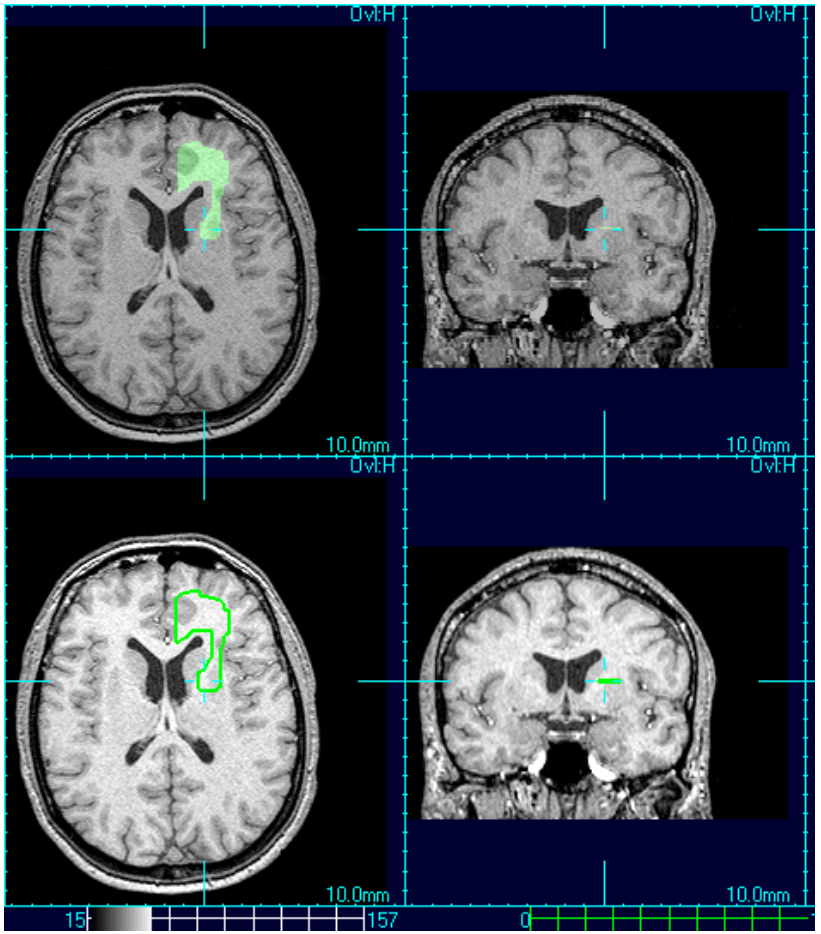
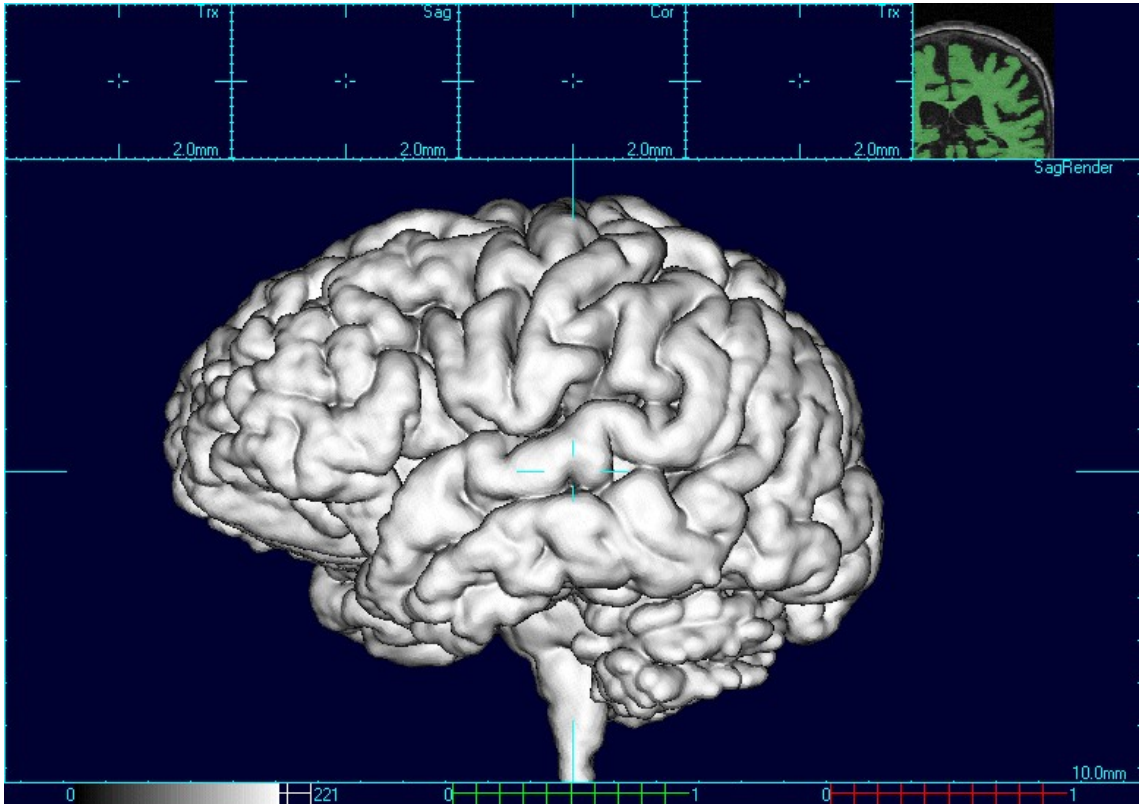
Elaborazione e Visualizzazione di Dati Volumetrici (3D)



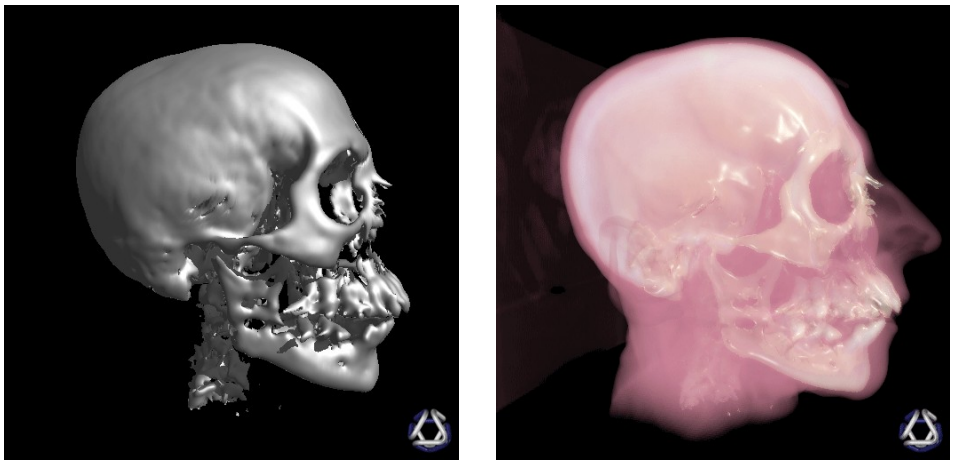
"www.colin-studholme.net"



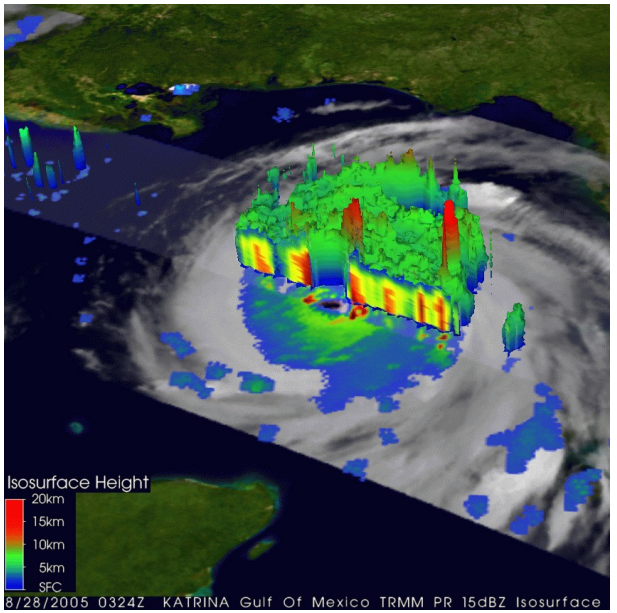
"www.colin-studholme.net"

Ambiti Applicativi

- Medical Imaging:
  - Estrazione isosuperfici da dati volumetrici;
  - Visualizzazione di volumi a livelli di trasparenza;
- Meteorological Imaging:
  - Estrazioni isosuperfici per la visualizzazione di un subset di dati



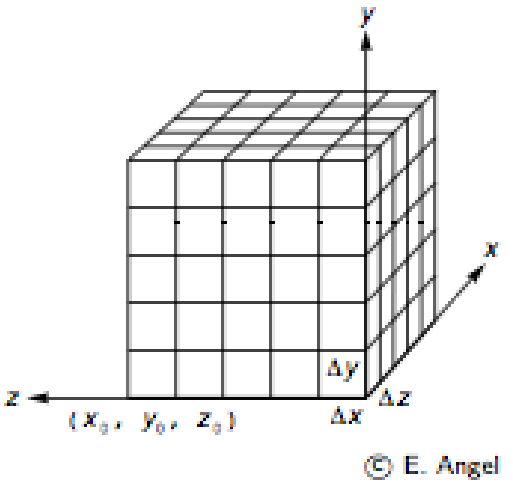
"www.vis.uni-stuttgart.de"



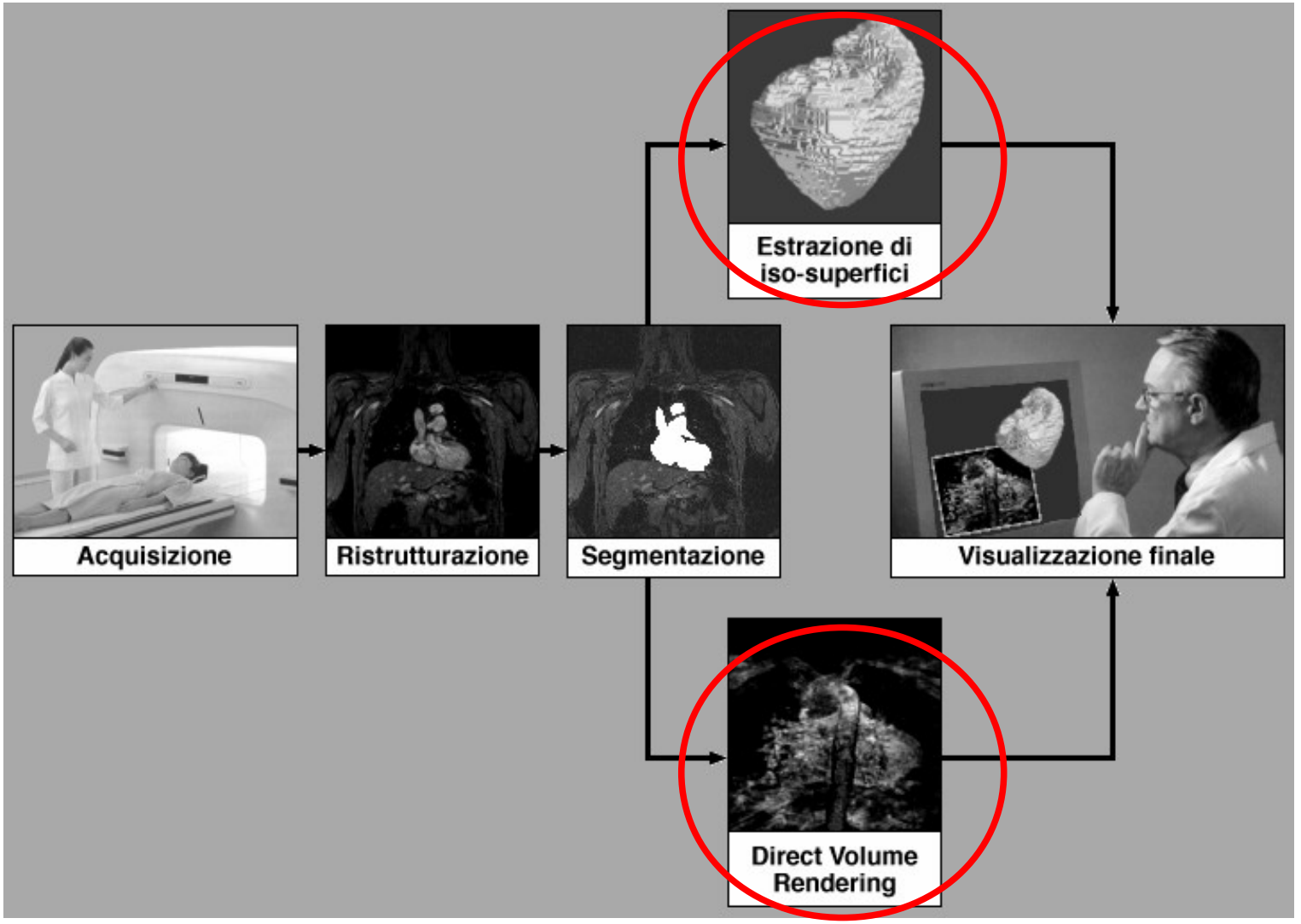
"trmm.gsfc.nasa.gov"

Dati Volumetrici

- I dati sono rappresentati da un campo scalare o vettoriale campionato in un insieme finito e discreto di punti;
- Ad ogni singolo punto dell' insieme possono essere associate più informazioni:
  - In ambito medico ad esempio il tipo di tessuto, la sua densità, valori di temperatura, …
  - In ambito geografico l' altitudine, la pressione, la densità di ossigeno, acqua, …
- Se il campionamento è avvenuto con un reticolo equispaziato possiamo associare ad ogni punto del campo un elemento di volume infinitesimo chiamato **voxel** (che ci tornerà molto utile in fase di elaborazione)



Pipeline

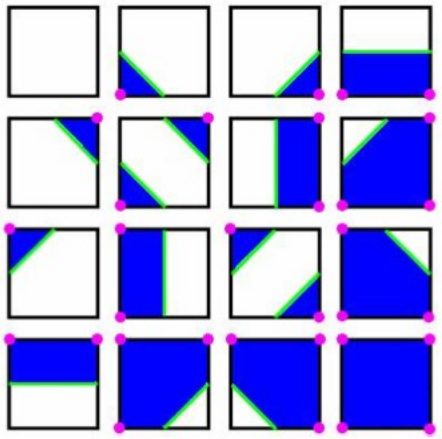


"www.crs4.it"

N.B.: sebbene lo schema si riferisca al campo medico la pipeline è assolutamente generale, eccezion fatta per la fase di segmentazione che non è sempre necessaria.

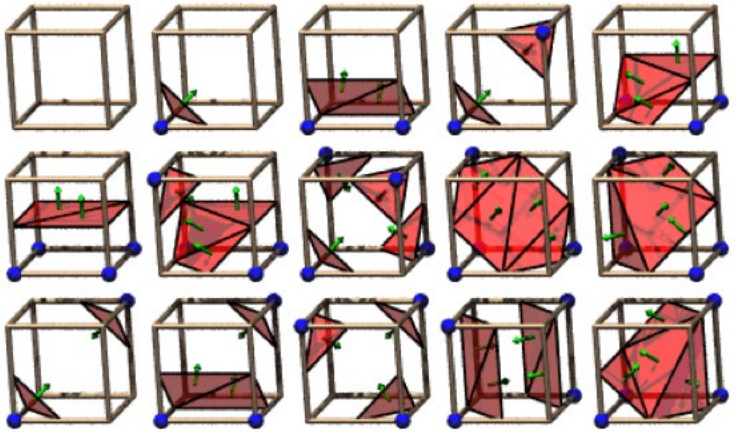
Estrazione di isosuperfici

- Caso particolare con set di dati a 3 dimensioni e subset da estrarre a 2 dimensioni;
- In realtà è un procedimento generico che riduce un insieme di dati n-dimensionali ad uno con (n-1) coordinate;
- Il principio è quello di fissare un valore costante per una delle dimensioni e ricavare i valori assunti dalle altre;
- Caso 2D --->
  ---> Marching Squares



"profs.sci.univr.it"

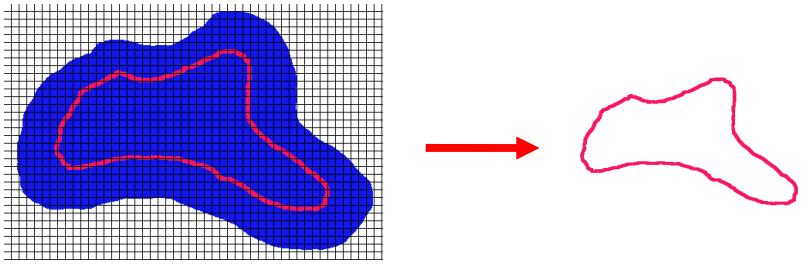
Caso 3D --->
---> Marching Cubes



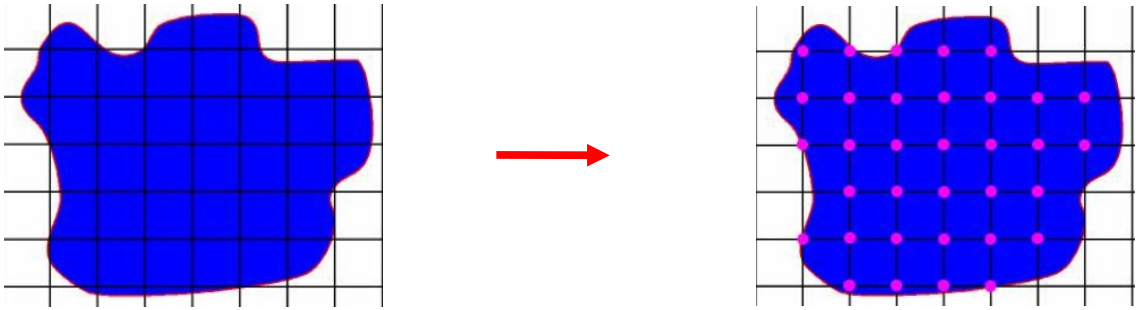
"www.exaflop.org"

Caso 2D: Marching Squares (1)

- Dato un campo scalare bidimensionale f(x,y) di cui si conoscono i campioni presi in un reticolo equispaziato;

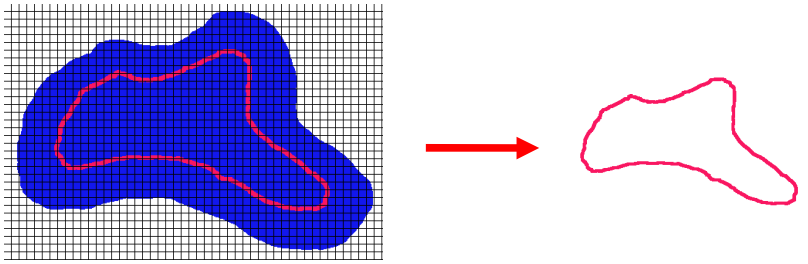


- Sia f(x,y)=C il valore associato alla isocurva che si vuole estrarre;
- Troviamo che tale valore divide il piano in due regioni: una per f(x,y)≥C ed una per f(x,y)<C;
- Marchiamo quindi tutti i punti del reticolo che soddisfano f(x,y)≥C, semplicemente confrontando il valore che assume ogni vertice di ogni quadratino (un voxel 2D in pratica):

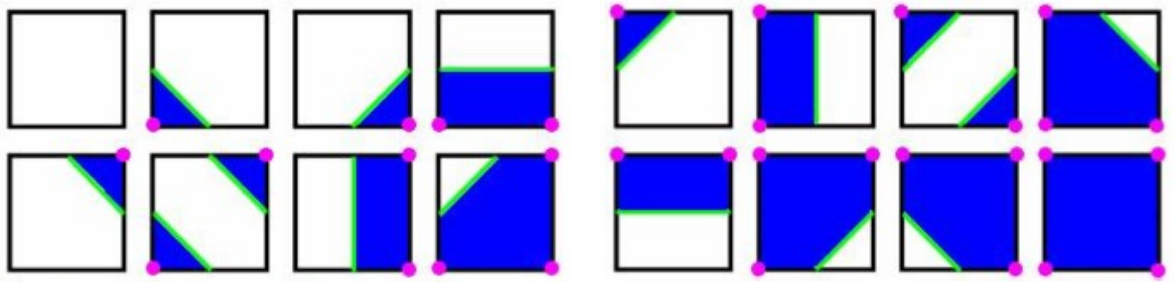


Caso 2D: Marching Squares (2)

- Assumendo ora che la funzione f(x,y) sia piuttosto regolare, possiamo affermare che essa assume il valore C esattamente una volta su ogni lato che collega un vertice marcato ad uno non marcato;



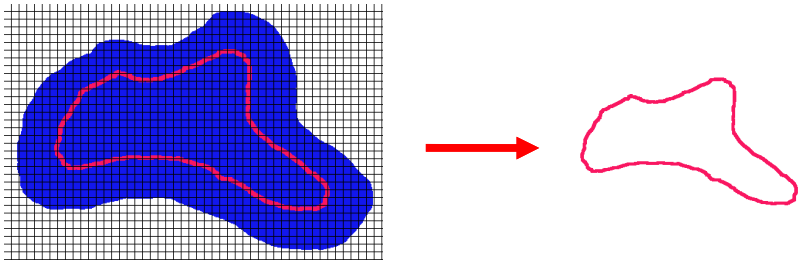
- Se così non fosse si potrebbe pensare di aumentare la risoluzione del campionamento;
- Grazie a questa considerazione possiamo trattare ogni voxel 2D separatamente, individuando 16 configurazioni possibili riguardanti l' approssimazione alla curva cercata:

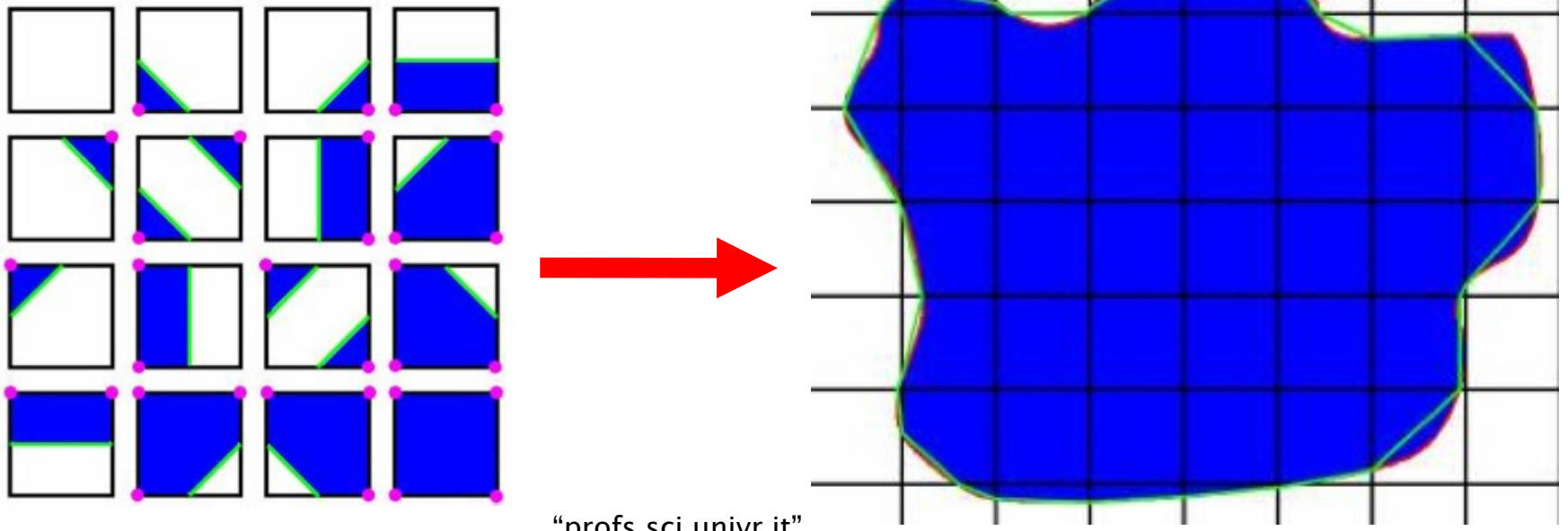


altre configurazioni possono essere ottenute per rotazione e/o per inversione

Caso 2D: Marching Squares (3)

- Prendendo come linea guida i 16 casi individuati, possiamo approssimare l' intersezione tra la isocurva ed ogni lato per interpolazione lineare tra i valori assunti dalla funzione f(x,y) nei vertici (di cui abbiamo i campioni) che delimitano tale lato;



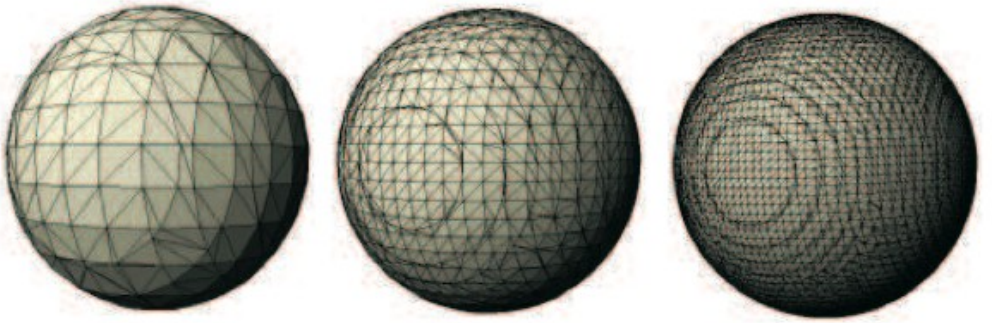


"profs.sci.univr.it"

- L' algoritmo ripete questo passo per ogni voxel 2D

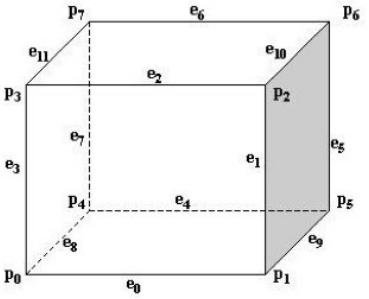
Caso 3D: Marching Cubes (1)

- Estensione dell' algoritmo Marching Squares;
- Il dataset in ingresso all' algoritmo è ora di tipo tri-dimensionale e l' output cercato è una mesh in 2D che approssimi l' isosuperficie definita dall' isovalore f(x,y,z)=C;

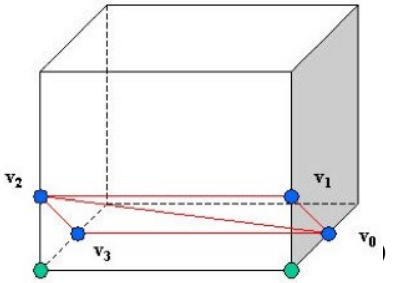


"www.sitobrusco.com"

- Ancora si assume che la funzione sia sufficientemente regolare, così che la superficie da estrarre intersechi una sola volta ogni cubetto infinitesimo (voxel 3D);
- Il primo passo consiste nell' individuare le cosiddette celle attive, ovvero le celle attraversate dall' isosuperficie:

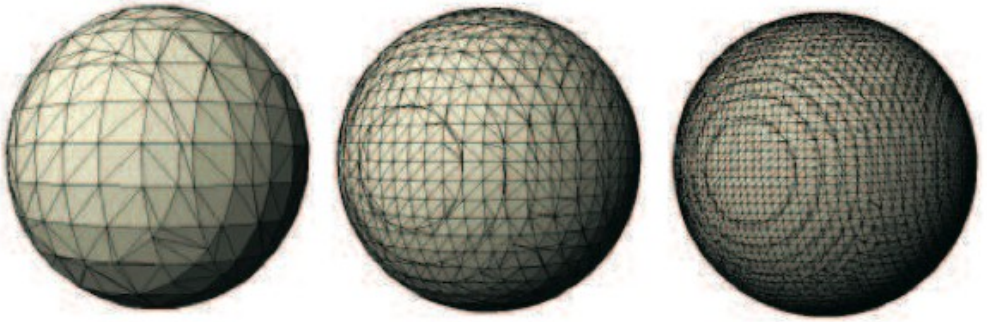


"Algorithms, Programming Methodologies and Tools for Grid Aware Visualization"



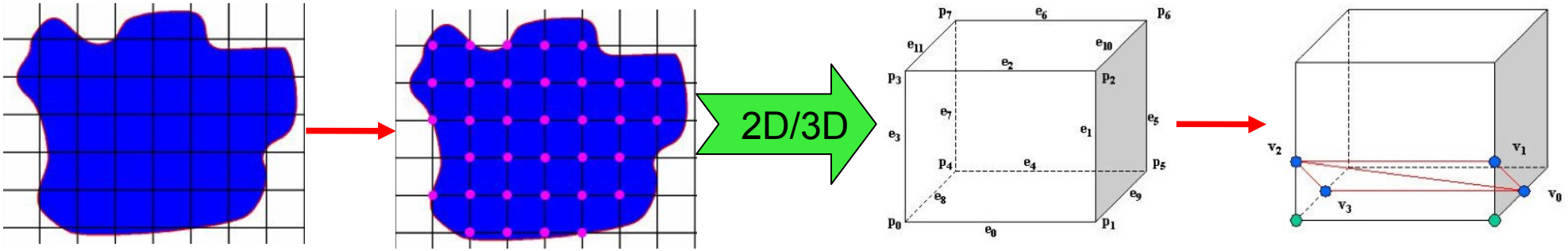
Caso 3D: Marching Cubes (2)

- Per ogni cella (voxel) confronto il valore assunto dai suoi vertici con l' isovalore;



"www.sitobrusco.com"

- Se il vertice è maggiore dell' isovalore lo contrassegno, altrimenti no (o viceversa);
- Le celle attive sono quelle che hanno un numero di vertici contrassegnati compreso tra 1 e 7 (estremi inclusi), ovvero le celle intersecate dalla superficie;
- Questo confronto è l' estensione di quello fatto nel caso 2D:

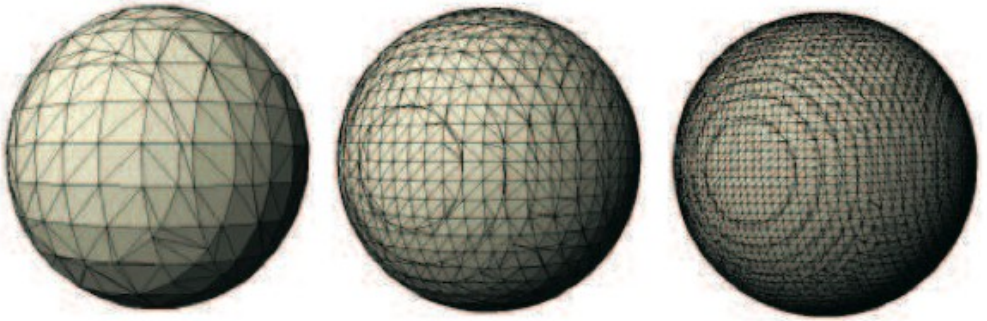


"profs.sci.univr.it"

"Algorithms, Programming Methodologies and Tools for Grid Aware Visualization"

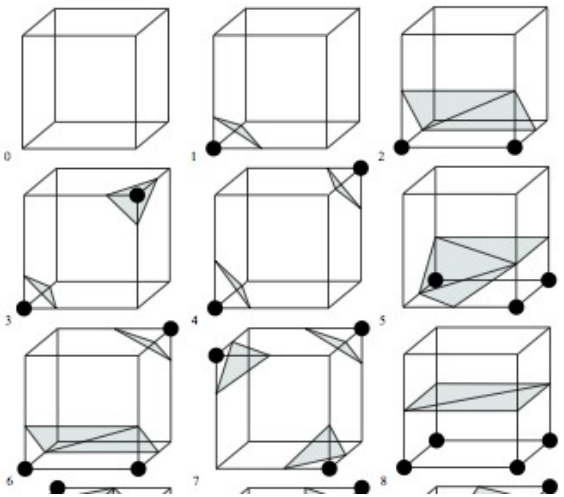
Caso 3D: Marching Cubes (3)

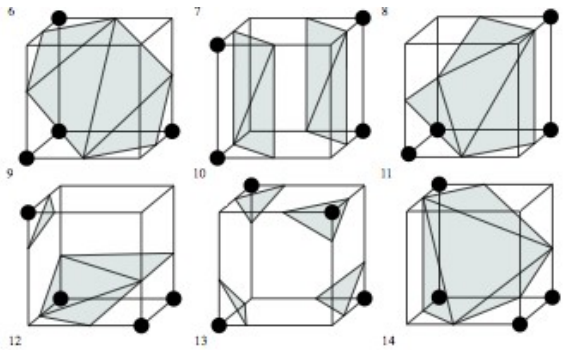
- Il secondo passo dell' algoritmo consiste nel determinare quale sia la mesh triangolare che meglio approssima l' intersezione dell' isosuperficie con la cella;



"www.sitobrusco.com"

- Come per il caso 2D possiamo considerare ogni voxel separatamente, e definire così una lookup table con i modi possibili in cui può avvenire l' intersezione:



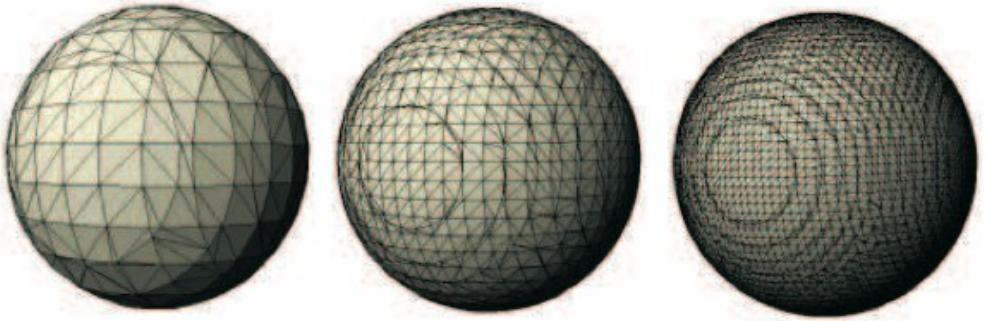


"www.sitobrusco.com"

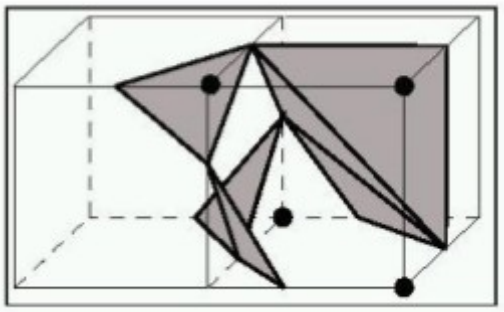
e come in 2D le altre configurazioni sono tutte ottenibili per rotazione e/o inversione

Caso 3D: Marching Cubes (4)

- Le 15 configurazioni minime trovate non sono però in questo caso sufficienti, poiché ci sono alcune ambiguità riguardo alle operazioni di inversione e/o rotazione:

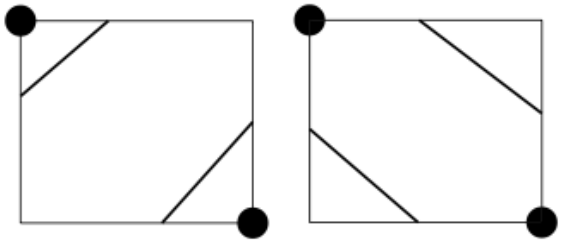


"www.sitobrusco.com"



"www.sitobrusco.com"

tale problema è dovuto alla situazione chiamata facce ambigue

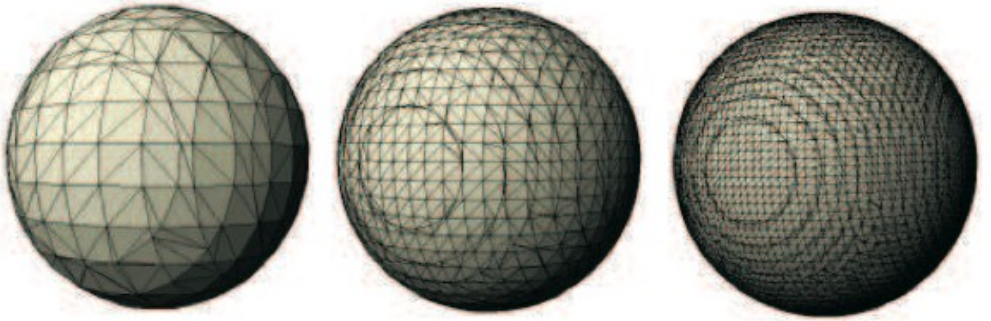


"www.sitobrusco.com"

- questa configurazione anomala si presenta quando ci sono 2 vertici marcati diagonalmente opposti e due vertici non marcati anch' essi opposti (in maniera simmetrica)

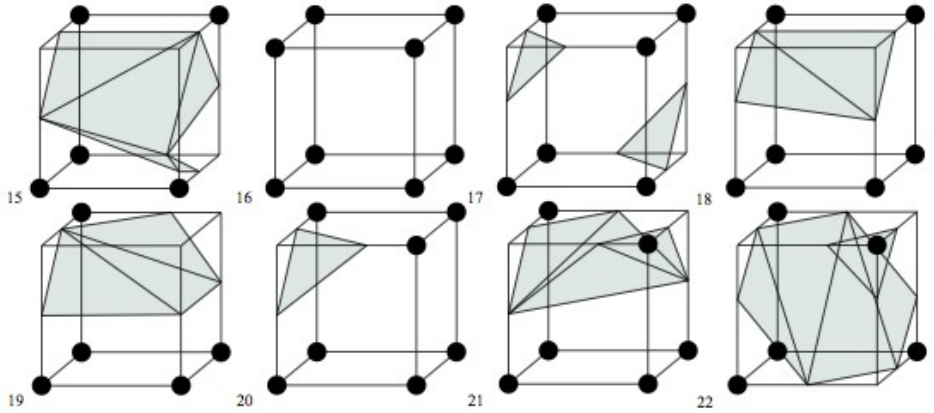
Caso 3D: Marching Cubes (5)

- L' ambiguità può essere risolta semplicemente aggiungendo altri 8 casi alla tabella di lookup, che rappresentano le configurazioni inverse e/o ruotate dei casi critici:



"www.sitobrusco.com"

"www.sitobrusco.com"

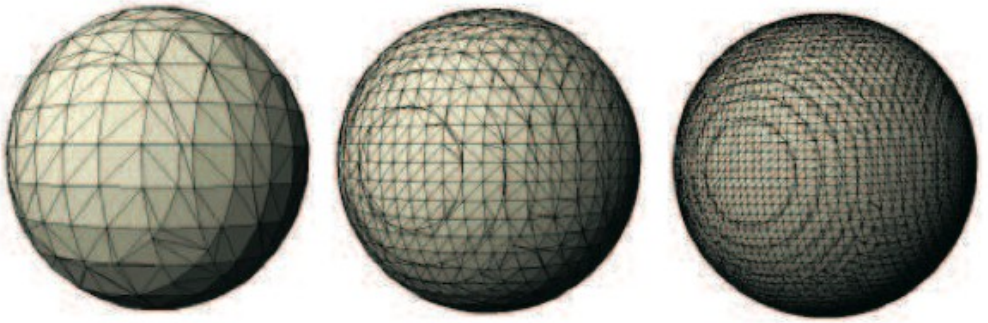


questi 8 casi cessano di essere considerati come ottenibili dai 15 originali, ma diventano casi del tutto nuovi;

- Così facendo, l' ambiguità è risolta poiché le configurazioni critiche non verranno più né ruotate né invertite

Caso 3D: Marching Cubes (6)

- L' operazione conclusiva dell' algoritmo è il calcolo delle intersezioni che i vertici dei triangoli originati dalla lookup table dovranno avere con i lati di ogni cella attiva;



"www.sitobrusco.com"

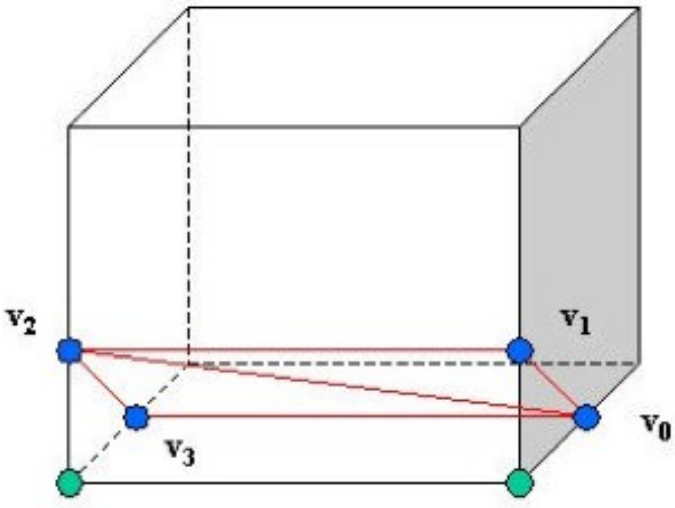
- Si usa a tal fine un' interpolazione lineare tra i vertici del voxel il cui lato in comune è intersecato dalla mesh approssimante (per trovare la posizione sul lato di uno dei 3 vertici del triangolo componente la mesh);

- Poiché si è ipotizzata la griglia di voxel regolare, i 2 vertici della cella differiscono per una sola coordinata, quindi il vertice del triangolo (intersezione) avrà come coordinate:

$$x = v1(x),$$

$$y = v1(y),$$

$$z = v1(z) + [C - v1(z)] / [v2(z) - v1(z)]$$



"www.sitobrusco.com"

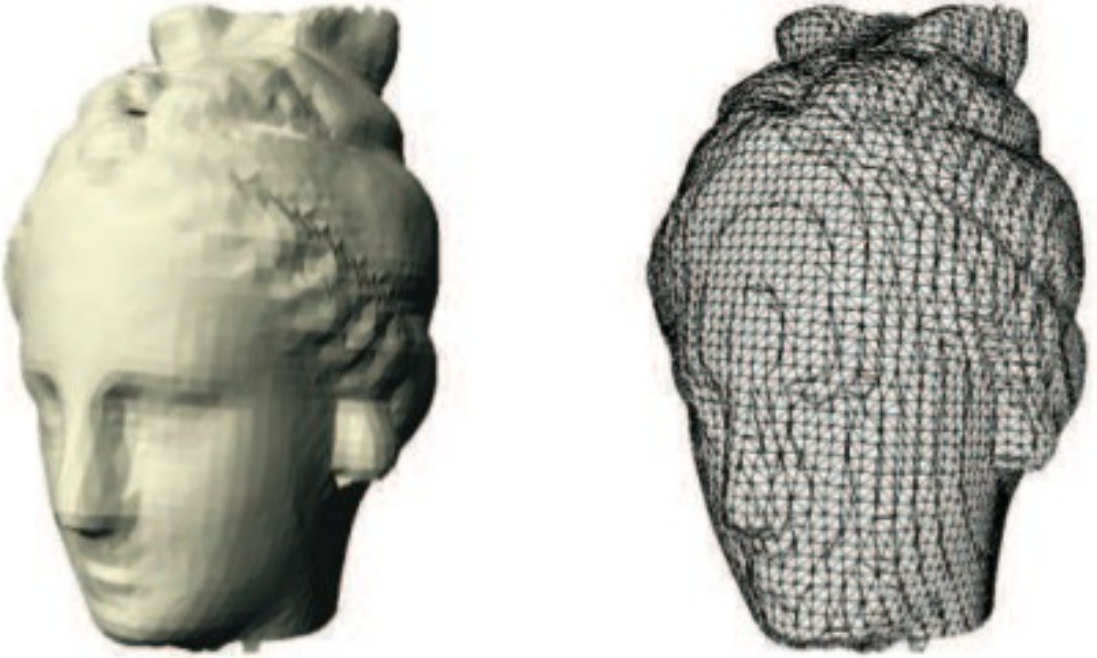


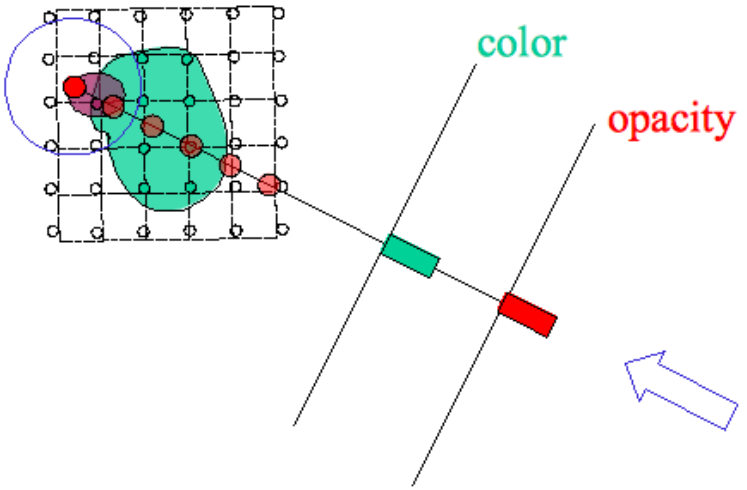
Immagine ottenuta con risoluzione 0,93 mm; 16.072 triangoli

Immagine ottenuta con risoluzione 0,45 mm; 72.415 triangoli

"www.sitobrusco.com"

Direct Volume Rendering (DVR)

- Visualizzazione diretta dei dati volumetrici (ancora campi scalari), ovvero senza geometria intermedia (cosa che invece avviene per le isosuperfici);
- Diverse tecniche per farlo:
  - Ray Casting;
  - Voxel Splatting;
  - Texture mapping;
  - …

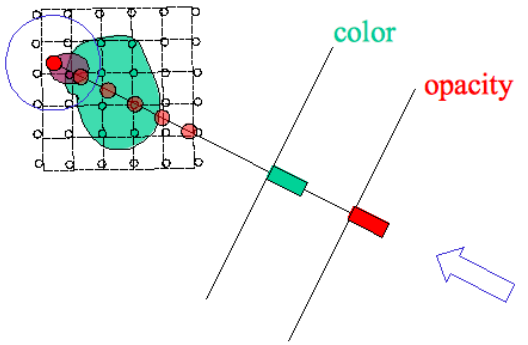


"www.cs.utk.edu"

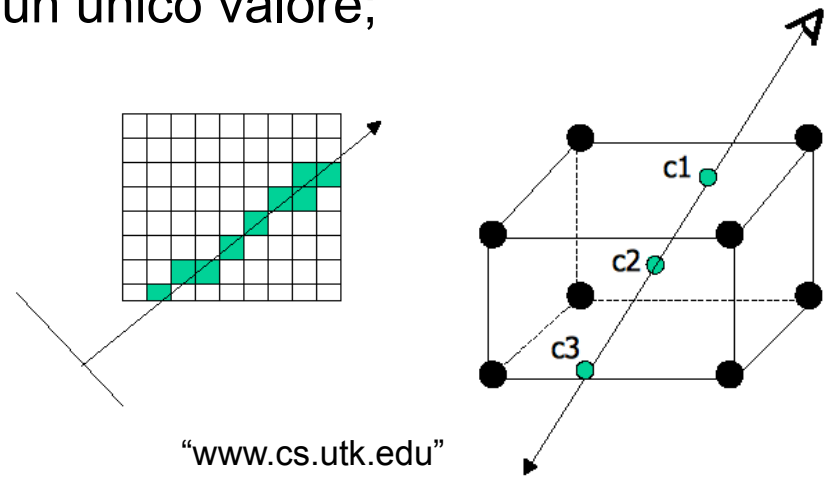
- Noi ci occuperemo del Ray Casting, che segue lo stesso principio del Ray Tracing: si proietta un raggio per pixel e si rende su schermo un' opportuna composizione dei valori di colore ed opacità dei voxel colpiti, per poter visualizzare opportune trasparenze.

Ray Casting (1)

- E' più grezzo del Ray Tracing:
  - Non considera raggi ombra;
  - Non considera raggi riflessi;
- Ma anche più potente:
  - Funzione di composizione di opacità e colore più complessa;
- Segue però lo stesso procedimento:
  - Per ogni pixel si proietta un raggio verso il volume (back-to-front oppure front-to-back);
  - Ogni voxel attraversato ha 8 vertici i cui valori di colore ed opacità vanno interpolati per ottenere un unico valore;
  - Tale valore viene composto con quelli precedenti (o successivi) e poi visualizzato



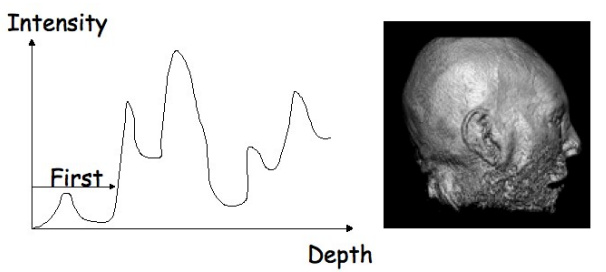
"www.cs.utk.edu"

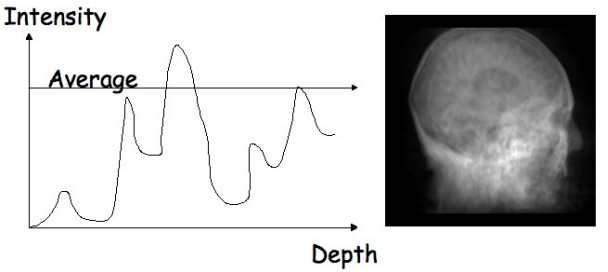


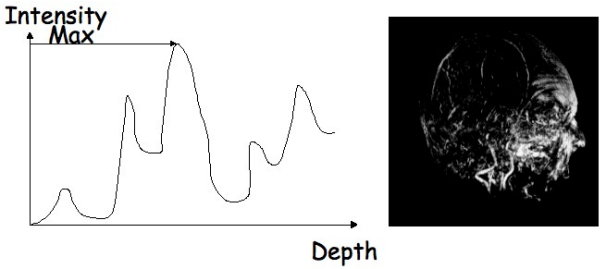
"www.cs.utk.edu"

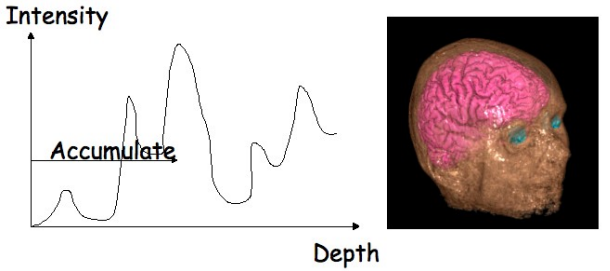
Ray Casting (2)

- Ci sono diversi modi per comporre colore ed opacità:
  - Ray Traversal First (isosurface-like)
  - Ray Traversal Average (XRay-like)
  - Ray Traversal MIP (Angiography-like)
  - Ray Traversal Accumulate (Transparency layers!)









"www.cs.utk.edu"

Volume Rendering Equation (VRE)

- Si pone dunque il problema di come accumulare colore ed opacità, che sappiamo essere in funzione della luce;
- Si usa una semplificazione dell' equazione del trasporto di Boltzmann (in forma integrale):

Integration path

$$L(\mathbf{r},\vec{\omega}) = e^{-\tau(\mathbf{r},\mathbf{r_B})} L_B(\mathbf{r_B},\vec{\omega}) + \int_{\Gamma(\mathbf{r},\mathbf{r_B})} e^{-\tau(\mathbf{r},\mathbf{r}')} Q(\mathbf{r}',\vec{\omega})\, dr'$$

"fast volume rendering using a shear-warp factorization of the viewing transformation"

$$\tau(\mathbf{r},\mathbf{s}) \equiv \int_{\Gamma(\mathbf{r},\mathbf{s})} \phi_t(\mathbf{r}')\, dr'$$

Extinction factor

$$Q(\mathbf{r},\vec{\omega}) \equiv \epsilon(\mathbf{r},\vec{\omega}) + \int_{S^2} k(\mathbf{r},\vec{\omega}' \rightarrow \vec{\omega}) L(\mathbf{r},\vec{\omega}')\, d\omega'$$

Emission + scattering

Boundary condition

- Ovvero la Volume Rendering Equation:

$$L(x) = \int_x^{x_B} e^{-\int_x^{x'} \phi_t(x'')dx''} \epsilon(x')dx'$$

Per la quale si fanno 4 ipotesi:
- Rifrazione singola;
- No dispersione tratto luce-voxel;
- Assorbimento indipendente dalla direzione;
- Sorgenti luce finite

Volumetric Compositing Approximation (α-compositing) - 1

- Il Ray Casting ci fornisce, per ogni voxel, i valori di colore ed opacità (o trasparenza);
- Noi dobbiamo combinarli sequenzialmente risolvendo la VRE!
- Usiamo, ovviamente, un' approssimazione a sommatoria:

$$L(x) = \sum_{i=0}^{n-1} e^{-\sum_{j=0}^{i-1}\phi_j \Delta x} \cdot \epsilon_i \Delta x = \sum_{i=0}^{n-1} \epsilon_i \Delta x \cdot \prod_{j=0}^{i-1} e^{-\phi_j \Delta x}$$

$$\epsilon_i \equiv \epsilon(x + i\Delta x) \quad \text{emissione}$$

$$\phi_i \equiv \phi_t(x + i\Delta x) \quad \text{assorbimento}$$

- E definiamo colore ed opacità in funzione di emissione ed assorbimento:

$$\alpha_i \equiv 1 - e^{-\phi_i \Delta x}$$

$$C_i \equiv (\epsilon_i / \alpha_i) \cdot \Delta x$$

$$c_i \equiv C_i \alpha_i$$

Ray Casting!

$$\begin{aligned} L(x) &= \sum_{i=0}^{n-1} c_i \cdot \prod_{j=0}^{i-1} 1 - \alpha_j \\ &= c_0 + c_1(1-\alpha_0) + c_2(1-\alpha_0)(1-\alpha_1) + \cdots \\ &\quad + c_{n-1}(1-\alpha_0)\cdots(1-\alpha_{n-2}) \\ &= c_0 \text{ over } c_1 \text{ over } c_2 \text{ over } \cdots \text{ over } c_{n-1} \end{aligned}$$

"fast volume rendering using a shear-warp factorization of the viewing transformation"

Volumetric Compositing Approximation (α-compositing) - 2

$$
\begin{aligned}
L(x) &= \sum_{i=0}^{n-1} c_i \cdot \prod_{j=0}^{i-1} 1-\alpha_j \\
&= c_0 + c_1(1-\alpha_0) + c_2(1-\alpha_0)(1-\alpha_1) + \cdots \\
&\qquad + c_{n-1}(1-\alpha_0)\cdots(1-\alpha_{n-2}) \\
&= c_0 \text{ over } c_1 \text{ over } c_2 \text{ over } \cdots \text{ over } c_{n-1}
\end{aligned}
$$

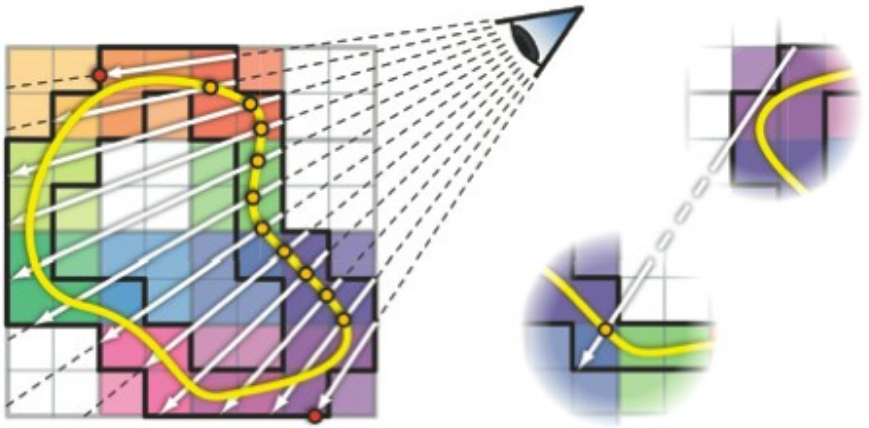
"fast volume rendering using a shear-warp factorization of the viewing transformation"

- Se ora inseriamo nel Ray Casting la formula appena trovata per la composizione di colore ed opacità, otteniamo un rendering volumetrico capace di sovrapporre diverse superfici a livelli di trasparenza!
- L' algoritmo Ray Traversal Accumulate è quindi:
  - Per ogni pixel, genera un raggio attraverso la scena 3D;
  - Per ogni voxel attraversato preleva i valori di colore ed opacità (per interpolazione tra i vertici);
  - Combina colore ed opacità trovati secondo la formula α-compositing

"www.imagingeconomics.com"

Strutture dati (1)

- Solitamente, nemmeno la metà della mole di dati volumetrici a disposizione è effettivamente importante;
  - In una tomografia o in una risonanza siamo sul 40%;
  - Nell' ipotesi interessino solo i vasi sanguigni (angiografia) scendiamo al 10%!
- Ciò significa che di tutti i voxel che compongono il volume, più della metà non interessano, quindi:
  - o avranno il colore di sfondo;
  - o saranno trasparenti.

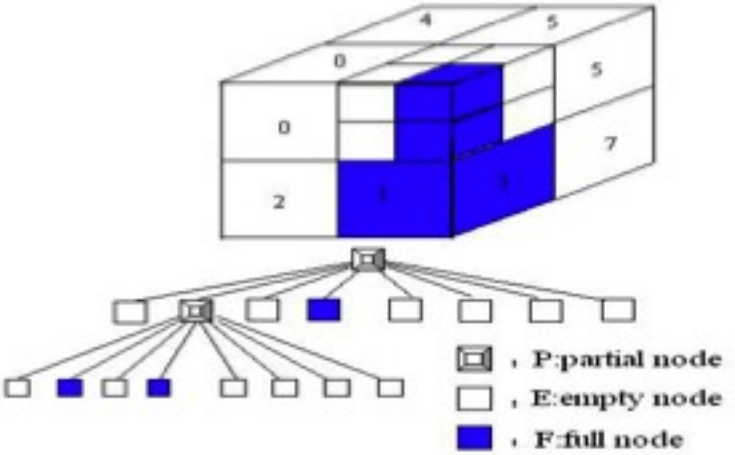


"Advanced Illumination Techniques for GPU-Based Volume Raycasting"

- Il Ray Casting è tutt' altro che efficiente: la maggior parte dei raggi generati (uno per pixel!) intersecano voxel inutili, quindi causano un rendering inutile!

Strutture dati (2)

- Si possono migliorare le prestazioni dell' algoritmo adottando una rappresentazione dei voxel tramite octree:
  - L' intero volume viene partizionato in 8 cubi delle stesse dimensioni;
  - Per ogni cubo, se esso è completamente pieno, lo si marca con F (Full), mentre se è completamente vuoto lo si marca con E (Empty);



"Advanced Illumination Techniques for GPU-Based Volume Raycasting"

  - Sempre per ogni cubo, se esso non è né marcato E né F si ripete il partizionamento, altrimenti ci si ferma e si passa ad un altro cubo.
- In questo modo possiamo facilmente saltare il rendering dei cubi vuoti (si noti che un cubo vuoto può comprendere centinaia di voxel!):
  - Per ogni raggio controlliamo se esso interseca un cubo vuoto (conosciamo le coordinate dei voxel che ci stanno dentro) guardando nell' albero dei cubi (l' octree);
  - Se troviamo che tale cubo è marcato E non ne facciamo il rendering e passiamo al prossimo raggio (pixel).

Librerie VTK

- Tutto ciò è molto bello, ma come lo faccio!? ☺

"www.vtk.org"