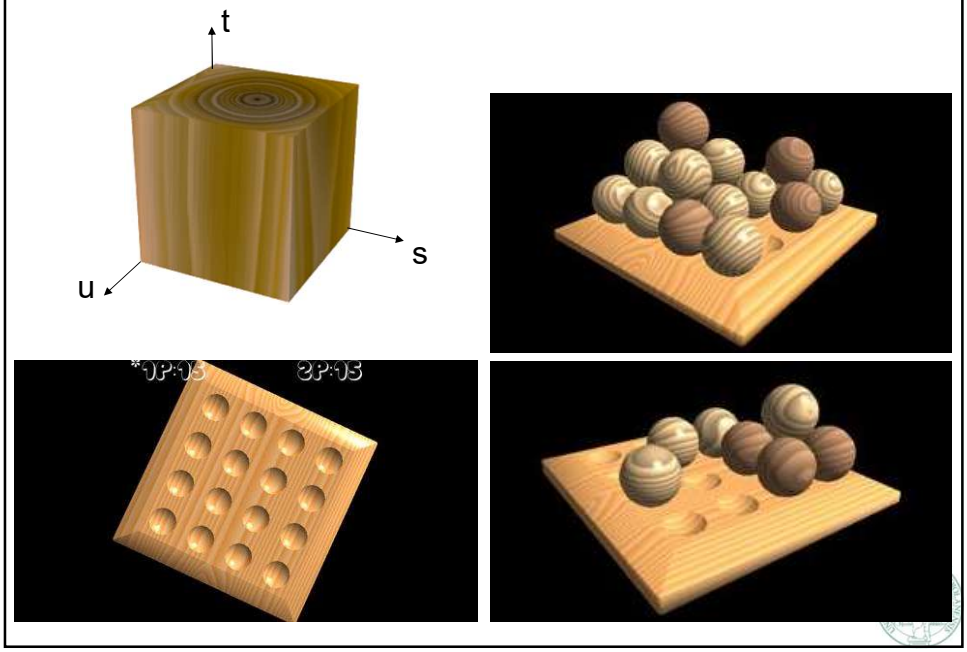
44

Un *colore* per voxel: “texture volumetriche”



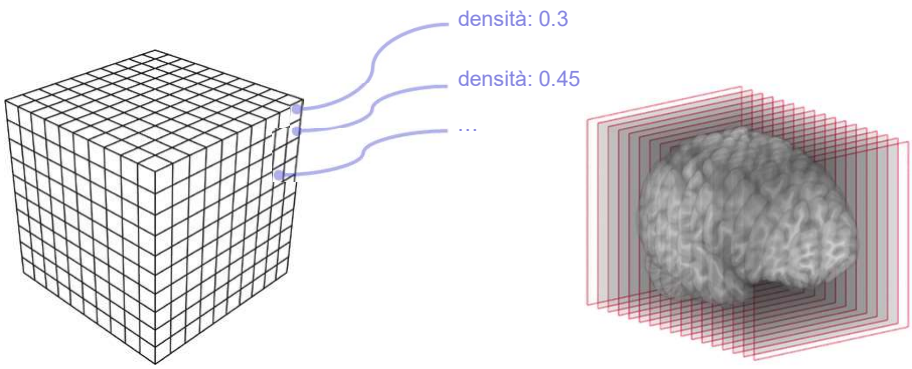
45

Volumetric Textures (o “solid Textures”)



46

Voxel = 1 scalare (es fra 0.0 e 1.0)

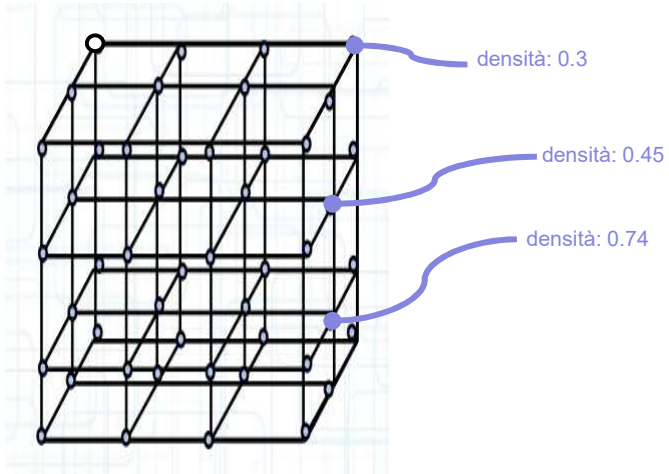


Esempio di File format: **DICOM**
(medicina)

```
Volume float [RES_X] [RES_Y] [RES_Z]
```

47

Voxelized models: uno scalare per voxel

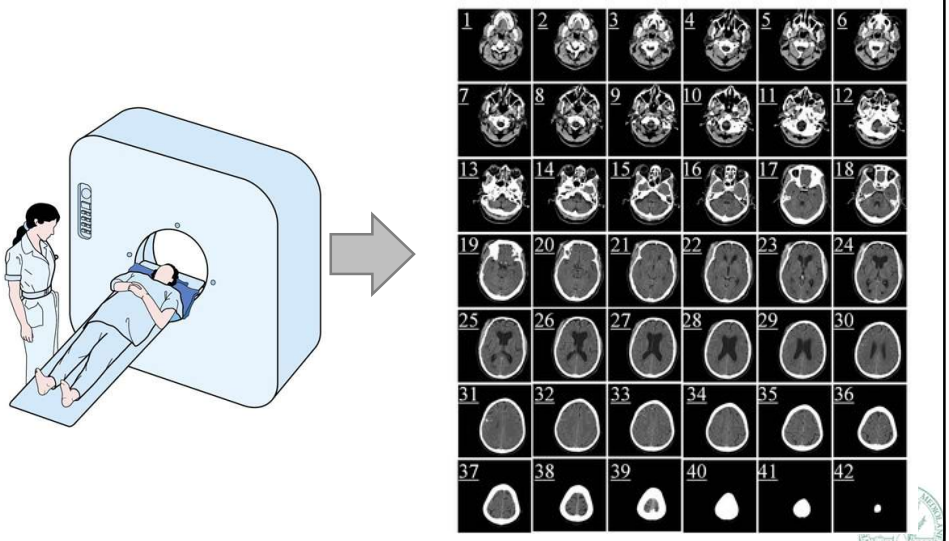



```
float volume[RES_X] [RES_Y] [RES_Z]
```

48

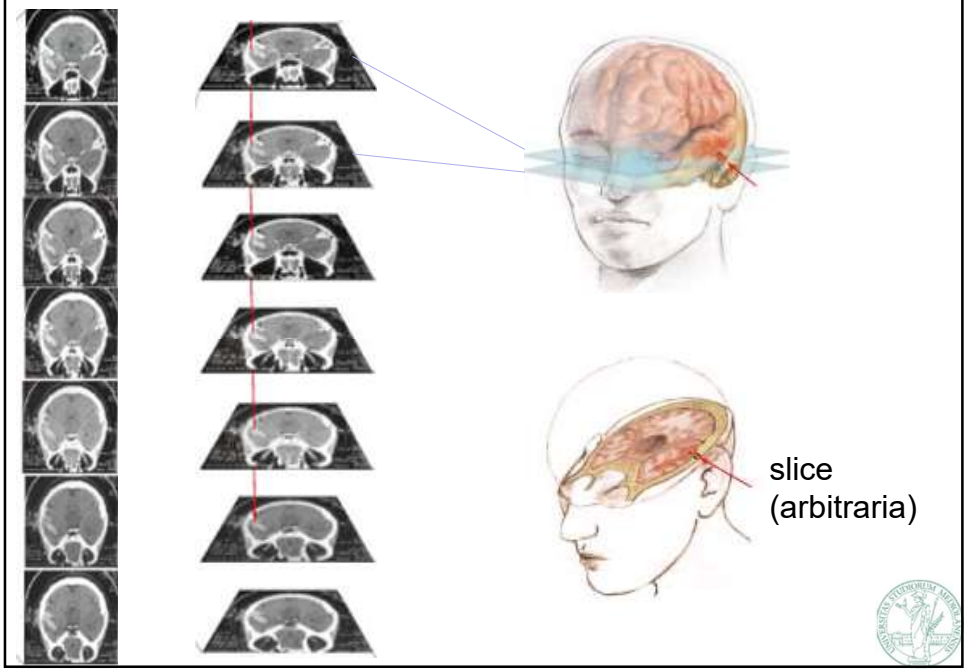
Se 1 voxel = 1 float (valori di densità)

✓Output naturale di **CT scans**



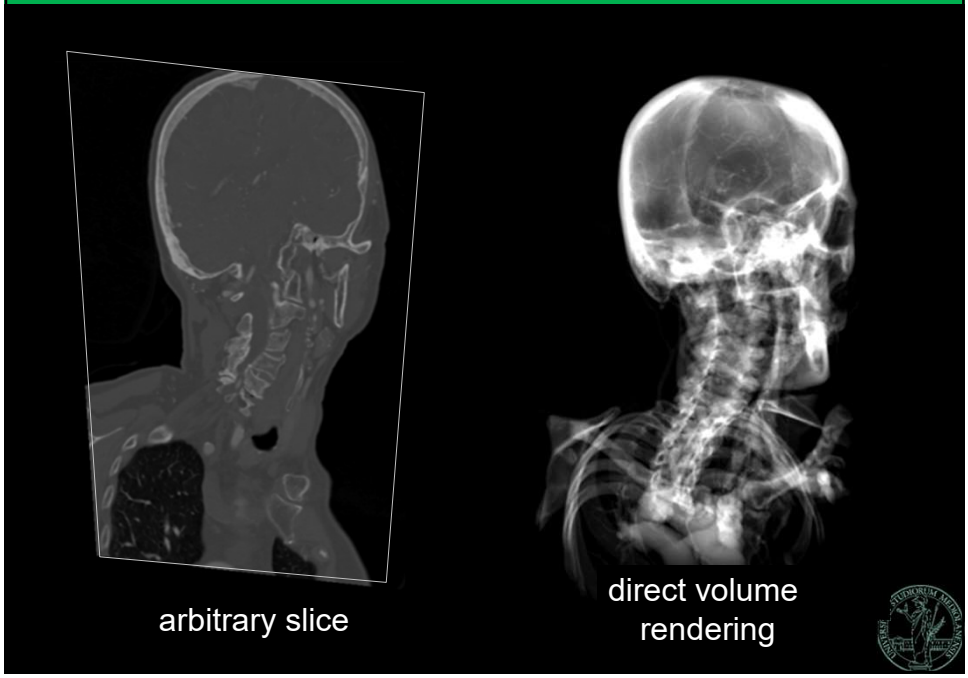
49

Se 1 voxel = 1 float (valori di densità)



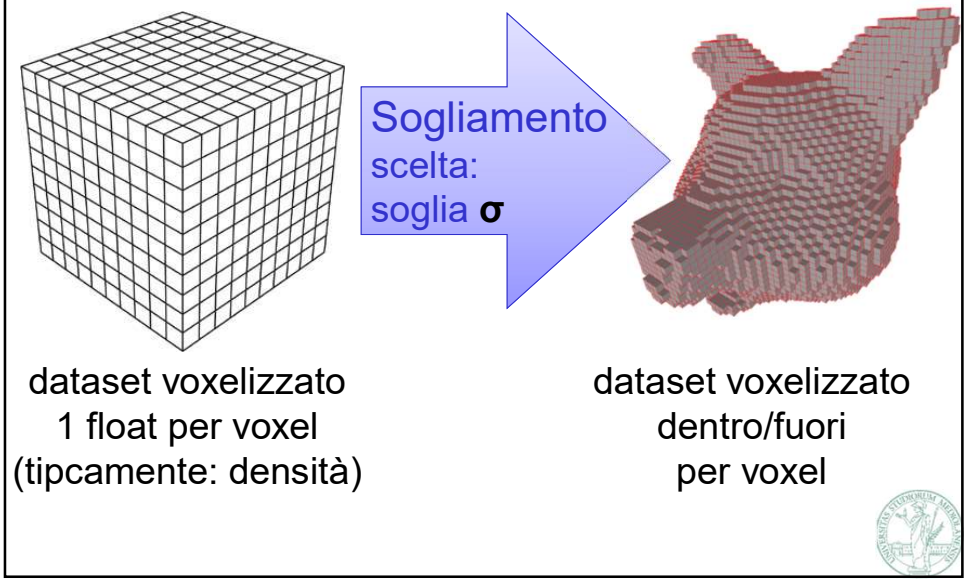
52

Voxelized volumes con voxel = 1 float fra 0 e 1



53

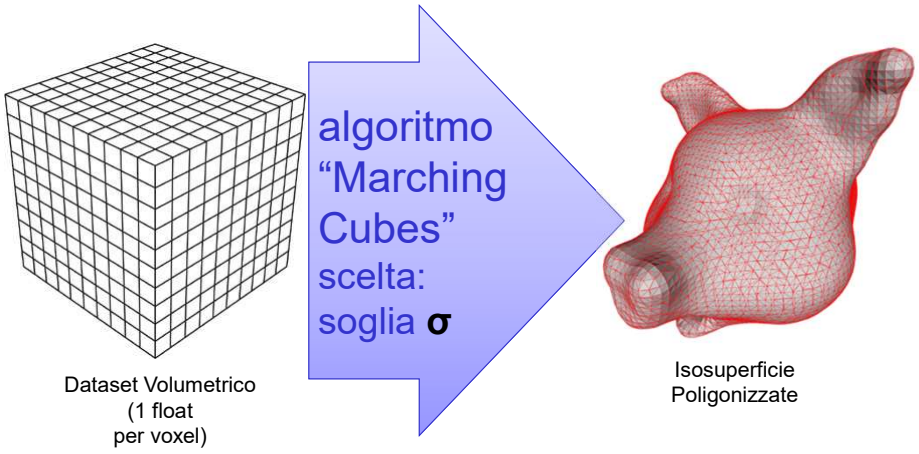
Soglia sulla densità



54

Poligonizzazione di un modello volumetrico

o segmentazione



55

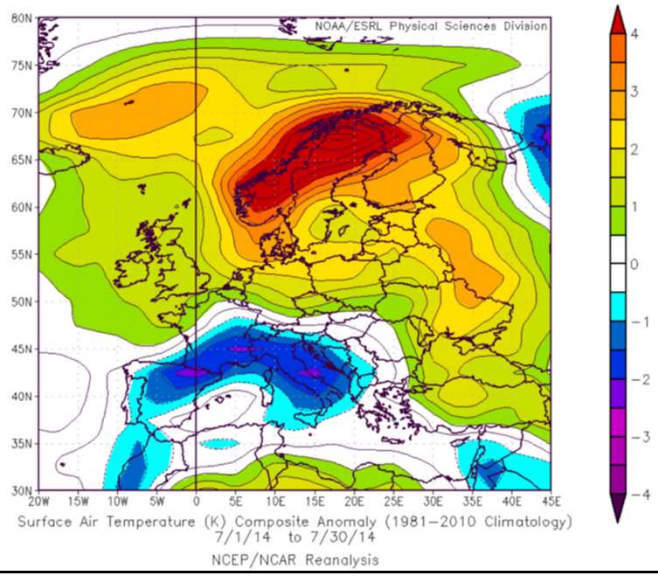
Isolinee e isosuperfici

- ✓ Su un piano (2D):
  - ⇒ ogni punto del piano ha un valore scalare (esempio: pressione, o altezza – height field)
  - ⇒ prendo tutta la regione 2D con valore > **σ**
  - ⇒ il bordo di questa regione è una linea … i cui punti hanno valore tutti **σ** e che racchiude tutti i valori di valore > **σ**
  - ⇒ è la « **isolinea di valore σ** » (linea di isovalori)
- ✓ Su un volume (3D):
  - ⇒ ogni punto dello spazio ha un valore scalare (esempio: densità, pressione, temperatura…)
  - ⇒ prendo la regione 3D con valore > **σ**
  - ⇒ il bordo di questa regione è una superficie… i cui punti hanno tutti valore è **σ** che racchiude tutti i valori di valore > **σ**
  - ⇒ è la « iso-superficie di valore **σ** »

56

Esempio di isolinea

✓ Isolinee di pressione



57

Poligonizzazione di un modello volumetrico

✓ Obiettivo:
- ⇒ dato un modello volumetrico voxel
  un voxel = un valore scalare (per es, densità, temperatura…)
- ⇒ produrre una tri-mesh (two-manifold, ben orientata, e chiusa) che racchiuda tutti i voxel di valore superiore ad una certo valore soglia σ
- ⇒ Si tratta cioè di produrre una **iso-superficie** di valore σ:
  la superficie di tutti i valori continui nel volume che valgono esattamente σ

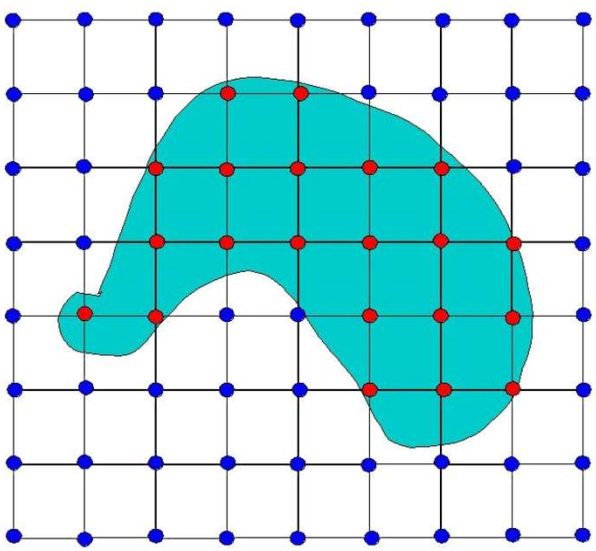
✓ Algoritmo: «**marching cubes**»

✓ Vediamo prima un analogo in 2D:
algoritmo «**marching squares**»
- ⇒ data un'immagine rasterizzta in cui
  un pixel = un valore scalare (per es, denistà, temperatura…)
- ⇒ produce una **iso-linea** linea chiusa di valore σ
  (appossimata da segmenti) che racchiude tutti i pixel di valore superiore a σ

58

Algoritmo marching squares (per 2D)

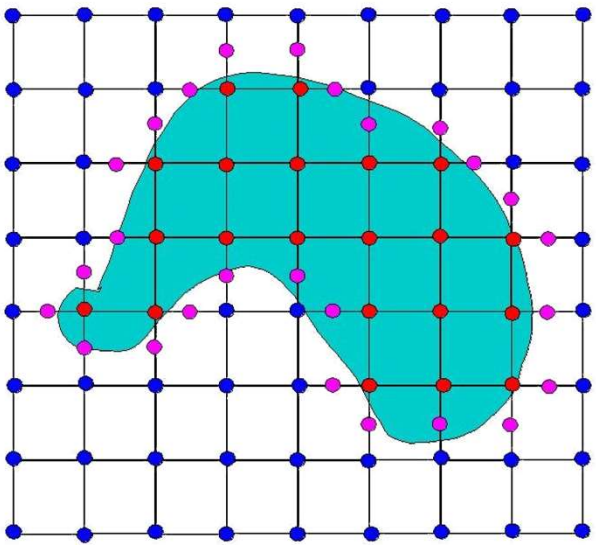
✓Sogliare voxels



60

Algoritmo marching squares (per 2D)

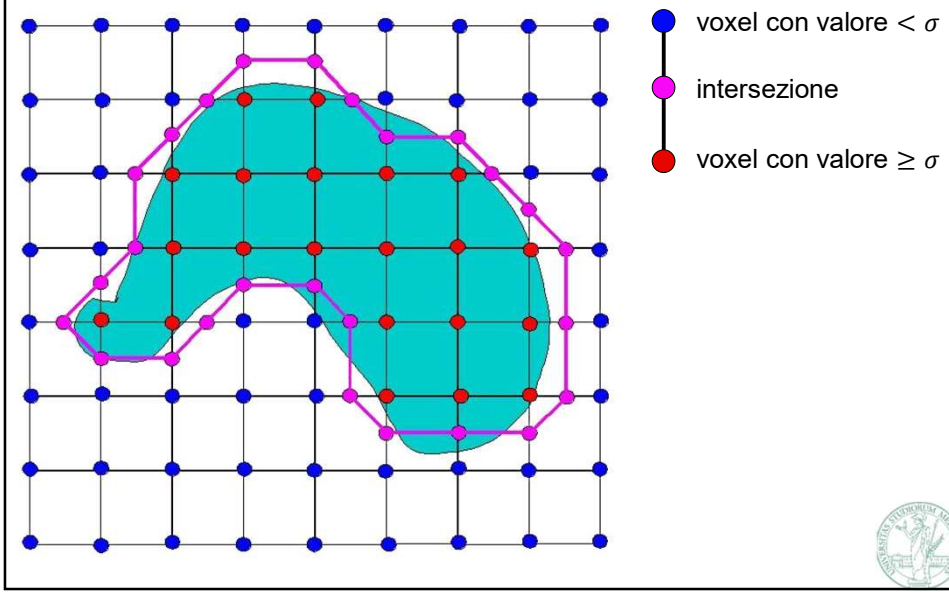
✓Trovare intersezioni



61

Algoritmo marching squares (per 2D)

✓Unire intersezionicreando segmenti

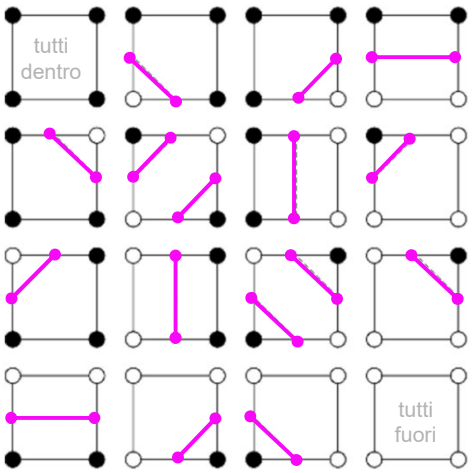


62

Algoritmo marching squares (per 2D)

Come trovare
i segmenti che connettono
le intersezioni?

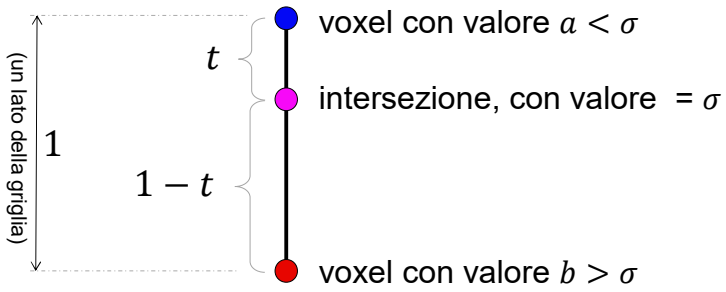
- ✓ Consideriamo un quadrato fra 4 voxel
- ✓ Ogni suo vertice è «dentro» $< \sigma$ o «fuori» $\geq \sigma$
- ✓ Per ogni combinazione, (e sono solo $2^4 = 16$) decido, una volta per tutte, i segmenti da costruire per unire le intersezioni
- ✓ ottengo questa tabella:



63

Algoritmo marching squares (per 2D)

✓Come trovare le intersezioni



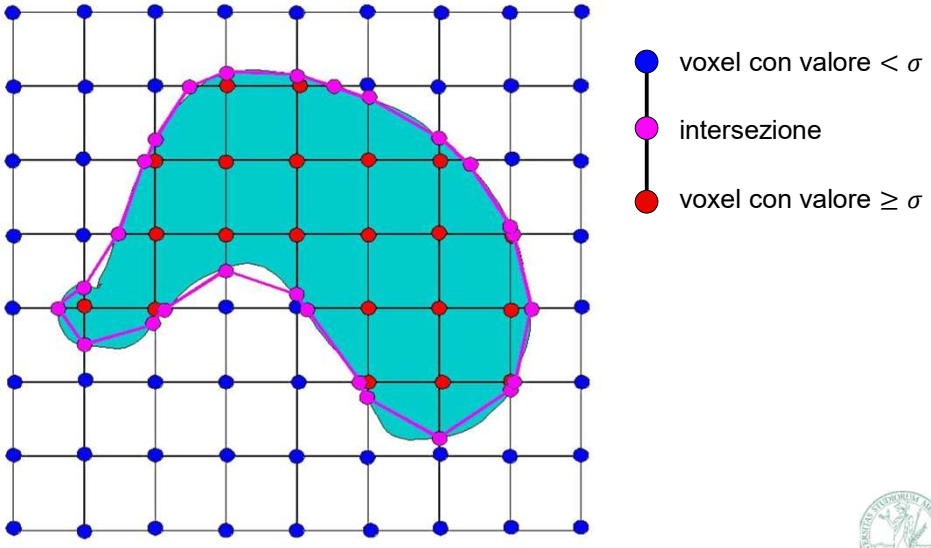
✓Ipotesi: segnale interpolato linearmente:

$$a(1-t)+b\,t = \sigma \quad \Leftrightarrow \quad t = \frac{\sigma - a}{b - a}$$

64

Algoritmo marching squares (per 2D)

✓Variante: trovare intersezioni e unirle



65