Ricostruzione tridimensionale di immagini

Maria Grazia Di Bono

Introduzione

Il problema della ricostruzione di una superficie tri-dimensionale (3-D), dato un insieme di contorni su piani (2-D) è molto importante in diversi campi. Per esempio, in campo biologico, dove i biologi tentano di studiare e caratterizzare le forme di oggetti microscopici, da una serie di sezioni successive dell'oggetto. Nel campo della medicina clinica, i dati generati attraverso le più moderne tecniche diagnostiche (TAC, ultrasuoni, NMR, ecc.) offrono una serie di slice successive che descrivono l'oggetto di studio, o ancora questo problema trova un campo fertile nella ricostruzione di modelli 3D, per la visualizzazione di terreni, a partire da una serie di contorni estratti dalle carte topografiche.

Ci sono essenzialmente due tipologie di approcci :

- *Approcci basati sui volumi*
- *Approcci basati sulle superfici*

Per gli approcci del primo tipo si fa l'assunzione che i dati siano disponibili come una griglia tri-dimensionale, mentre, per quelli del secondo tipo, i dati sono definiti come l'intersezione tra una superficie ed un piano delle varie sezioni.

Ovviamente, quale dei due approcci sia il più applicabile, dipende dalla natura dei dati. Nel nostro caso, ci troviamo di fronte ad un approccio basato sulle superfici. I dati di input, consistono in una serie di sezioni, ciascuna contenete uno o più contorni, codificate, come detto precedentemente, in un insieme di file con formato ".vet" già disponibili.

Diamo le seguenti definizioni :

Def. 1. Si definisce un *contorno*, come un semplice poligono che rappresenta l'intersezione della superficie di un oggetto ed il piano di una sezione.

Def. 2. Si definisce una *sezione*, come l'insieme dei contorni formati da specifiche regioni di interesse di una slice. I contorni di una sezione,

non appartengono necessariamente ad uno stesso oggetto, inoltre esso può essere rappresentato da più di un contorno in una sezione. Si assume che i contorni siano dei poligoni semplici e che nessuno di essi, ne intersechi un altro del data-set.

Il problema della generazione di una superficie da un insieme di contorni, può essere suddiviso in diversi sottoproblemi:

- Il problema della *"Corrispondenza"*, la cui risoluzione consiste nel determinare le relazioni di adiacenza topologica tra i diversi contorni del data-set.

- Il problema del *"Tiling"*, ossia il problema della ricostruzione della superficie dell'oggetto, una volta determinate le corrispondenze dei contorni adiacenti nel data-set. Tale problema è risolto, generando le migliori relazioni di adiacenza topologica tra i punti appartenenti a coppie di contorni di sezioni adiacenti, costruendo una mesh triangolare tra questi punti.

- Il problema del *"Branching"*, che nasce nel momento in cui un oggetto è rappresentato da un differente numero di contorni in sezioni adiacenti. In questi casi, i metodi standard per la risoluzione del problema del Tiling, non possono essere usati direttamente.

- Il problema del *"Surface-Fitting"*, ossia l'adattamento della migliore superficie alla mesh generata dopo la risoluzione dei precedenti problemi. Una soluzione a tale problema produce una dettagliata descrizione della geometria della superficie così ricostruita.

Ricostruzione tridimensionale di immagini

Maria Grazia Di Bono

Introduzione

Il problema della ricostruzione di una superficie tri-dimensionale (3-D), dato un insieme di contorni su piani (2-D) è molto importante in diversi campi. Per esempio, in campo biologico, dove i biologi tentano di studiare e caratterizzare le forme di oggetti microscopici, da una serie di sezioni successive dell'oggetto. Nel campo della medicina clinica, i dati generati attraverso le più moderne tecniche diagnostiche (TAC, ultrasuoni, NMR, ecc.) offrono una serie di slice successive che descrivono l'oggetto di studio, o ancora questo problema trova un campo fertile nella ricostruzione di modelli 3D, per la visualizzazione di terreni, a partire da una serie di contorni estratti dalle carte topografiche.

Ci sono essenzialmente due tipologie di approcci :

- *Approcci basati sui volumi*
- *Approcci basati sulle superfici*

Per gli approcci del primo tipo si fa l'assunzione che i dati siano disponibili come una griglia tri-dimensionale, mentre, per quelli del secondo tipo, i dati sono definiti come l'intersezione tra una superficie ed un piano delle varie sezioni.

Ovviamente, quale dei due approcci sia il più applicabile, dipende dalla natura dei dati. Nel nostro caso, ci troviamo di fronte ad un approccio basato sulle superfici. I dati di input, consistono in una serie di sezioni, ciascuna contenete uno o più contorni, codificate, come detto precedentemente, in un insieme di file con formato ".vet" già disponibili.

Diamo le seguenti definizioni :

Def. 1. Si definisce un *contorno*, come un semplice poligono che rappresenta l'intersezione della superficie di un oggetto ed il piano di una sezione.

Def. 2. Si definisce una *sezione*, come l'insieme dei contorni formati da specifiche regioni di interesse di una slice. I contorni di una sezione,

non appartengono necessariamente ad uno stesso oggetto, inoltre esso può essere rappresentato da più di un contorno in una sezione. Si assume che i contorni siano dei poligoni semplici e che nessuno di essi, ne intersechi un altro del data-set.

Il problema della generazione di una superficie da un insieme di contorni, può essere suddiviso in diversi sottoproblemi:

- Il problema della *"Corrispondenza"*, la cui risoluzione consiste nel determinare le relazioni di adiacenza topologica tra i diversi contorni del data-set.

- Il problema del *"Tiling"*, ossia il problema della ricostruzione della superficie dell'oggetto, una volta determinate le corrispondenze dei contorni adiacenti nel data-set. Tale problema è risolto, generando le migliori relazioni di adiacenza topologica tra i punti appartenenti a coppie di contorni di sezioni adiacenti, costruendo una mesh triangolare tra questi punti.

- Il problema del *"Branching"*, che nasce nel momento in cui un oggetto è rappresentato da un differente numero di contorni in sezioni adiacenti. In questi casi, i metodi standard per la risoluzione del problema del Tiling, non possono essere usati direttamente.

- Il problema del *"Surface-Fitting"*, ossia l'adattamento della migliore superficie alla mesh generata dopo la risoluzione dei precedenti problemi. Una soluzione a tale problema produce una dettagliata descrizione della geometria della superficie così ricostruita.

Nel nostro caso, i problemi della corrispondenza e del branching sono risolti a priori, poiché si dispone già in partenza di tali informazioni (etichetta volume, connessioni dei contorni), nei file di input "*.vet". Rimane quindi da affrontare, solo il problema del tiling, per costruire il modello geometrico degli oggetti in esame, necessario per la definizione della rete neurale.

Panoramica sugli algoritmi di ricostruzione di superfici da contorni

Il problema della ricostruzione di superfici da un insieme di contorni, è stato oggetto di molti lavori, già a partire dagli anni settanta.

Per primo Keppel [Kepp75], ridusse il problema di trovare punti corrispondenti in contorni successivi, ad un problema di ricerca su un grafo a forma di toroide, come illustrato nella figura 1.

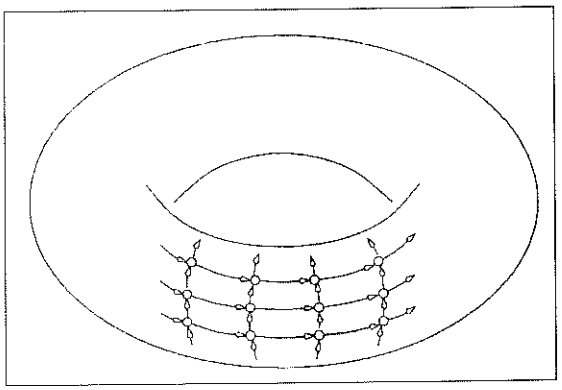
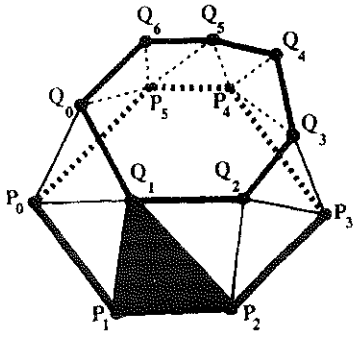


Figura 1. Porzione di grafo rappresentante le connessioni cilindriche tra due contorni.

Successivamente Fuchs ed altri [Fuchs77], fecero un'analisi estensiva del problema di ricerca e svilupparono un efficiente metodo per il problema. In questo metodo, i contorni sono rappresentati da liste ordinate di punti. Gli archi che connettono punti vicini sullo stesso contorno sono denominati *"segmenti del contorno"*, mentre gli archi che connettono punti di un contorno con punti dell'altro contorno, sono chiamati *"corde"*. Il metodo si sviluppa associando un nodo del grafo a

ciascuna corda. Il grafo quindi risulta essere una densa griglia bidimensionale, sovrapposta ad un toroide. Un arco nel grafo definisce un triangolo formato da un segmento appartenente ad un contorno, connesso con due corde ad un punto appartenente all'altro contorno. Agli archi del grafo, sono associati dei costi, definiti da una funzione calcolabile in base al triangolo che essi definiscono. In questo modo, una superficie che connette due sezioni adiacenti, è rappresentata da un ciclo di costo ottimo nel grafo, come si può vedere in figura 2.



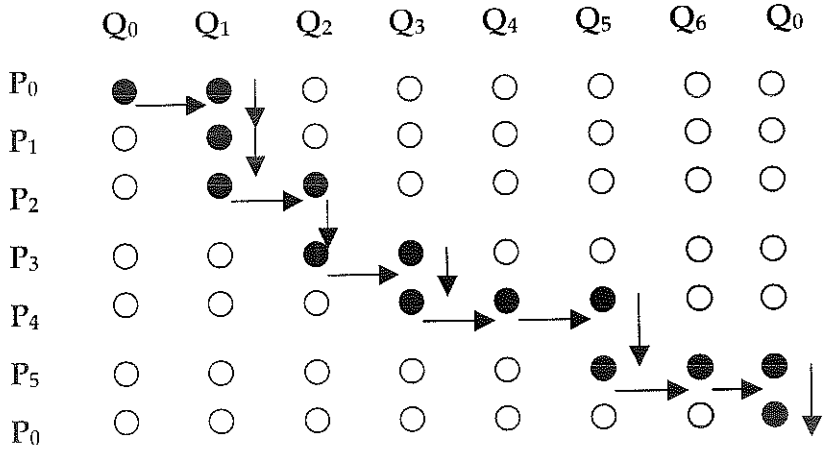


Figura 2. Problema del Tiling espresso come un problema di ricerca su grafi.

Molte sono le superfici che si possono generare per due sezioni adiacenti, ma per scegliere una superficie "corretta", si ottimizza rispetto ad una funzione obiettivo.

Sono stati utilizzati molti approcci per risolvere tale problema, differenziandosi l'uno dall'altro nella scelta della funzione obiettivo da

ottimizzare. Le più importanti funzioni si possono sintetizzare come segue:

- *Volume*. La funzione obiettivo (f.o.) è definita dal volume totale dell'oggetto, che viene massimizzato. Utilizzata da Keppel [Kepp75], tale metrica è ovvia per oggetti convessi, ma non produce buoni risultati per oggetti che presentano regioni concave. Inoltre risulta anche difficile da usare.

- *Area*. In questo caso, la f.o. è definita dall'area della triangolazione che si sta generando, che viene minimizzata. Tale metrica, utilizzata da Fuchs ed altri [Fuchs77], è semplice da implementare e produce buoni risultati, se usata in concomitanza alla normalizzazione dei contorni.

- *Lunghezza delle corde*. Christiansen e Sederberg [Chri78] descrivono un metodo "greedy" basato sulla minimizzazione della lunghezza delle corde, che, per effettuare la ricostruzione 3D, non usa il metodo di ricerca sul grafo.

- *Corrispondenze delle direzioni*. Cook ed altri [Cook80], hanno usato un metodo basato sul far corrispondere le direzioni dei punti dal centro di massa del rispettivo contorno.

Descrizione degli algoritmi implementati

Il problema della ricostruzione della superficie di un oggetto, data una serie di slice parallele, può essere semplificato, considerando solo una singola coppia di slice successive. In seguito, l'unione o la concatenazione dei vari modelli generati, rappresenterà il modello soluzione di tutto il problema.

Esaminiamo quindi il seguente problema:

Dati due contorni, ossia due poligoni P e Q in piani paralleli in R^3, si vuole connettere P con Q, il ché significa costruire un "cilindro triangolato" incollato a P su un lato ed a Q sull'altro.

In un linguaggio topologico, il cilindro è una "omotopia" tra P e Q.

Ciascun triangolo connette un segmento di P con un vertice di Q e viceversa. Lungo questi archi, un triangolo è connesso al predecessore ed al successore, intorno al cilindro, come si può vedere in figura 3.

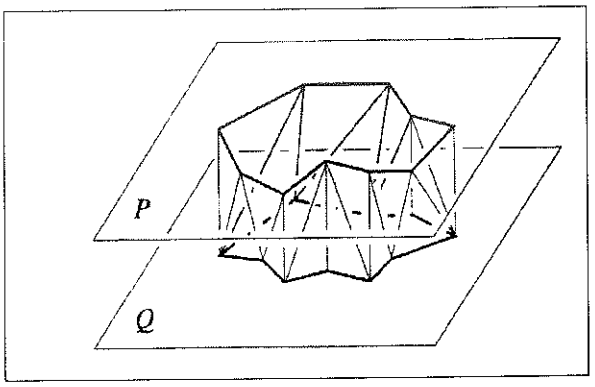


Figura 3. Connessioni cilindriche tra due contorni.

In generale, non è detto che i due poligoni abbiano lo stesso numero di vertici, pertanto è stato considerato il caso in cui P abbia i vertici etichettati da *0* a *m-1*, mentre Q li abbia da *0* a *n-1*, con $m \neq n$. In effetti capita molto di rado ottenere, dopo la segmentazione, contorni relativi allo stesso oggetto, con lo stesso numero di vertici.

Una volta formalizzato il problema, si sono considerati due particolari approcci, l'uno prendendo spunto dal metodo greedy basato sulla minimizzazione della lunghezza delle corde, l'altro basandosi sulla ricerca su grafi, con la minimizzazione dell'area.

1. Algoritmo greedy "Min Arc Length"

L'algoritmo Min Arc Length, prende come input la matrice dei vertici *MV*, che codifica l'informazione relativa ai vertici dei contorni di ogni sezione. Tale matrice è costruita leggendo le coordinate bidimensionali *(x,y)* di tali vertici dai file binari *.vet, ed aggiungendo la terza coordinata *z* in base al numero di sezione (slice) esaminata. Inoltre viene aggiunta anche l'informazione riguardante l'etichetta del contorno di appartenenza di ogni punto. In questo modo, ogni oggetto da ricostruire è identificato da una etichetta. L'output dell'algoritmo consiste in una matrice *MF*, che rappresenta la matrice delle facce che descrivono la mesh *M* ottenuta.

La messa a punto di questo algoritmo di ricostruzione 3D, si è svolta attraverso l'implementazione delle seguenti quattro funzioni :

- *Ricostruction_3D(MV, slice)*
- *start_point(vet1, vet2)*
- *Triangulation(ind1, ind2, vindex1, vindex2, vet1, vet2)*
- *triangle(vindex1, vindex2, vet1, vet2)*

La funzione ***ricostruction_3D***, è la funzione principale, ossia il suo input ed il suo output coincidono con quelli dell'intero algoritmo. In

dettaglio, essa ha come parametri di input la matrice dei vertici *MV* e la variabile *slice*, che è una matrice di strutture che per ogni contorno e per ogni sezione, contiene l'informazione sull'indice, nella matrice *MV*, del primo vertice del contorno in esame e del numero di vertici che vi appartengono. La matrice *slice*, avrà quindi tante righe quante le slice del data-set esaminato e tante colonne quanto il massimo numero di contorni nelle sezioni.

La funzione *ricostruction_3D*, per ogni oggetto da ricostruire, seleziona i vertici appartenenti a coppie di contorni corrispondenti su slice successive. In seguito, per ogni coppia, essa richiama al suo interno le funzioni *start_point e Triangulation*. La prima, ha il compito di individuare i primi due vertici corrispondenti *ind1* e *ind2*, sui due contorni in esame, in base ad un certo criterio che sarà discusso in seguito. La seconda, si occupa di calcolare la matrice delle facce per i due contorni in esame, ordinando i vettori dei vertici dei due contorni a partire dai punti iniziali corrispondenti e chiamando al suo interno la funzione *triangle*, che effettua la triangolazione in base al criterio prestabilito.

Le matrici parziali, ottenute per ogni coppia di slice successive, saranno infine concatenate per formare la matrice totale delle facce in grado di descrivere l'oggetto considerato.

Il diagramma di flusso della funzione *ricostruction_3D* è riportato in figura 3.6.

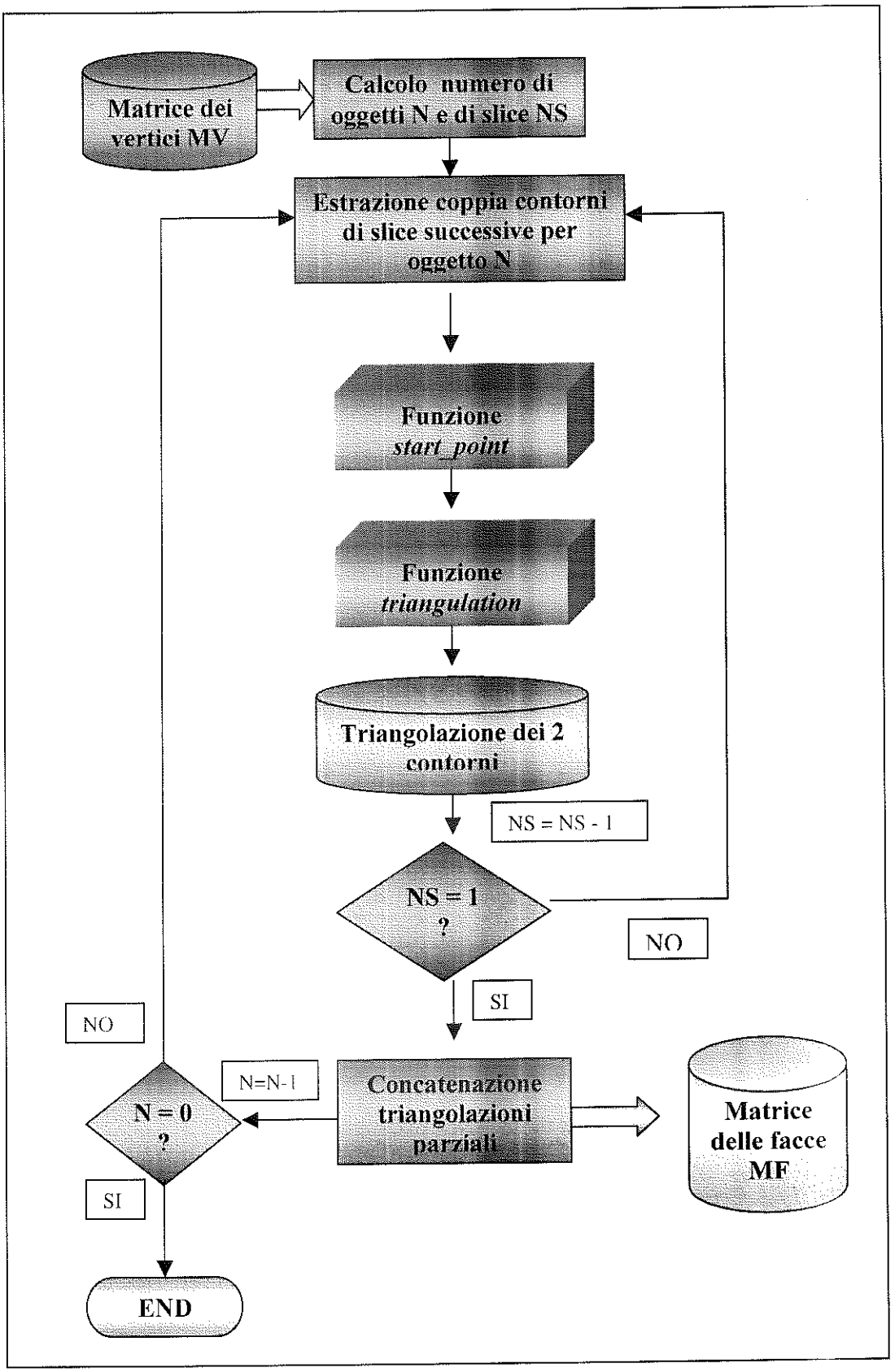


Figura 4. Diagramma di flusso della funzione ricostruction_3D.

La funzione ***start_point***, ha come input i due vettori (vet1 e vet2) di coordinate nello spazio (x,y,z) dei vertici appartenenti ai due contorni in esame e restituisce la posizione dei primi due punti corrispondenti, da cui partire per generare la triangolazione. Inizialmente, si calcolano i centroidi dei due vettori. Successivamente, si calcola l'asse di minima inerzia del primo contorno e si approssima l'intersezione tra tale asse ed il contorno, cercando il punto (x_i,y_i) più vicino, nel piano, al vettore (x_a,y_a) che descrive l'asse di minima inerzia. Allo stesso modo, si approssima l'intersezione tra l'asse di minima inerzia trovato ed il secondo contorno, determinando il punto (x_j,y_j) corrispondente. Si restituiscono quindi le posizioni dei punti trovati, che saranno usate per etichettare i vertici iniziali per la triangolazione tra i due contorni. Una descrizione più dettagliata degli aspetti matematici per il calcolo dell'asse di minima inerzia è riportata in appendice B. In figura 5. è riportato il diagramma di flusso che descrive la funzione *start_point*.

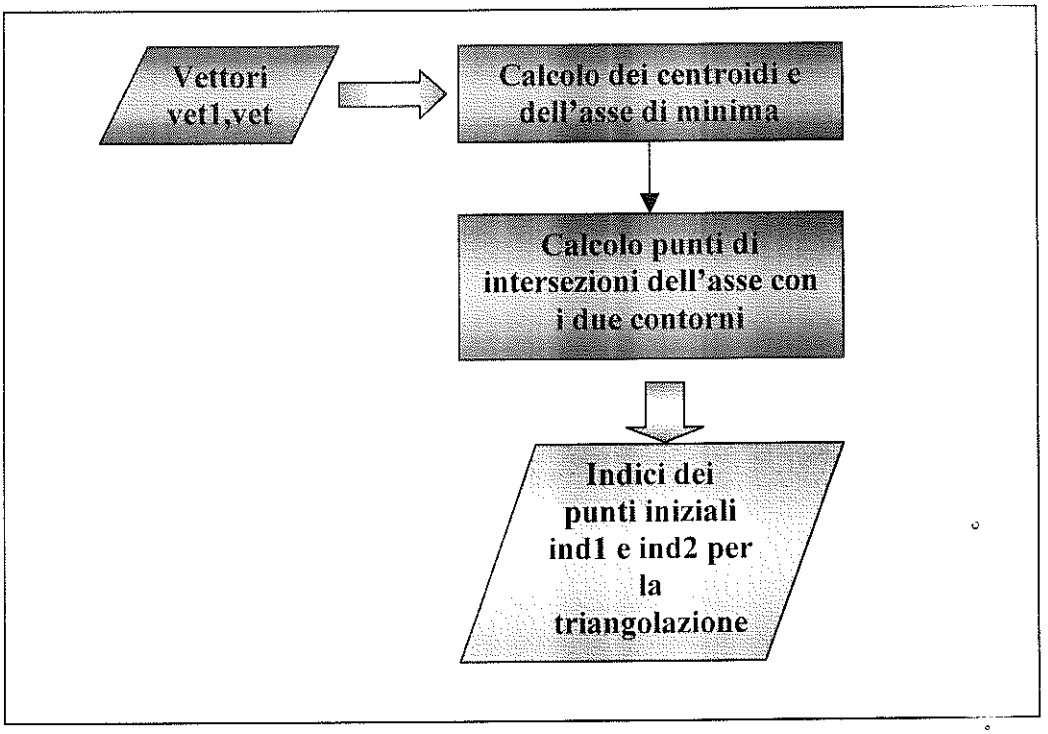


Figura 5. Diagramma di flusso della funzione *start_point*.

La funzione ***triangulation*** riceve in input i vettori dei vertici, che descrivono i due contorni, e gli indici dei punti iniziali, si occupa di ordinare tali vettori a partire dai vertici iniziali e richiama al suo interno la funzione *triangle* che si occuperà di generare la matrice delle facce per i due contorni. L'ultimo passo consiste nell'eliminazione di eventuali facce di area nulla, dovute ad eventuali swap.Il diagramma di flusso per tale funzione è illustrato in figura 6.

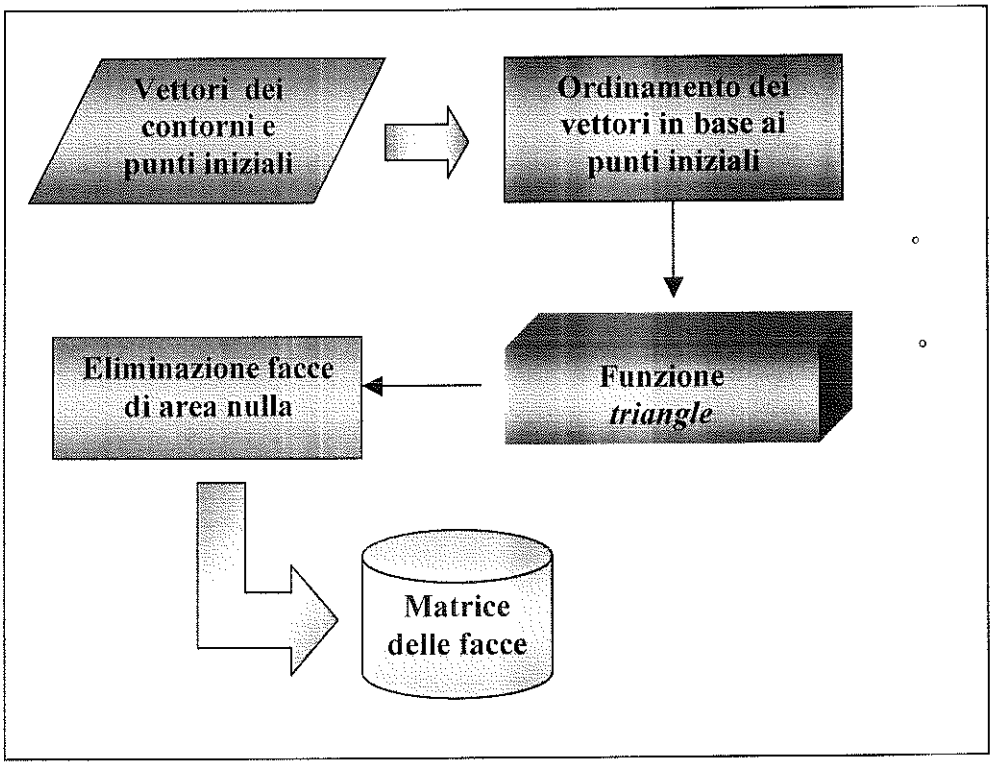


Figura 6. Diagramma di flusso per la funzione *triangulation*.

La funzione ***triangle***, riceve in input i vettori degli indici e delle coordinate (x,y,z) dei vertici che descrivono i due contorni. Inizialmente, si settano i punti correnti (i_1 ed i_2), ossia gli ultimi due vertici in esame che al primo passo saranno i punti iniziali, ed il contorno corrente (cc), ossia il contorno di appartenenza dell'ultimo punto esaminato. Il passo successivo dell'algoritmo consiste nel calcolo della lunghezza tra ciascun punto corrente ed il punto successivo all'altro, ossia si calcola la lunghezza delle due possibili corde da aggiungere per la formazione del triangolo corrente.

Tra le due corde possibili, si sceglie quella di lunghezza minima, si aggiornano i punti e lo slice correnti. Gli indici selezionati, sono inseriti di volta in volta in un vettore di ordinamento, che tiene traccia del percorso dal primo all'ultimo vertice da collegare. Successivamente, la matrice delle facce è ottenuta concatenando, tre per volta, gli elementi del vettore di ordinamento, ossia ogni faccia è determinata dagli elementi del vettore in posizione *(i, i+1, i+2)* con *i=1:L-2*, ed *L* che rappresenta la lunghezza di tale vettore. La matrice delle facce così ottenuta, può avere delle facce di area nulla, in quanto si possono presentare delle situazioni di *"swap"*.

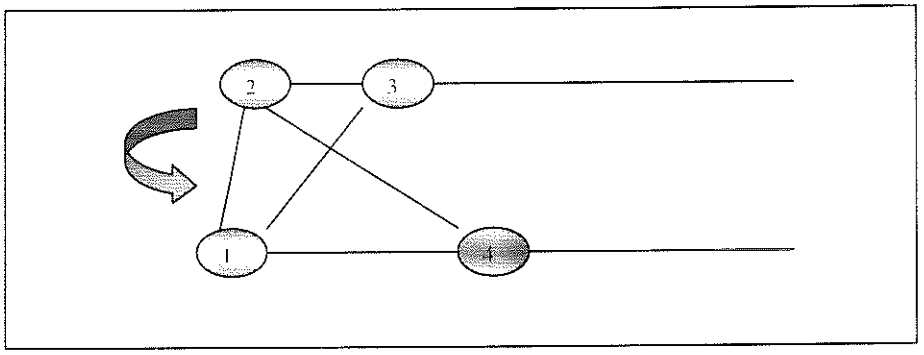


Figura 7. Esempio di swap.

Come si vede in figura 7, se si parte dal contorno in basso, i primi due elementi nel vettore di ordinamento saranno (1, 2). Se la corda di minima lunghezza fosse la (2, 4), nel vettore si aggiungerebbe il vertice 4, ottenendo il triangolo [1, 2, 4], ma in realtà la corda di lunghezza minima è la (1, 3). Poiché un triangolo è definito da due corde ed un segmento, la successione degli indici nel vettore ordinamento deve definire due corde e non una corda ed un segmento. Pertanto si torna indietro al vertice 1, inserendo nel vettore ordinamento il vertice 1 ed il nodo 3 scelto, ottenendo come risultato (1, 2, 1, 3), che genera le facce f1=[1,2,1] , di area nulla, e f2=[2,1,3]. Le facce di area nulla saranno eliminate nella funzione *triangulation*. Il diagramma di flusso della funzione *triangle* è rappresenato in figura 8.

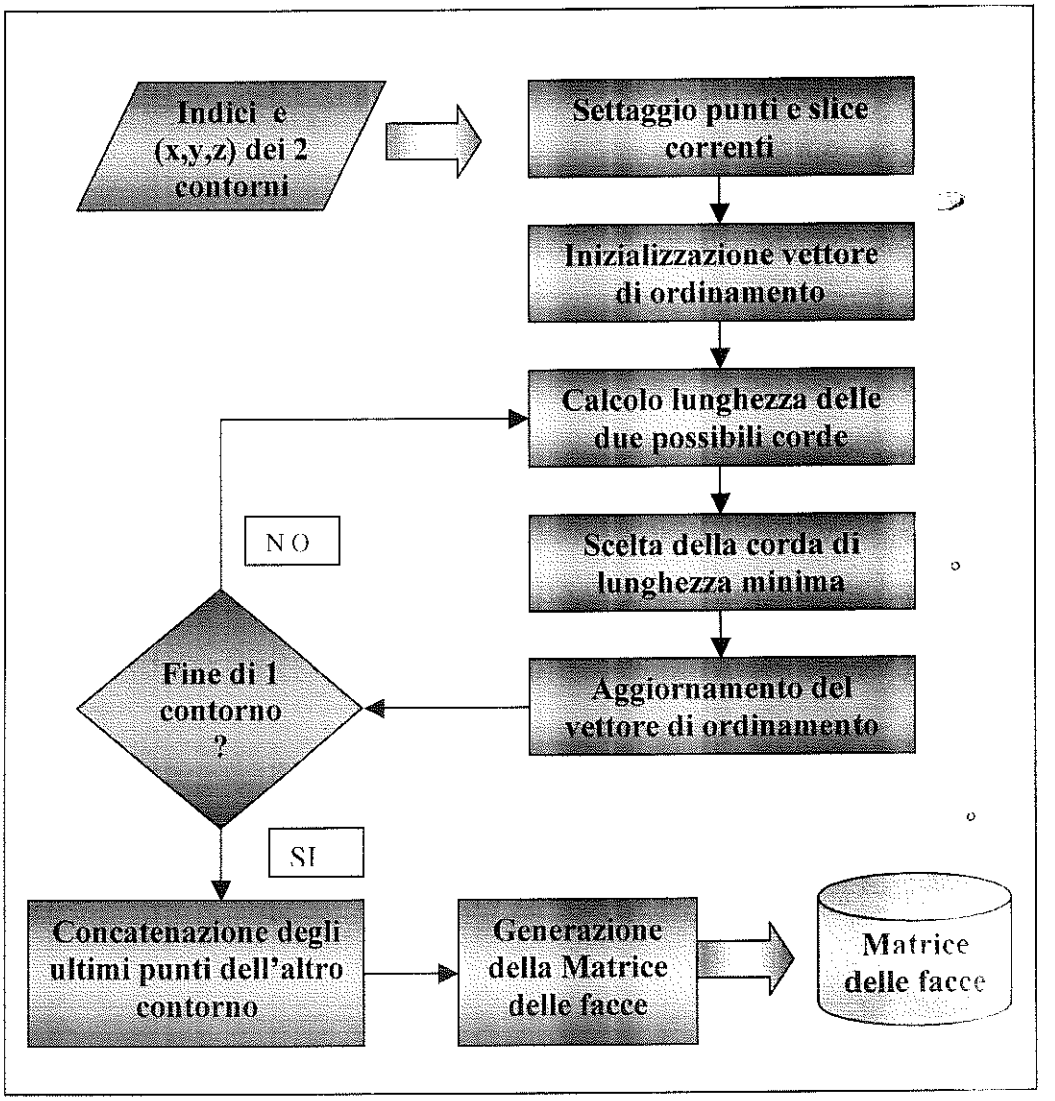


Figura 8. Diagramma di flusso della funzione *triangle*.

L'algoritmo è stato testato su diversi insiemi di file *.vet, relativi sia alla faccia, al cervello, occhi, che ai ventricoli ed alla massa tumorale di un paziente in esame ed ha prodotto risultati soddisfacenti.

2. Algoritmo di ricerca su grafi "Min Area"

Tale algoritmo, si basa su un tipo di approccio diverso dal precedente. Il problema della triangolazione viene ricondotto ad un problema di ricerca su grafi, come precedentemente descritto nella sezione 3.3.1. Il cilindro di area minima tra tutti i cilindri possibili, può essere calcolato, con tecniche di programmazione dinamica in un tempo di $O(mn)$.

Le funzioni coinvolte per l'implementazione di questo algoritmo sono le seguenti:

- *Ricostruction_3D_ma(MV, slice)*
- *start_point(vet1, vet2)*
- *Triangulation_ma(ind1, ind2, vindex1, vindex2, vet1, vet2)*
- *Connecting_contours(vindex1, vindex2, vet1, vet2)*
- *Calcola_area(a,b,c,vet1,vet2,s)*

Le prime tre funzioni sono identiche, per funzionalità, alle prime tre del precedente algoritmo, mentre la funzione *triangulation_ma* richiama al suo interno la funzione *connecting_contours*, che a sua volta richiama la funzione *calcola_area*.

La funzione ***connecting_contours***, costruisce due matrici, l'una (A) per la memorizzazione del cacolo delle aree dei possibili triangoli, l'altra (C) per tener traccia dei possibili cammini.

In particolare, la matrice delle aree è costruita nel seguente modo : sia m il numero di vertici del primo contorno ed n quello del secondo, partendo dai punti iniziali corrispondenti, l'algoritmo cerca per ogni nodo (i,j) la minima area totale $A_{i,j}$ del percorso della triangolazione parziale dal nodo $(0,0)$ iniziale al nodo (i,j) come segue:

for $i = 0 : m$

  for $j = 0 : n$

    $A_{i,j} = min\left\{A_{i-1,j} + area(i-1, i-j), A_{i,j-1} + area(i, j-1, j)\right\}$

  end

end

Si fa l'assunzione che $A_{i,j} = 0$ e $area(i,j,k) = 0$ per qualche indice negativo. La minima area del cilindro contenente la corda dei due punti iniziali è data da $A_{m,n}$.

Nella matrice C, si tiene traccia dei possibili cammini sul grafo, con la convenzione che $C_{i,j} = 0$ se $A_{i,j}$ è ottenuta da $A_{i,j-1}$, mentre $C_{i,j} = 1$ se $A_{i,j}$ è ottenuta da $A_{i-1,j}$.

Si ricostruisce il cammino ottimo, ottenuto il quale, si ricostruisce la matrice delle facce, collegando i nodi attraversati dal cammino.

La funzione ***calcola_area***, dati i due vettori di coordinate, gli indici *(a, b, c)* dei vettori *(vet1 , vet2)* e lo slice di appartenenza di *b* e *c*, identificato dalla variabile *s*, calcola l'area del triangolo nello spazio, individuato dalle coordinate dei punti di indici *a, b, c.*

Nella figura 9, è illustrato il diagramma di flusso della funzione *connecting_contours* .

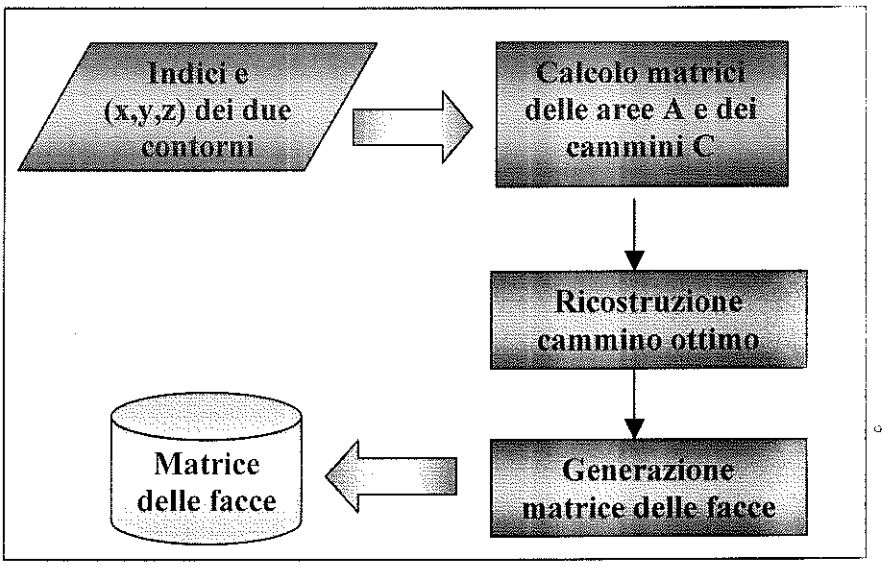


Figura 9. Diagramma di flusso della funzione *connecting_contours*.

Complessità computazionale degli algoritmi implementati

Per quanto riguarda gli algoritmi di triangolazione implementati, il primo "Min Arc Length" essendo un algoritmo greedy ha una complessità inferiore all'algoritmo di ricerca su grafi "Min Area", pertanto è stato scelto per le ricostruzioni effettuate.

Il primo, ha complessità data dalla seguente :

$$C = O(NS * \max(N, M))$$

dove:

- NS è il numero di slice disponibili
- N è il numero di vertici del primo contorno
- M è il numero di vertici del secondo contorno

Il secondo algoritmo, ha la seguente complessità :

$$C = O(NS * (N * M))$$

poiché la ricostruzione di ogni superficie relativa a ciascun coppia di contorni ha una complessità quadratica.

Risultati Sperimentali

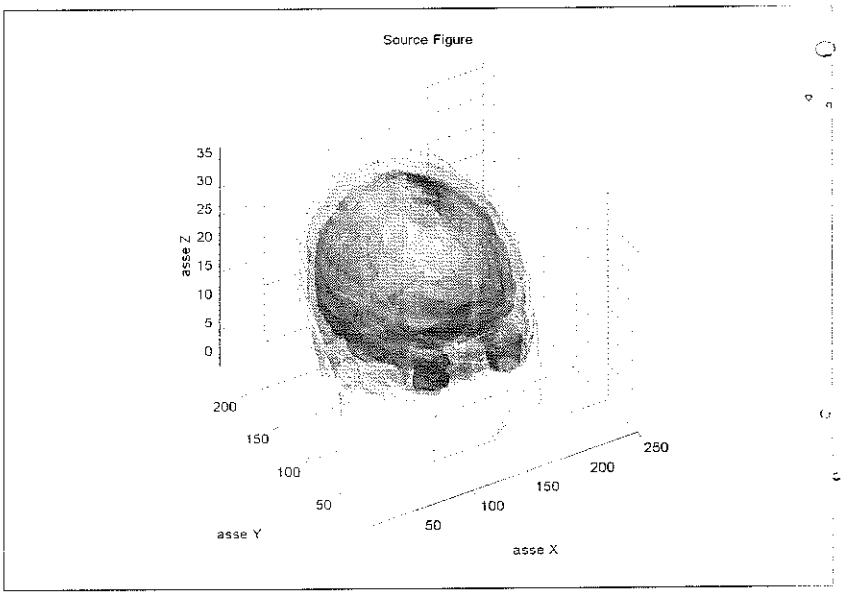


Figura 10. Immagine relativa alla ricostruzione tridimensionale dei volumi segmentati.

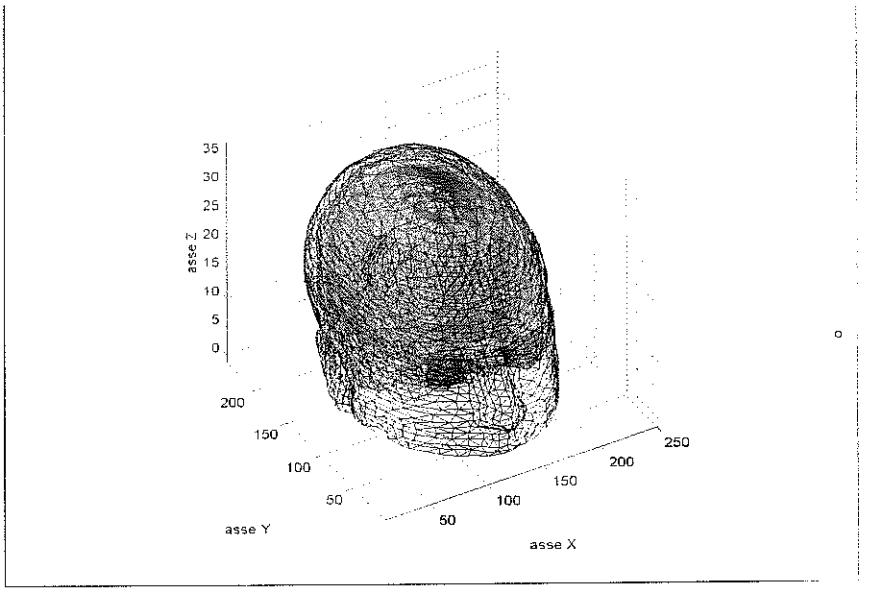


Figura 11. Mesh relative ai volumi ricostruiti.

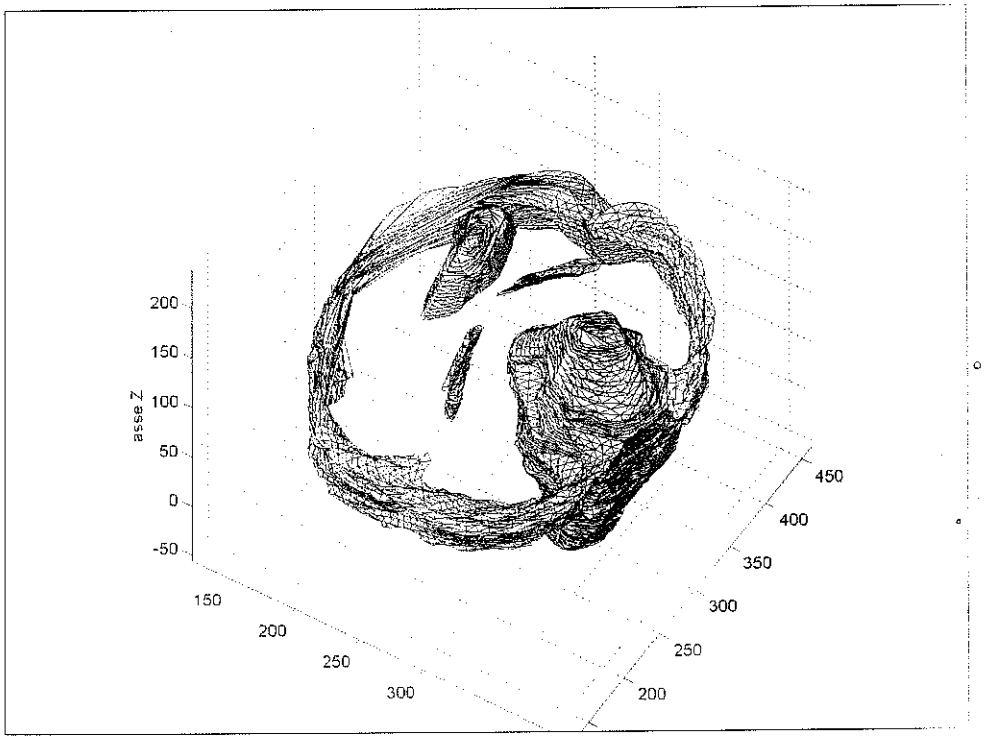


Figura 12. Ricostruzione 3D di una scena : sez. cervello, ventricoli e massa tumorale

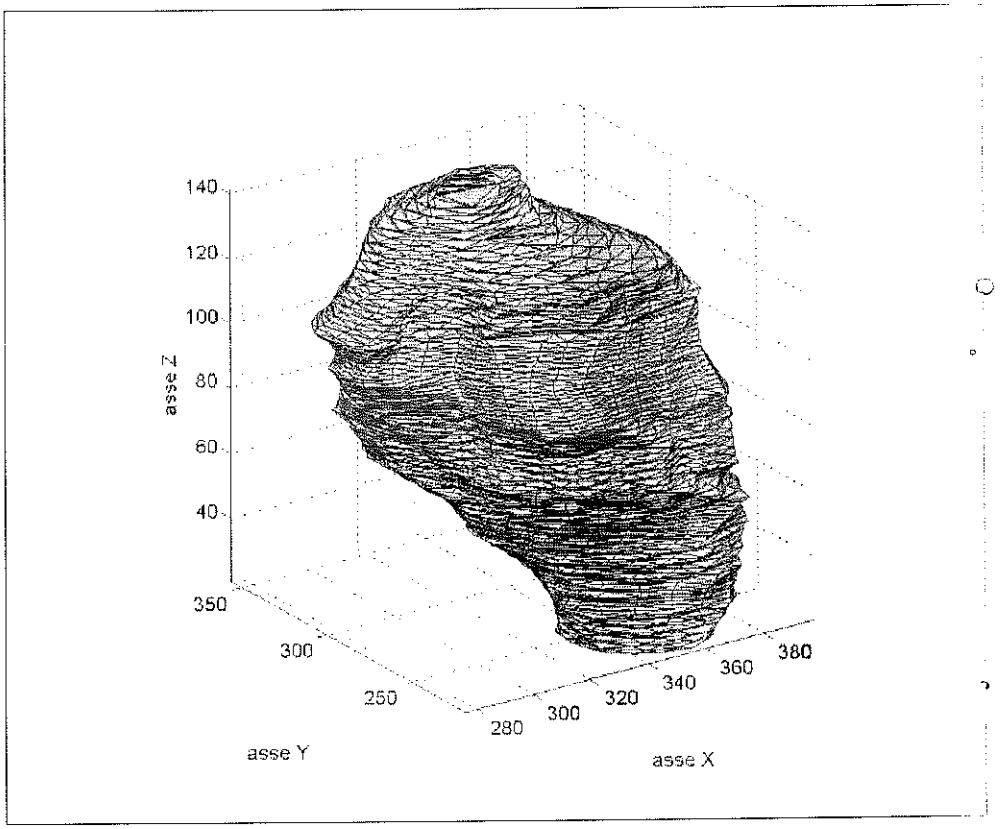


Figura 13. Ricostruzione 3D relativa ad una massa tumorale.

Bibliografia

[Mey92] **David Meyers, Shelley Skinner, Kenneth Sloan.** Surfaces from Contours. *ACM Transactions on Graphics, Vol. 11, No. 3, July 1992, pages 228-258.*

[Sloan87] **Kenneth R. Sloan, Jr. James Painter.** From Contours to Surfaces: Testbed and Results. *Department of Computer Science, University of Washington, 1987.*

[Hop92] **Hugues Hoppe, Tony De Rose, Tom Duchamp, John McDonald, Werner Stuetzle.** Surfece Recostruction rom Unorgzed Points. *ACM Transactions on Computer Graphics, 26, 2, Julay 1992.*

[Bareq94] **Gill Barequet, Micha Sharir.** Piecewise-Linear Interpolation between Polygonal Slice. *10th Computational Geometry 94-6/94 Stony Brook, NY, USA, 1994.*

[Kepp75] **Keppel E.** Approximating complex surfaces by triangulation of contou lines. *IBM J. Res. Dev. 19 (Jn. 1975), 2-11.*

[Fuchs77] **Fuchs H., Kedem Z. M., Uselton S. P.** Optimal surface reconstruction from planar contours. *Commun. ACM 20, 10 (Oct 1977), 693-702.*

[Chri78] **Christiansen H. N., Sederberg T. W.** Conversion of complex contourn line definition into polygonal element mosaics. *Computer graphics 12, 2 (Aug. 1978), 187-192.*

[Cook80] **Cook L. T., Cook P. N., Lee K. R., Batnitzky S., Wong B. Y. S., Fritz S. L., Ophir J., Dwyer S. J., Bigongiari L. R., Templeton A. W.** A algorithm for volume estimation based on polyedral approssimation. *IEEE Trans. Biomed. Eng. BME-27, 9 (Sept. 1980), 493-500.*

[Sloan81] **Sloan K. R., Hrechanyk L. M.** Surface reconstruction from sparse data. *In IEEE Conference on Pattern recognition and Image Processing (Aug.) 1981,* IEE, New York, pp. 5-48.