Rendering volumetrico

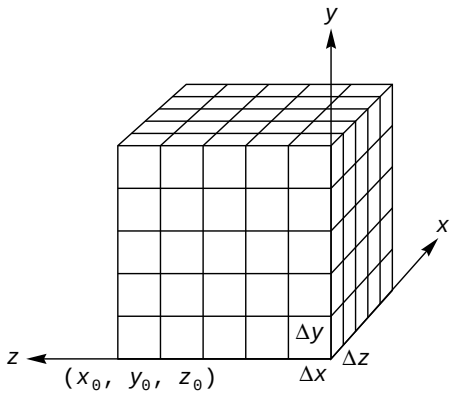
Dove si introducono tecniche per ottenere proiezioni 2D di insiemi campionati di dati 3D

- **Introduzione**
- **Rendering volumetrico diretto**
- **Estrazione di isosuperfici**

Introduzione

Dati volumetrici

- Il soggetto della visualizzazione è un campo scalare tridimensionale (una funzione scalare definita sullo spazio 3D), campionato in un insieme finito e discreto di punti.
- Se il reticolo di campionamento è equispaziato possiamo associare ad ogni punto del reticolo un elemento di volume (sempre uguale), che prende il nome di **voxel**.



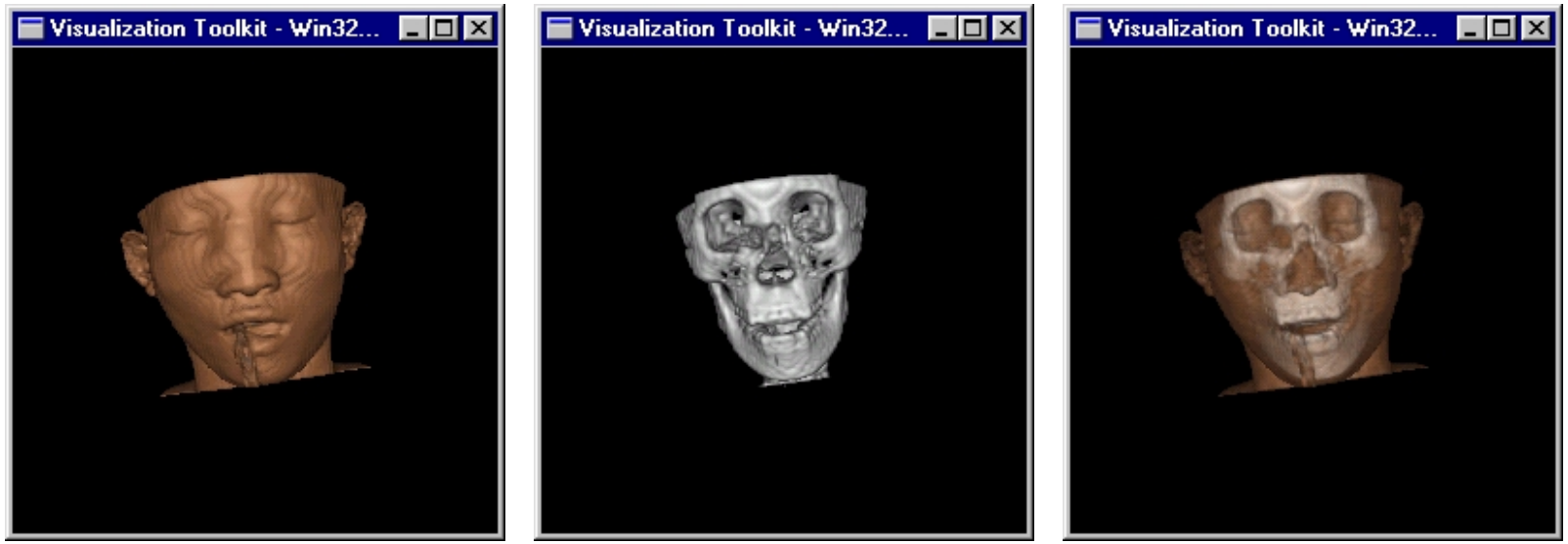
© E. Angel

- Vi sono vari esempi di campi scalari rappresentati con una struttura a voxel:
  - Tomografie assiali, che permettono di rappresentare un corpo umano (o parte di esso) con valori scalari (in genere legati all'assorbimento della radiazione da parte del tessuto) distribuiti in modo regolare su una griglia volumetrica
  - Misure di temperatura all'interno di un corpo o di un fluido.
  - ...

- Studieremo sono due tecniche base per la visualizzazione di dati volumetrici:
  - **Rendering volumetrico diretto**: si vuole visualizzare una proiezione dell'intero volume inteso come un unico blocco di dati: in principo tutti i voxel contribuscono alla visualizzazione.
  - **Isosuperfici**: fissato un valore del campo scalare (isovalore), si estrae la isosuperficie corrispondente in forma di maglia poligonale e si procede al rendering di quest'ultima.
- Ricordiamo che si possono avere anche rappresentazioni volumetriche di oggetti opachi, in cui i voxel assumono valori binari (pieno, vuoto). Entrambe le tecniche sopra menzionate si applicano in questo caso (con lievi semplificazioni per quanto riguarda la prima tecnica).
- Un'altra tecnica (**slicing**) consiste nel tagliare il volume con un piano definito (interattivamente) dall'utente e fare il rendering dei valori del campo scalare nel piano (per interpolazione).

Rendering Volumetrico diretto

- Lo scopo del rendering volumetrico è quello di mostrare in una immagine le caratteristiche di una regione solida (compreso l'interno).
- Tutti i voxel contribuiscono alla immagine finale.
- Per prima cosa si deve assegnare a ciascun voxel un valore (colore) ed una opacità.
- Questo si ottiene defininendo una **funzione di trasferimento RBGA** che mappa il valore contenuto in ciascun voxel in un valore di RGBA.
- Si ha così un parallelepipedo di materiale colorato e (semi)trasparente che può essere reso graficamente (*rendered*) impiegando tecniche quali:
  - **Ray Casting** (image-order)
  - **Voxel Splatting** (object-order)
  - **Shear Warp** (object-order)
- Si usa di solito la proiezione ortografica perchè più semplice, ma rende meno la profondità.



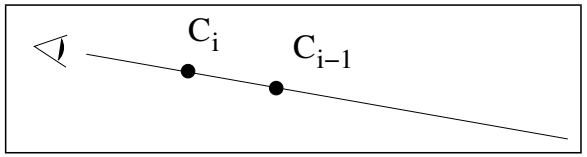
Immagini ottenute con diversi assegnamenti di opacità.

Ray casting

- Un fascio di raggi (di solito paralleli), uno per pixel, attraversano il volume.
- Ad ogni raggio viene assegnato un valore di colore, calcolato in base al colore ed alla opacità dei voxel che incontra.
- Si procede front-to-back, e ci si ferma sul primo voxel completamente opaco che si incontra, oppure all'uscita dal volume.
- Dato il voxel (i)-esimo che il raggio attraversa, sia C il suo colore ed O la sua opacità; allora il colore accumulato sul raggio dopo la vista del voxel (i)-esimo sarà dato da

$$C_i = OC + (1 - O)C_{i-1}$$

dove C_{i-1} è il valore calcolato per il voxel successivo (siccome procediamo front-to-back, è quello che sta dietro).

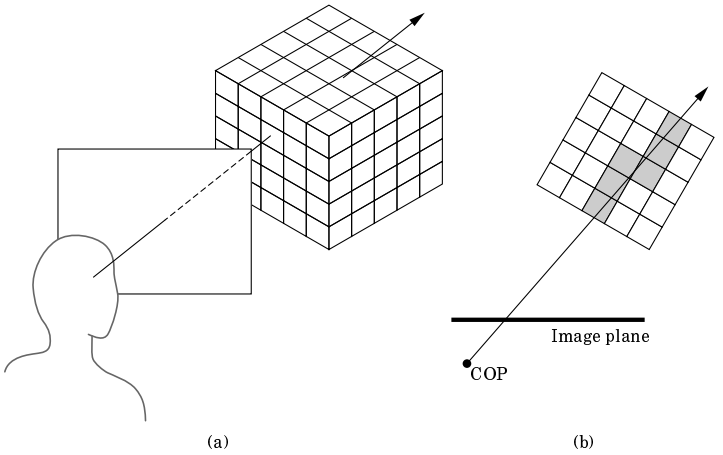


- La formula è ricorsiva: per valutare C_i serve C_{i-1} che verrà calcolato al passo successivo. Se la si "srotola" si vede che bisogna ricordarsi anche l'opacità dei pixel processati, oltre che il colore.

- Si può anche procedere back-to-front, come nell'algoritmo originale di Levoy (1987). In tal caso si applica la stessa formula di composizione ma in modo iterativo (invece che ricorsivo).
- Si parte dal fondo e quindi C_{i-1} viene calcolato prima di C_i.
- È come un algoritmo del pittore, e non serve ricordare la trasparenza.
- Però i voxel vengono processati sempre tutti: anche se c'è un voxel completamente opaco vicino all'osservatore (sul front) si processano (inutilmente) tutti i voxel che gli stanno dietro (tipico del pittore).

- Nel ray casting bisogna trovare i voxels attraverso cui il raggio passa.

- Il problema della intersezione raggio-oggetto è di solito complicato, ma nel caso di rappresentazione a voxels diventa banale: basta tracciare il raggio nello spazio voxelizzato con le stessa tecnica usata per disegnare una linea nell'immagine pixelizzata (v. rasterizzazione di segmenti).



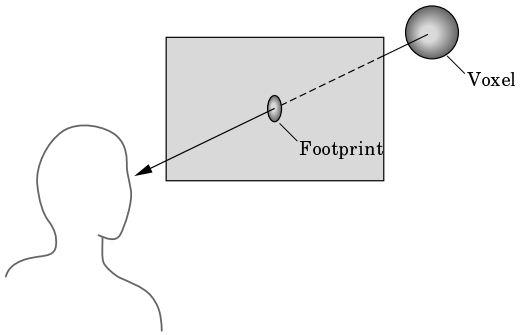
©E. Angel

- Non tutti i voxel però contribuiscono nella stesso modo: infatti ciascuno deve contribuire in ragione della lunghezza del percorso compiuto dal raggio al suo interno.

- Si prendono punti equispaziati sul raggio, e per ciascun punto si calcola il contributo con una media pesata dei voxel vicini (estensione 3D dell'interpolazione bilineare).

Voxel splatting

- Un'altro modo di effettuare il rendering, che prende il nome di **splatting**, consiste invece nel visitare i voxel e proiettarli sul piano immagine.
- Procede in *object-order*, mentre ray casting è *image-order*.
- A ciascun voxel viene assegnata una semplice forma che viene proiettata.
- L'impronta del voxel sull'immagine è una "macchia" che prende il nome di **splat** o **footprint**.
- Tipicamente si usa uno splat gaussiano di ampiezza opportuna (si dovrebbe usare la proiezione di un sinc tridimensionale, per il teorema del campionamento).



©E. Angel

- Si procede back-to-front, per piani a distanza decrescente dall'osservatore applicando la stessa tecnica di composizione del ray casting.
- Il vantaggio è che i dati vengono processati a fette (non serve mantenerli tutti in memoria contemporaneaente).

Shear Warp

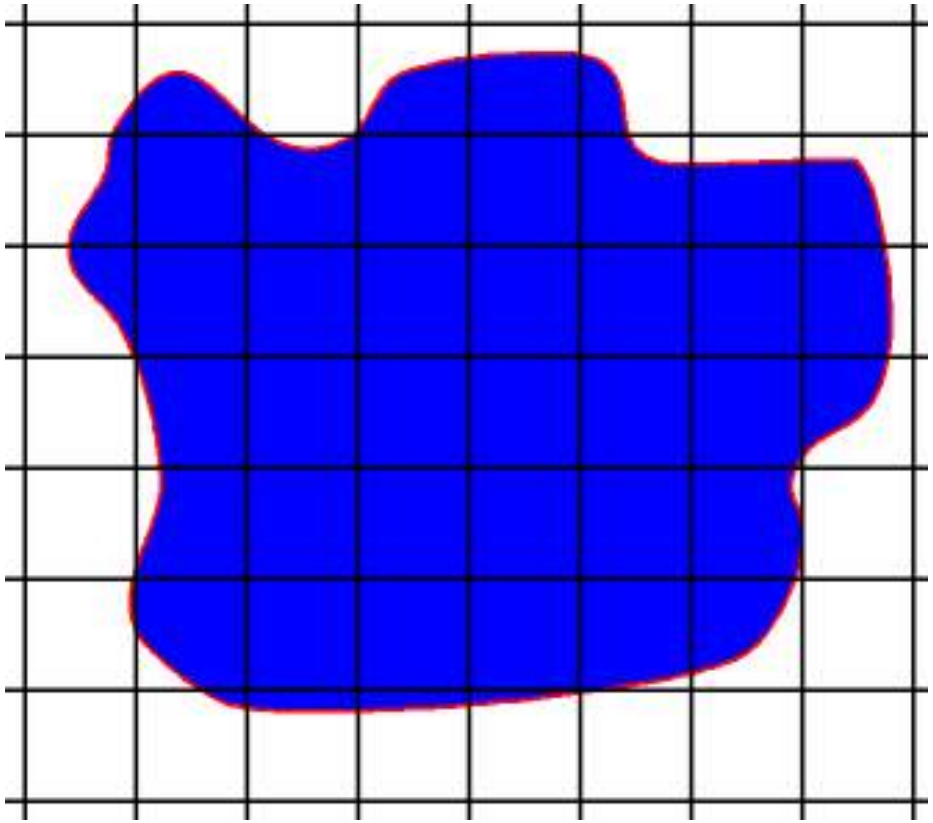
- È una tecnica a metà tra ray-casting e volume rendering. Come il voxel splatting procede in object order, ma come il ray-casting consente la *early ray termination* ed evita le complicazioni della determinazione del footprint.
- Il volume viene prima trasformato (**shear**) in modo che una faccia sia parallela al piano immagine (questo comporta ricampionamento e interpolazione)
- Si procede al rendering del volume (facile) procedendo front-to-back (come nel ray casting, ma senza dover tirare i raggi).
- L'immagine intermedia che ne risulta viene trasformata (**warp**) opportunamante per ottenere l'immagine finale corretta.

Estrazione di isosuperfici

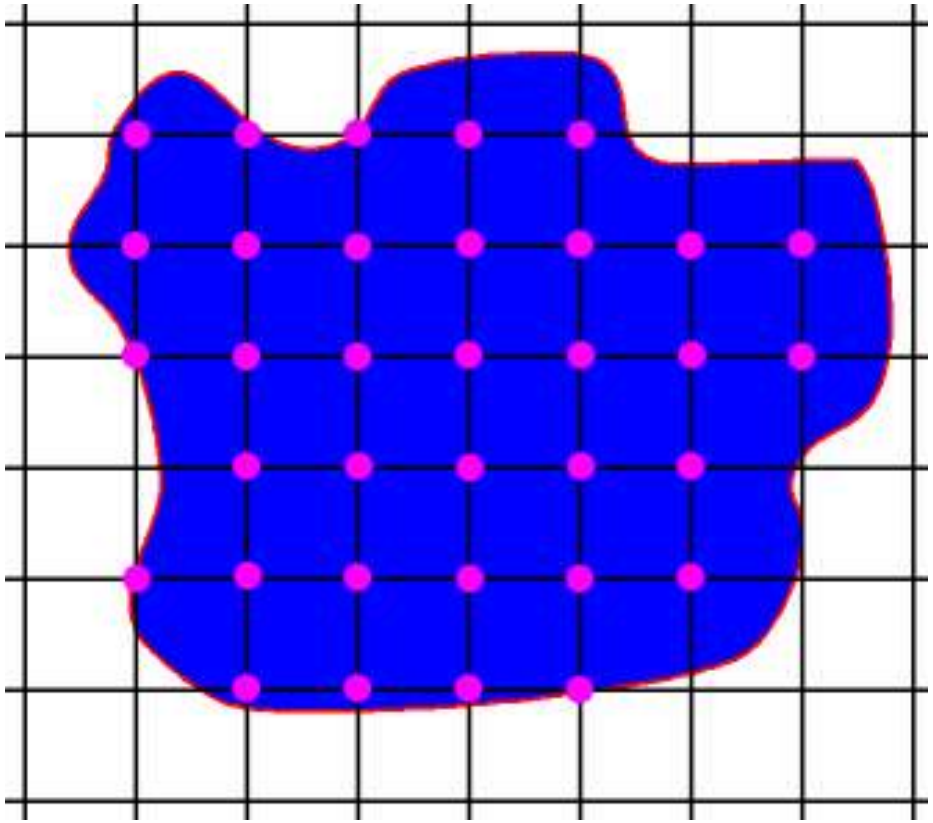
- Del tutto in generale, per rappresentare in uno spazio $n-1$ dimensionale dei dati scalari n dimensionali si possono usare delle **isosuperfici**
- Ovvero, si scelgono determinati valori nel range di variazione del campo scalare da rappresentare (isovalori) e si disegnano le superfici del campo scalare che posseggono tali valori (isosuperfici)
- Per esempio, in 2 dimensioni, se $f(x, y)$ è il campo scalare (un valore per ogni punto del piano), allora l'isocurva corrispondente al valore z è ottenuta dall'equazione $f(x, y) = z$
- Normalmente il campo scalare è rappresentato, come detto prima, da una matrice regolare di valori (voxel); in tal caso esiste una tecnica molto semplice ed efficiente per estrarre isosuperfici dai dati
- Vediamo prima il caso bidimensionale nel dettaglio, discuteremo dopo il caso volumetrico.

Caso 2D: marching squares

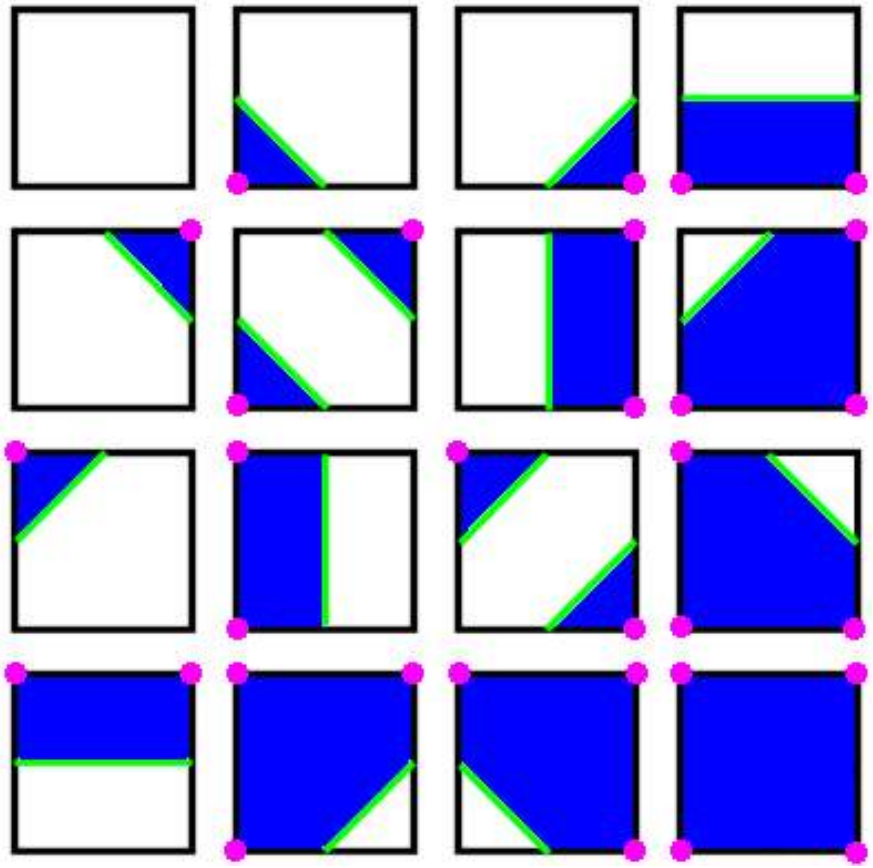
- Sia dato dunque un campo scalare bidimensioanle $f(x, y)$ campionato su un reticolo equispaziato (conosciamo i campioni, non f).
- Sia $f(x, y) = z$ la isocurva che si vuole approssimare.
- La isocurva divide il piano in due regioni, quella per cui $f(x, y) \geq z$ e quella per cui $f(x, y) < z$



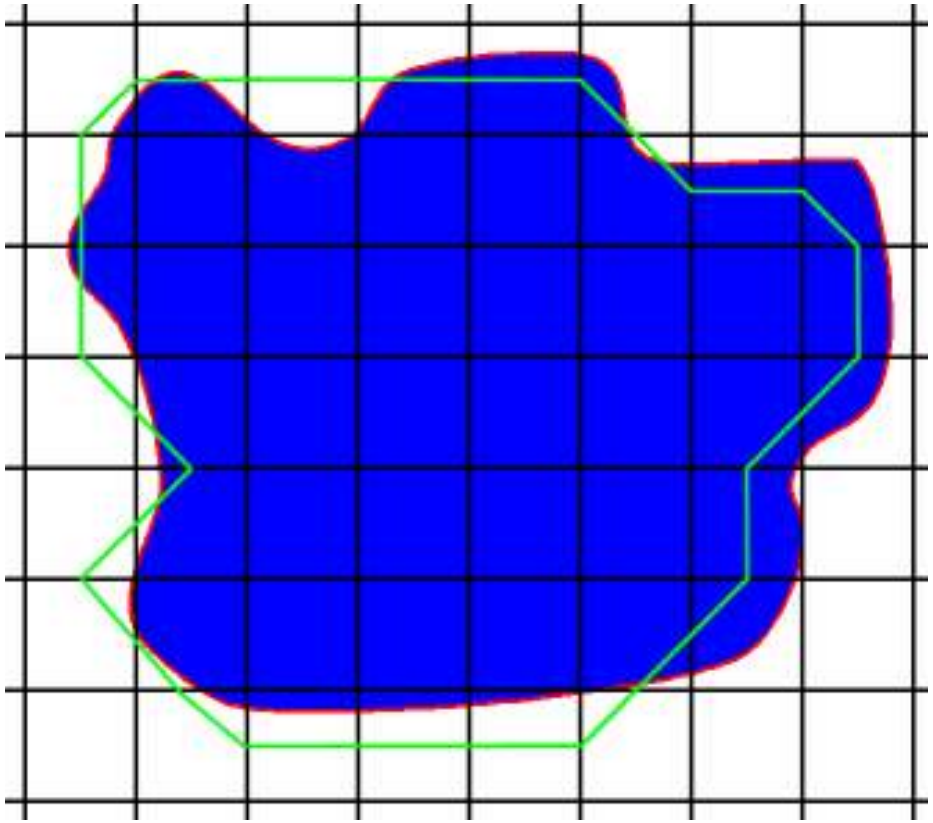
- Si marcano tutti i punti del reticolo che appartengono alla regione $f(x, y) > z$.
- La cosa equivale semplicemente a guardare il valore del campo in ciascun voxel e marcarlo o meno a seconda che tale valore sia maggiore o minore di z



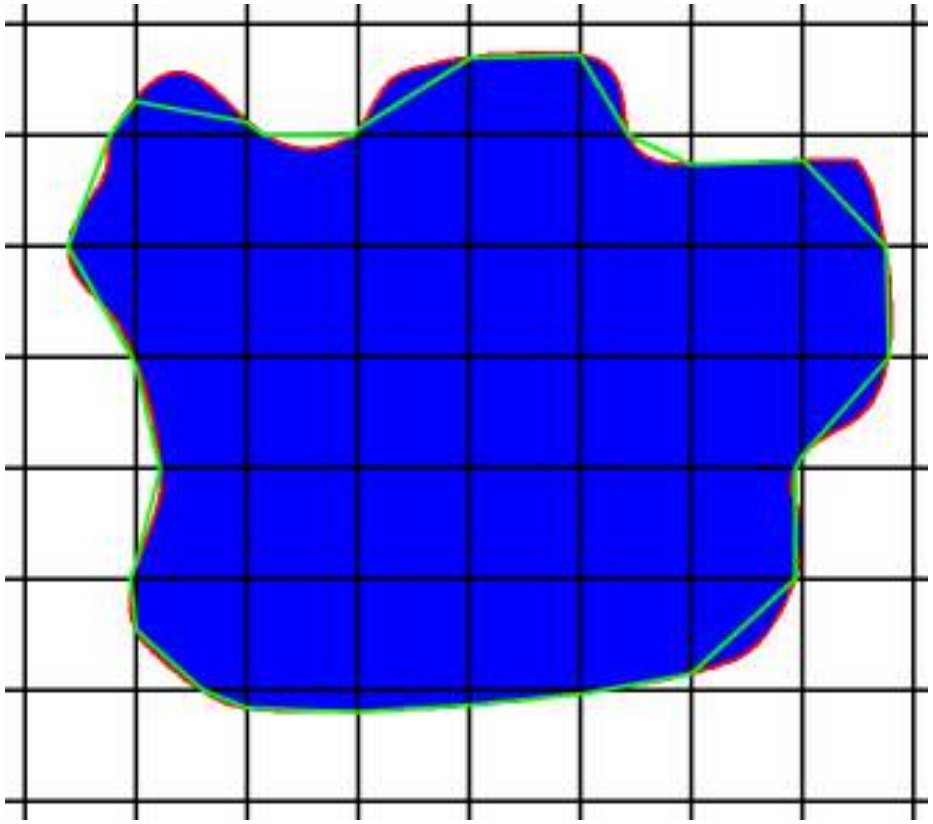
- Assumendo che la funzione $f(x, y)$ soggiacente sia ragionevolmente regolare, essa assume una volta il valore z su ogni lato che collega un vertice marcato con uno non marcato.
- Dato un quadrato qualsiasi del reticolo, è possibile quindi determinare la forma che deve assumere l'isocurva relativamente a quel singolo quadrato
- Si vede che in tal modo i quadrati sono elaborabili separatamente (da qui il nome dell'algoritmo)



- In prima approssimazione, posso decidere di collocare il punto in cui $f(x, y) = z$ esattamente a metà del lato.
- Il contorno che si trova in tal modo in realtà può non essere molto buono.



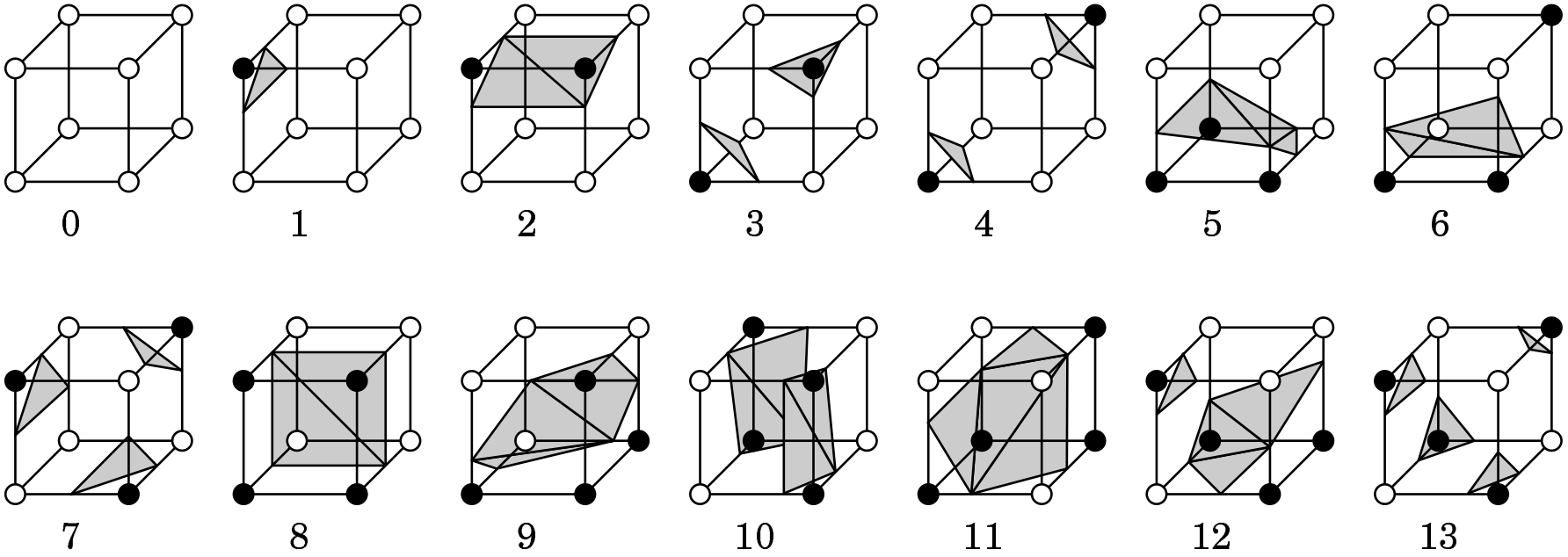
- È possibile migliorarlo decidendo il punto di intersezione dell'isocurva con il quadrato usando una interpolazione lineare tra i valori dei vertici del lato
- Il risultato è decisamente migliore.



- Riassumendo, dato un campo scalare bidimensionale discreto, si determinano i voxel che hanno valore del campo associato maggiore di z
- Nel reticolo formato con i voxel nei vertici (il duale), si esamina ciascun quadrato e si cerca il corrispondente nella tabella vista sopra (si può usare in modo efficiente una look-up table)
- In base a tale configurazione si disegna un singolo segmento (oppure nulla)
- Si "marcia" con l'algoritmo su tutti i quadrati
- Chiaramente si possono costruire più isocurve e disegnarle contemporaneamente per dare un'idea della struttura del campo scalare
- Da notare che per simmetria del problema, la tabella di look-up può essere ridotta da 16 elementi a soli 4

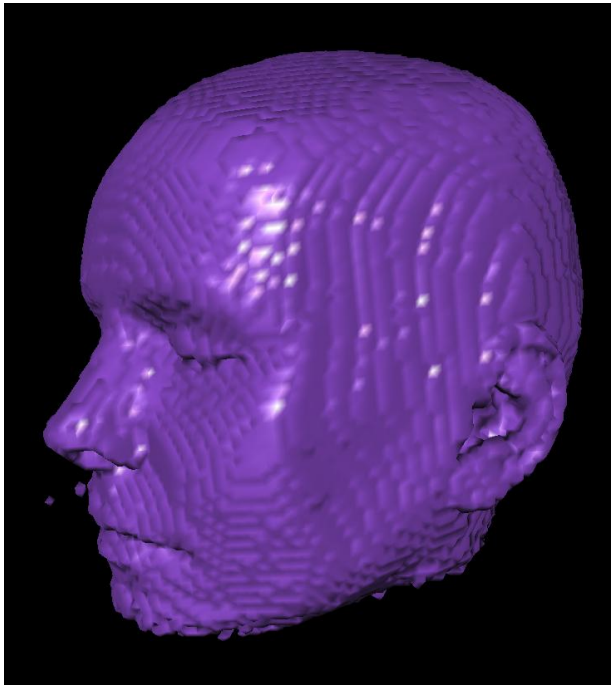
Caso 3D: marching cubes

- L'algoritmo di marching cube (Lorensen e Kline 1987) è identico, ma il risultato finale è una mesh triangolare che rappresenta la isosuperficie
- La tabella di look-up è più complessa: sono 256 casi, riconducibili a 14 per simmetria.

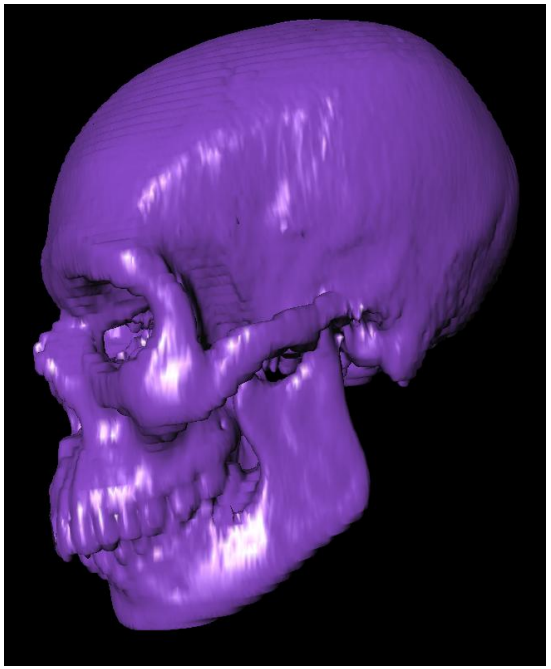


©E. Angel

Un esempio di estrazione di isosuperfici con marching cubes:



(4) isovalue=4



(5) isovalue=8

- In realtà l'algoritmo, sia in versione bidimensionale che in quella tridimensionale, presenta delle **ambiguità**, in generale risolvibili con una suddivisione della griglia
- Come al solito abbiamo appena toccato l'argomento che è, ovviamente, ben più vasto
- Notare che l'algoritmo marching cubes permette di trasformare una rappresentazione volumetrica (a voxels) di un oggetto solido in una rappresentazione a maglia poligonale della sua superficie.