Laboratory for Advanced Planning
and Simulation project

Algoritmi di conversione
CSG - RayRep e RayRep - CSG

Relazione finale sul tirocinio formativo svolto da
Daniela Lecca

Tutor Aziendale
Dott. Piero Pili
Dott. Gregorio Franzoni

Tutor Universitario
Prof. Stefano Montaldo

Indice

Algoritmi di conversione CSG - RayRep e RayRep - CSG

Daniela Lecca

Agosto 2003

Abstract

Nell'ambito della realizzazione di modelli materiali di entità geometriche solide, con l'aumentare della complessità delle forme da realizzare stanno acquistando crescente importanza la *ricerca delle frontiere* degli oggetti e le *tecniche di riempimento automatico* del loro interno. Il presente lavoro costituisce un passo nella direzione descritta ed in particolare:

- si analizzano alcuni schemi di rappresentazione geometrica utili allo scopo: CSG, B-Rep, Ray-Rep;
- si propone l'idea di utilizzare la Ray-Rep per generare riempimenti su una particolare tecnica di produzione, la Fused Deposition Modelling;
- si sviluppano degli algoritmi di conversione dallo schema CSG allo schema Ray-Rep e viceversa, per la verifica dei risultati di riempimento su solidi campione.

Questo lavoro si inserisce all'interno della collaborazione scientifica in corso tra il CRS4 e il gruppo di ricerca del Dipartimento di Matematica dell'Università di Cagliari coordinato dal Prof. Renzo Caddeo, Prof. Stefano Montaldo e Prof.ssa. Paola Piu.

Parole chiave: Modellazione Geometrica Solida, Rapid Prototyping, Algoritmi di Conversione tra Schemi di Rappresentazione.

1 Introduzione

Il contesto in cui questo lavoro si inserisce è quello della modellazione fisica di oggetti attraverso tecniche di Prototipazione Rapida (in inglese Rapid Prototyping, spesso abbreviato

con la sigla RP) [14]. Con questo termine si intende un insieme di processi tecnologici rapidi, flessibili ed altamente automatizzati che, partendo da un modello matematico tridimensionale, realizzano modelli e componenti solidi strato dopo strato (layer by layer), per addizione di materiale, diversamente dalle tecniche tradizionali (tornio e fresa) che agiscono per asportazione di materiale da un pieno. Si usa spesso il termine *stampa* 3D riferendosi alla realizzazione mediante RP, e le macchine RP sono spesso chiamate stamparti 3D [14].

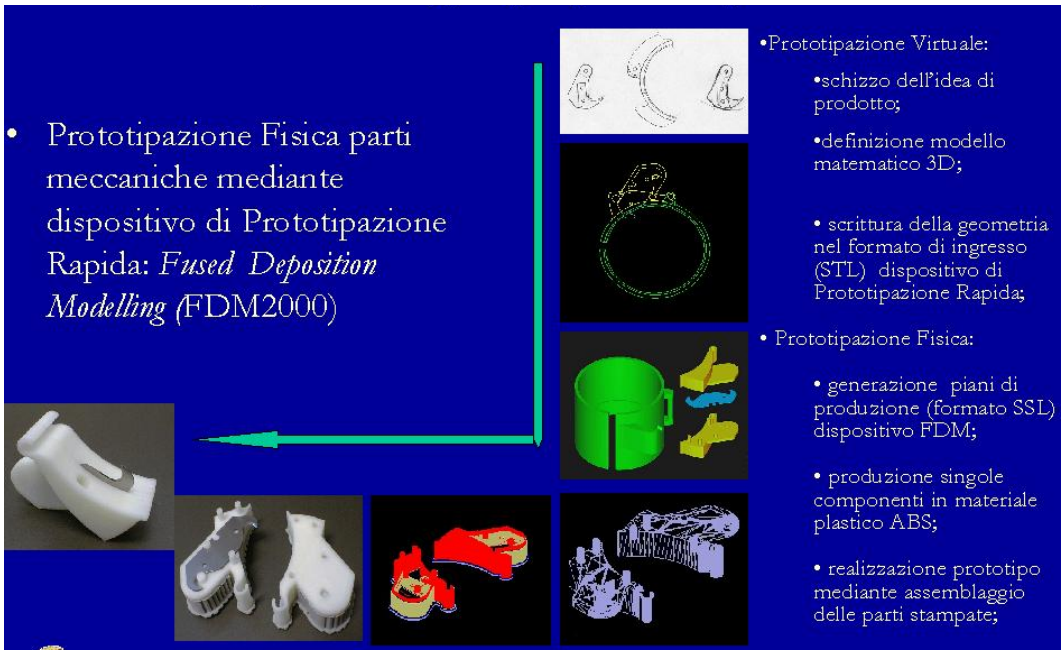

Figure 1: La pipeline operativa per la produzione fisica di un modello solido inizia con l'idea schizzata dal disegnatore. L'idea viene dettagliata con l'utilizzo di un CAD per generare un modello solido. Tale modello viene convertito in una rappresentazione adatta per il dispositivo di prototipazione. Questa rappresentazione viene poi elaborata per produrre le sezioni di stampa.

L'ambito di utilizzazione della RP è tipicamente quello della prototipazione industriale. In questo contesto, le rappresentazioni digitali degli oggetti da produrre fisicamente provengono da programmi CAD (Computer Aided Design), i quali provvedono anche ad una esportazione nel formato standard della RP, chiamato STL (StereoLitography)[3]. L'STL è un formato che memorizza una approssimazione del modello CAD non approssimato ottenuta attraverso una triangolazione della frontiera del solido. La tipica pipeline (si veda la Figura 1) seguita dalle tecnologie attualmente disponibili è la seguente:

- si parte da un modello solido prodotto al CAD o definito mediante nuvola di punti, acquisiti con tecnica di campionamento della frontiera (*reverse engineering*). In Figura

2 è illustrato il solido campione modellato al CAD;

- si costruisce una triangolazione della superficie del modello solido e la si esporta in formato STL. Tutti i CAD 3D dell'ultima generazione forniscono moduli per esportare il modello CAD in questo formato. La Figura 3 mostra la rappresentazione in formato STL del solido campione;
- si effettua un sezionamento (*slicing*) del modello triangolare che rappresenta la frontiera determinando (un insieme di linee spezzate chiuse) la frontiera sul piano di sezionamento. Sotto il controllo dell'utilizzatore vengono determinati eventuali *supporti* di stampa, che forniscono gli strati di appoggio dove il modello non è presente (sottosquadra) e che vengono eliminate a produzione ultimata. La Figura 4 mostra l'insieme delle sezioni orizzontali dello stesso solido dopo l'operazione di *slicing*, con i supporti necessari per la sua stampa 3D.;
- si procede alla determinazione dei percorsi utensile che realizzano il riempimento delle sezioni. Il formato prodotto al termine di questa fase è quello che viene inviato al prototipatore e contiene tutte le informazioni necessarie alla produzione fisica del pezzo. I percorsi sono chiamati *roads*, nel caso di deposizione di filamenti (Fused Deposition Modeling);
- si effettua la stampa e al termine di questa si asportano i supporti dal pezzo stampato.

La Figura 3 mostra la rappresentazione in formato STL del solido campione.

Generalizzando il problema della produzione automatica di modelli con tecniche di Prototipazione Rapida, occorre:

- determinare la frontiera del modello;
- riempire l'interno;
- generare i supporti.

Le strutture dati e gli algoritmi per rivolvere questa tipologia di problemi ricadono nella disciplina conosciuta come Modellazione Geometrica Solida che viene discussa nella prossima sezione.

2 Modellazione geometrica solida

Per poter realizzare fisicamente un oggetto solido, si deve disporre di un suo modello matematico valido, ovvero di una sua rappresentazione che definisca in maniera non ambigua l'interno e l'esterno del solido stesso (si vedano [7] e [8] per dettagli sull'argomento). In

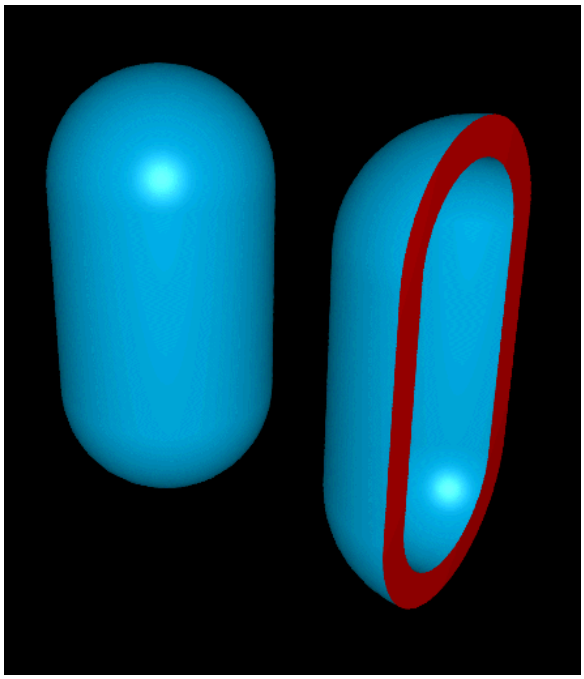

Figure 2: Il solido campione che abbiamo scelto per illustrare la tecnica è relativamente semplice. Nella sez. 7.8 viene descritto con lo schema Constructive Solid Geometry. Esso è stato ottenuto applicando un offset di 2 mm ad un solido che è l'unione di due sfere di raggio pari a 10 mm, e di un cilindro, di raggio uguale a quello delle sfere ed altezza pari a 20 mm, centrato nell'origine.

altre parole, dato un punto arbitrario dello spazio, si deve poter dire se esso è interno o esterno al solido.

Le tecniche di modellazione solida conosciute ed utilizzate sono molteplici [8], [17]. Nel nostro contesto quelle più interessanti sono: Quelle evidenziate nel presente lavoro sono: *Constuctive Solid Geometry* (CSG), la *Boundary-Representation* (B-Rep) e la *Ray-Representation* (Ray-Rep). Ciascuna di esse viene brevemente presentata nella sezione successiva.

2.1 Rappresentazione CSG

Attraverso questo schema è possibile definire solidi complessi mediante composizione di solidi più semplici [11], [9]. Per creare il modello procedurale di un solido complesso vengono usati gli operatori booleani della teoria degli insiemi: unione, intersezione, differenza. Il modello geometrico è rappresentato da un albero binario dove i nodi foglia sono i solidi ottenuti per definizione diretta (le primitive), e i nodi interni sono gli operatori booleani. I solidi definiti come primitive e quelli risultanti da operazioni booleane devono avere una regione

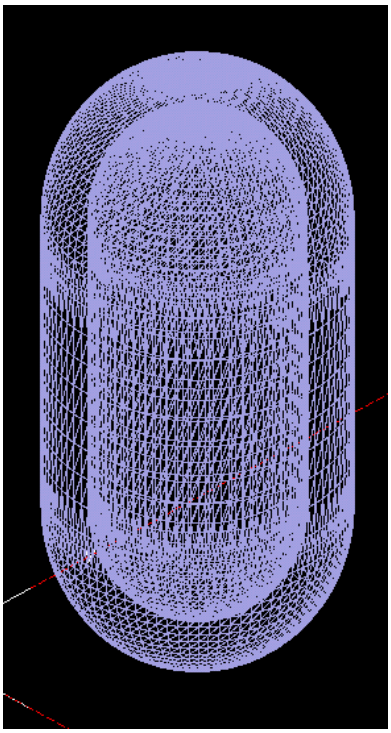

Figure 3: Solido campione nella posizione di stampa dopo che la sua frontiera è stata tassellata (numero di facce triangolari: 21995.)

interna limitata e definita in maniera univoca, ovvero le superfici che definiscono il solido devono separare lo spazio tridimensionale in due regioni: una interna e l'altra esterna al solido stesso. Un solido che soddisfi queste condizioni è detto *valido*. La condizione che le operazioni booleane tra solidi validi diano luogo ancora a solidi validi è detta chiusura rispetto alle operazioni [9].
Un albero CSG consiste dunque nella definizione di un certo numero di primitive in uno spazio a tre dimensioni, e delle operazioni booleane da applicare ad esse. Ciò che ne risulta è una struttura dati ad albero binario i cui nodi interni sono degli operatori booleani e i cui nodi foglia sono delle primitive. La struttura dati che ne deriva è piuttosto concisa anche per solidi molto complessi. Per contro, la frontiera del solido è definita in maniera implicita e questo determina un elevato tempo di calcolo per la sua determinazione [9], [11]. Le primitive implementabili sono numerosissime [6]. Per gli scopi di questo lavoro abbiamo utilizzato due primitive:

- sfera centrata nell'origine con raggio unitario ($x^2 + y^2 + z^2 \leq 1$);
- cilindro ($x^2 + y^2 \leq 1$)avente raggio ed altezza unitari e limitato dai piani di equazione $z = \frac{1}{2}$ e $z = -\frac{1}{2}$;

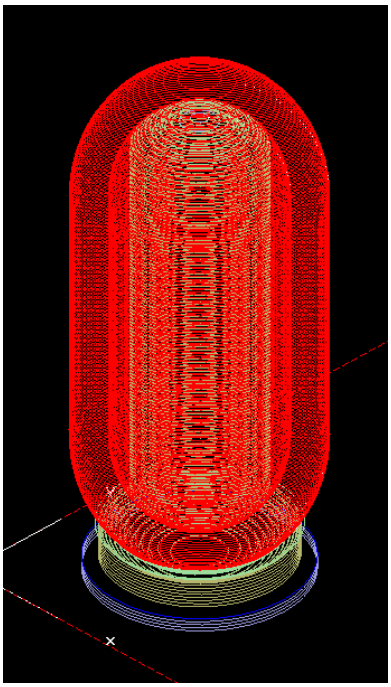

Figure 4: sezioni orizzontali dell'STL solido campione con supporti

2.2 Rappresentazione Boundary (B-Rep)

I modelli B-Rep rappresentano i solidi in maniera indiretta, attraverso una definizione delle superfici che delimitano il solido [12], [17]. Un solido viene rappresentato come un volume racchiuso da un insieme di facce, unitamente a delle informazioni topologiche che definiscono le relazioni tra le facce. La frontiera di un solido separa i punti interni da quelli esterni. I modelli B-Rep possono rappresentare una vasta classe di solidi, ma le strutture dati coinvolte sono pesanti e la rappresentazione di oggetti geometricamente complessi può richiedere grandi quantità di memoria.

Nei modelli B-Rep maggiormente usati, una faccia è una regione limitata di una superficie piana, o di una quadrica, o di un toro. La regione della superficie che forma una faccia è rappresentata da una curva chiusa che giace sulla superficie stessa. Una faccia può contenere diversi contorni chiusi, che rappresentano buchi nel solido. Le curve che delimitano le facce costituiscono gli spigoli. La porzione di curva che forma uno spigolo è rappresentata da due vertici.

Vi sono vari modi per descrivere le facce [17]. Ad esempio, una faccia piana può essere rappresentata in molti modi diversi: attraverso un'equazione analitica o parametrica, oppure con un vettore normale e un punto sulla superficie, ecc. In più, è necessaria una curva chiusa (in generale sghemba) per definire la faccia stessa. Alcuni sistemi CAD recenti usano anche rappresentazioni più complesse, che permettono la costruzione di solidi cosiddetti freeform.

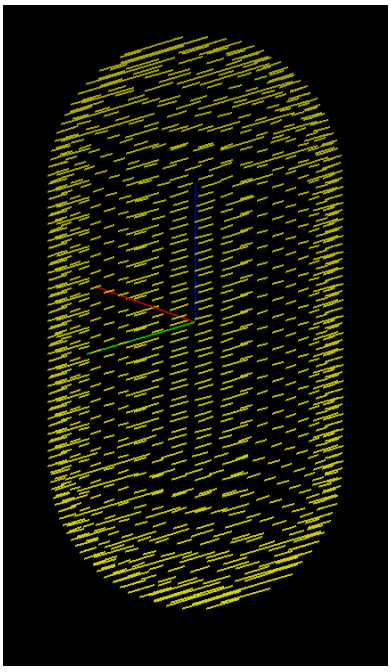

Figure 5: Ray-Rep del solido campione. Campionando il solido campione con una griglia di raggi paralleli all'asse delle y e distanti 0.5 mm, si ottiene la rappresentazione Ray-Rep illustrata. Il campionamento del solido campione CSG con l'algoritmo Raycast [11] produce l'insieme degli in-segmenti.

Tra queste la più comune è la B-Spline, una categoria di superfici che utilizza polinomi a coefficienti parametrici [1].

2.3 Rappresentazione per Raggi (Ray-Rep)

Sia G una griglia di raggi, ossia un insieme di rette parallele tra loro. Una Ray-Representation, detta brevemente Ray-Rep, di un solido A è una rappresentazione degli in-segmenti, ossia dei segmenti di raggio contenuti all'interno del solido, che derivano dalla classificazione di ogni raggio della griglia rispetto al solido di partenza [7]. A differenza delle due precedenti rappresentazioni, una Ray-Rep non può rappresentare un solido ma è una sua approssimazione. D'altra parte una Ray-Rep può essere resa completa introducendo elementi aggiuntivi, detti *htag* (più precisamente si parla di *enhanced* ray-rep), e ponendo opportune condizioni sulla griglia di raggi [7].
A partire da un solido A consideriamo le sue m-facce $\{f_1, f_2, \ldots, f_n\}$, dove una *m-faccia* f_i è il più grande sottoinsieme bidimensionale di un semispazio limitato che appartiene alla frontiera del solido. Senza perdere di generalità possiamo rappresentare un hatg con un'equazione polinomiale del tipo $F \leq 0$ ottenuta prendendo come F il polinomio di grado maggiore il

cui insieme degli zeri contiene la m-faccia f_i. Un semispazio identificato con un htag è detto *semispazio naturale*. Prima di descrivere quali sono le condizioni, che deve soddisfare la griglia, per ottenere una ray-rep completa, introduciamo il concetto di decomposizione naturale disgiuntiva.
Sia $H_n = \{h_1, h_2, \ldots, h_n\}$ l'insieme dei semispazi naturali associati ad un solido A e siano $\{ch_1, ch_2, \ldots, ch_n\}$ i loro complementari. Una *cella naturale* è un insieme di punti definito come intersezione dei semispazi o dei loro complementari, ossia:

$$C_j = g_1 g_2 \ldots g_n, \text{ tale che } g_k \in \{h_k, ch_k\}.$$

Se i semispazi naturali sono n, allora le celle naturali sono 2^n, ed esse costituiscono una partizione dello spazio tridimensionale, che è chiamata *decomposizione naturale disgiuntiva* [7], [10]. Osserviamo che le celle naturali possono non essere connesse ma si suddividono in componenti connesse, dette *componenti naturali*[13]. Supponiamo ora che tutte le componenti naturali di una cella sono siano classificate allo stesso modo rispetto al solido A. Sia $M(X, R)$ una funzione che classifica l'insieme X, rispetto all'insieme R, diciamo che una componente naturale C_{jk} è una in-componente di A se $M(C_{jk}, A) = \text{in}$. Analogamente una componente naturale C_{jk} è una on-componente di A se $M(C_{jk}, A) = \text{in}$, e la frontiera di C_{jk} appartiene alla frontiera di A. Una *o-faccia* è una m-faccia di una on-componente.
PROPOSIZIONE: Sia H_n l'insieme dei semispazi naturali ottenuto scegliendo per ogni m-faccia di un solido A il polinomio di grado maggiore il cui insieme degli zeri contiene quella m-faccia. Allora H_n, la decomposizione naturale $D_n = \{C_j\}$ e l'insieme delle o-facce, è unico per un fissato solido A.

Consideriamo ora una Ray-Rep $RR(G, A)$, con htag $\tau_1, \tau_2, \ldots, \tau_n$ che identificano l'insieme dei semispazi naturali $H_n = \{h_1, h_2, \ldots, h_n\}$. Esistono allora due diverse condizioni sulla griglia affinché la Ray-Rep sia completa.
I: Una Ray-Rep $RR(G, A)$ di un solido A è completa se la griglia di raggi G soddisfa le seguenti condizioni:

1. ogni m-faccia del solido A è colpita da almeno un raggio;
2. ogni naturale in-componente è attraversata da almeno un raggio.

O: Una ray-rep $RR(G, A)$ di un solido A è completa se la griglia di raggi G soddisfa le seguenti condizioni:

1. ogni o-faccia del solido A è colpita da almeno un raggio;
2. ogni naturale on-componente è attraversata da almeno un raggio.

Riepilogando: una ray-rep di un solido A con htag $\tau_1, \tau_2, \ldots, \tau_n$ è completa se la griglia di raggi G soddisfa le condizioni I oppure le O.

3 Ray-Rep e Fused Deposition Modelling

Si è detto che il tipico contesto applicativo della RP è quello della prototipazione industriale. In esso, la validità dei solidi è sotto la responsabilità del programma CAD utilizzato nella progettazione. In contesti applicativi meno convenzionali, quali, ad esempio nelle applicazioni biomediche [15], [16] e nella modellazione di superfici complesse (per es. non orientabili) validare il solido da stampare può diventare un'operazione molto difficile. In ambito medico un caso tipico è la replica di parti anatomiche a partire dal dataset proveniente da una TAC [4].

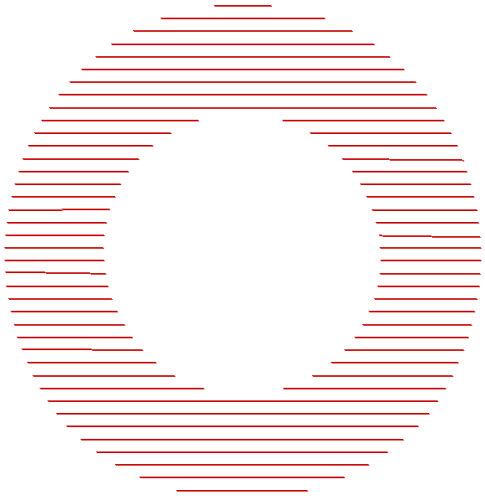

Figure 6: Ray-Rep della sezione assiale del solido campione ad altezza 14 mm dalla base costituita dal piano di lavoro.

Qui l'informazione di cui si dispone consiste, dopo opportune operazioni di segmentazione delle immagini che rappresentano il paziente, nell'insieme delle frontiere delle sezioni piane parallele della parte anatomica (es. un osso) che si vuole riprodurre. Da questa rappresentazione, ricavare la topologia globale della superficie che delimita la parte è un'operazione molto complessa e, nel caso della ricostruzione ai fini della stampa 3D, questa maniera di procedere può determinare un forte decadimento della precisione dei risultati.
Questo lavoro propone la possibilità di utilizzare la Ray-Rep per risolvere il problema della produzione layer-by-layer [6].

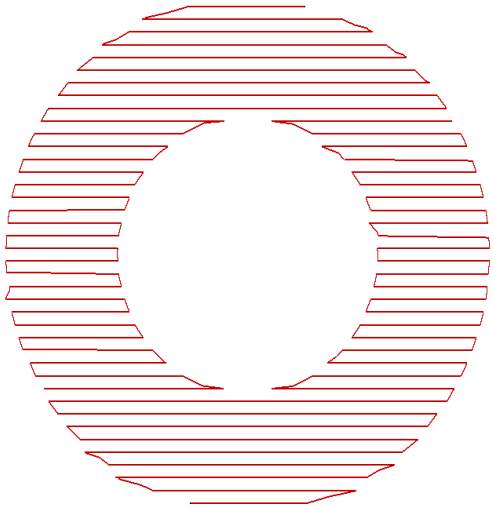

Figure 7: Percorso di riempimento interno del solido campione. A partire dalla Ray-Rep, occorre chiudere i segmenti *in* percorsi (roads). La copertura dell'intera sezione necessita di informazioni di carattere topologico non descritte nella Ray-Rep. Per coprire la sezione in esame sono necessari due percorsi continui, che hanno per estremi il vertice in alto a destra (percorso A) e quello in basso a sinistra (percorso B).

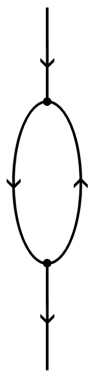

Figure 8: Grafo di Reeb della sezione $z = 14$ mm. I grafi di Reeb sono un valido strumento per verificare la correttezza dei cambiamenti di topologia delle sezioni.

L'idea è di utilizzare esclusivamente la Ray-Rep bidimensionale su ciascuna sezione come struttura dati per gli algoritmi di riempimento del solido con tecnica RP. In altre parole, i segmenti di raggio che definiscono i vuoti e i pieni relativamente al solido possono essere pensati come parte dei percorsi della testa dell'utensile RP. Questo procedimento permette di evitare la ricostruzione 3D della frontiera nel caso di prototipazione di parti anatomiche e

generare direttamente i percorsi di produzione. È comunque possibile utilizzare la Ray-Rep per estrarre la frontiera B-Rep [7]. Possiamo quindi utilizzare la Ray-Rep per generare direttamente il file STL. Questo permette di utilizzare la Ray-Rep nelle usuali pipeline operative descritte in Figura 1.

Partendo dalla Ray-Rep completa (Figura 6) del solido campione è possibile unire i raggi successivi ottenendo linee continue che la testa di stampa 3D segue per depositare il materiale che costituirà il modello di stampa (Figura 7).

La copertura di tutti i percorsi è ottenibile affiancando, per ogni sezione, la Ray-Rep con il relativo grafo di Reeb [2]. Un grafo di Reeb si ottiene quozientando una sezione di un solido n-dimensionale, con $n \geq 2$, rispetto alla relazione d'equivalenza che identifica due punti se e solo se questi stanno sulla stessa componente connessa della sezione considerata (le sezioni sono perpendicolari ad una data direzione di analisi). Nel nostro caso si tratta di sezioni unidimensionali delle sezioni bidimensionali del solido. In altre parole, le componenti continue che si estendono in direzione perpendicolare a quella di analisi vengono contratte in punti. La Figura 8 mostra il grafo di Reeb relativo alla sezione in esame. Solo gli in-segmenti sono rappresentati. Il campionamento è stato eseguito con raggi paralleli all'asse x. L'offset utilizzato (dz) è uguale a quello del prototipatore FDM per la produzione del medesimo oggetto (0.254 mm).

Infine (Figura 9), i percorsi sono completati mediante le linee chiuse di contorno della sezione e delle sue eventuali cavità. Nella Figura 9, a differenza della 7, In questa immagine, a differenza di quella rappesentata nella Figura 7 compaiono delle linee interne ed esterne aggiuntive, che rappresentano il contorno reale della sezione del solido da stampare. Questo genera un solido che è più grande della metà del diametro del filo di estrusione. Questo problema viene risolto con l'utilizzo degli operatori di Minkowski [5], [6], che permettono di ottenere degli offset di solidi combinando i solidi stessi con altri solidi secondo particolari operazioni di somma e differenza. Intuitivamente, ad esempio, la somma di Minkowsky (o M-somma) nel caso 2D, ottenuta dall'aggiunta di una regione piana B ad una regione piana A è quella che si ottiene trasportando B, parallelamente a se stessa, lungo la curva che definisce la frontiera di A e considerando come regione risultante l'unione di A e di tutti i punti del piano coperti da B lungo la sua traiettoria [6].

Per il momento abbiamo scalato il solido campione dell'opportuna quantità.
I problemi ancora aperti riguardano le modalità di utilizzo della Ray-Rep per generare i percorsi utensile ottimizzati e il controllo della topologia della sezione di lavoro tramite i grafi di Reeb. Volendo dimostrare l'uso automatico della Ray-Rep come schema geometrico di lavoro per la prototipazione rapida, è necessario dapprima disporre di algoritmi che realizzino la conversione da questa rappresentazione ad una sicuramente valida (es. CSG o B-Rep) e successivamente effettuare confronti tra le Ray-Rep ottenute per testare la validità dei solidi

definiti. A tal fine abbiamo realizzato gli algoritmi di conversione da CSG a Ray-Rep e da Ray-Rep a CSG, descritti nel paragrafo 4.

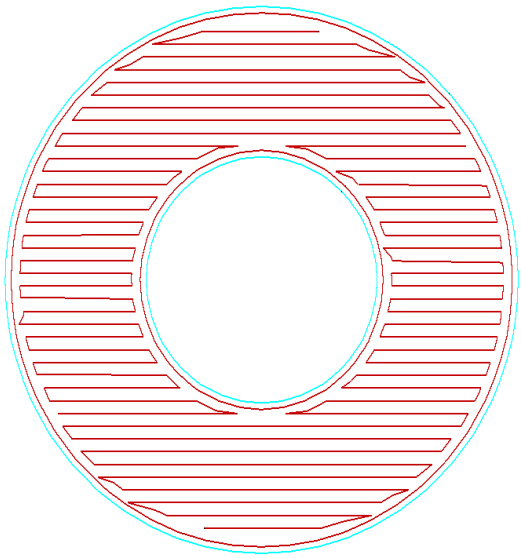

Figure 9: Percorso di riempimento interno del solido campione, con spessore.

4 Conversione CSG - Ray-Rep

Ad una versione semplificata di un modellatore Raycast CSG, sviluppato precedentemente nell'area GEMS [18], sono stati affiancati dei moduli aggiuntivi, ognuno dei quali realizza delle funzioni ben precise.
Per la conversione da CSG a Ray-Rep, l'algoritmo esegue:

- la creazione della lista di htag, a partire dal solido CSG;
- la creazione della ray-rep come lista concatenata di in-segmenti appartenenti ad una griglia di raggi, che soddisfa le condizioni I.

Mentre per quanto riguarda l'algoritmo di conversione da ray-rep a CSG, esse esegue:

- la creazione delle in-componenti;
- la creazione del solido come unione delle in-componenti, detta DNF del solido;
- la conversione della DNF in solido CSG.

4.1 Funzioni implementate

Gli algoritmi di conversione sono stati sviluppati in linguaggio C. Le funzioni cardine degli algoritmi di conversione sono quelle atte a:

- creare la lista degli htag:
  la funzione "listaHtag" prende come argomento un albero CSG, ne cerca le primitive e, per ogni primitiva, trova gli htag che essa genera;
- creare la ray-rep del solido:
  la funzione "CSGRAYREP" genera una matrice di raggi paralleli che campiona il solido. Per ogni raggio, se questo interseca il solido, viene memorizzato un elemento della ray-rep, in particolare vengono memorizzati:
  - il parametro relativo al punto d'intersezione raggio solido;
  - l'htag al quale il punto d'intersezione appartiene;
- creare le in-componenti:
  la funzione "CreaInCell" riceve in ingresso un punto dello spazio, e per ogni htag della Ray-Rep stabilisce se quel punto appartiene al semispazio rappresentato dall'htag oppure al suo complementare;
- creazione del solido come unione delle in-componenti, detta DNF del dolido:
  la funzione "CreaDNF" riceve in ingresso la ray-rep di un solido, prende punti random appartenenti a in-segmenti, chiama la funzione "CreaInCell" che crea la in-componente alla quale appartiene quel punto e la memorizza nella lista concatenata che rappresenta la DNF del solido;
- conversione della DNF in solido CSG:
  la funzione "DNFCSG" riceve come argomento la DNF e ne restuituisce l'albero CSG corrispondente;
- combianzioni di Ray-Rep:
  la funzione "CombinaRR" prende in ingresso due Ray-Rep ed esegue tra esse le operazioni booleane di: unione, intersezione e differenza.

4.2 Test effettuati su solido campione

Per verificare la correttezza degli algoritmi di conversione è stato effettuato un test su casi campione. La Figura 5 mostra la Ray-Rep ottenuta applicando l'algoritmo di conversione da CSG a Ray-Rep su solido campione, costruito in CSG unendo due sfere di raggio unitario a un cilindro con raggio unitario ed altezza due.

5 Conclusioni e sviluppi futuri

Il problema della costruzione layer-by-layer trova nella Ray-Rep un adeguato schema di rappresentazione. I primi risultati mostrano che è possibile costruire i percorsi di riempimento di oggetti semplici basandoci escusivamente sulla Ray-Rep. Nell'ottica della proposta di una rappresentazione di supporto per la stampa 3D, è stata studiata una evoluzione della Ray-Rep che fa uso di informazioni aggiuntive che la rendono completa, e quindi utilizzabile per la stampa 3D automatica. Sono stati implementati degli algoritmi di conversione tra le rappresentazioni CSG e Ray-Rep e sono stati effettuati dei test su semplici solidi campione che hanno confermato la correttezza del metodo seguito. Problemi ancora aperti riguardano la gestione della Ray-Rep negli aspetti seguenti:

- costruzione diretta di geometrie con topologia complessa;
- utilizzo degli operatori di Minkowski per la generazione dell'offset di lavorazione e utilizzo integrato dei grafi di Reeb delle funzioni di Morse per la creazione dei percorsi di riempimento interni e della frontiera.

6 Bibliografia

[1] G. Farin *Curves and Surfaces for Computer Aided Geometric Design*, Academic Press, 1993

[2] A.T. Fomenko, T.L. Kunii *Topological Modeling for Visualization*, Springer, 1997

[3] Gregorio Franzoni, Roberto De Leo, Frabrizio Murgia, Piero Pili, Gabriella Pusceddu, Alan Scheinine, Massimiliano Tuveri, *Realizzazione di un prototipo di carotide con tecnica Fused Deposition Modelling*, Tech. Report CRS4 N. 02/07, 2002

[4] Gabor T. Herman *Geometry of Digital Spaces*, Birkhauser Boston, 1998

[5] J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, London, 1982

[6] Jai Menon, Richard J. Marisa and Jovan Zagajac, *More Powerful Solid Modeling through Ray Representations*, IEEE Computer Graphics and Applications 1994

[7] Jai P. Menon, Hebert B. Voelcker, *On the completeness and conversion of ray representation of arbritrary solids*, Symposium on Solid Modeling and Applications, 175-286, 1995

[8] Aristides A. G. Requicha, *Representations for Rigid Solids: Theory, Methods, and Systems*, Computing Surveys, Vol.12, N.4, (04/1980)

[9] Aristides A. G. Requicha, H. B. Voelcker *Constructive Solid Geometry*, Tech. Memo. N. 25, Production Automation Project, Univ. Rochester, 1977

[10] Jaroslaw R.Rossignac, Hebert B.Voelcker, *Active Zones in CSG for Accelerating Boundary Evaluation, Redundancy Elimination, Interference Detection, and Shading Algoritms*, ACM Transactions on Graphics, Vol.8, N. 1, 51-87, (01/1989)

[11] Scott D. Roth, *Ray Casting for Modeling Solids*, Computer Graphics and Image Processing 18, 109-144 (1982)

[12] H. B. Voelcker, Aristides A. G. Requicha, *Boundary evaluation procedures for objects defined via constructive solid geometry*, Tech. Memo. N. 26, Production Automation Project, Univ. Rochester, 1977/78

[13] Vadim Shapiro, Donald L. Vossler, *Construction and optimization of CSG representations*, Computer-Aided Design, Vol. 23, N. 1, 4-19, Jan/Feb 1991

[14] T. Wholers *Wohlers Report 2001, Rapid Prototyping and Tooling State of the Industry Annual Worldwide Progress Report*, Wholers associates, 2003

[15] CAS 2001 *Atti Convegno Computer Assisted Surgery and Rapid Prototiping in Medicine*, Nurnberg, 10/2001

[16] CARS 2003 *Atti Computer Assisted and Radiology and Surgery*, Internetional Congress Series 1256, 2003

[17] M. Manttyla *Solid Modeling*, Computer Science Press, 1995

[18] G. Franzoni *Modellatore GSG*, Technical Report 03/22. CRS4, Center for Advanced Studies, Research and Development in Sardinia. Cagliari, Italy, apr 2003

7 Appendice: Implementazione della Ray-Rep e Conversione verso CSG

7.1 Include file rrep.h

```
#ifndef CSG_H
#define CSG_H
#define flt double
#define IN 1
#define OUT 0

typedef struct tipo_materiali {
  int  id;                      /* identificatore del materiale */
  char nome[20];
  float Red, Blue, Green, Alfa;
  struct tipo_materiali *next;
} MATERIALE;
```

7.2 Struttura Dati HTags

Le primitive elencate nello schema CSG (2.1) si possono esprimere come intersezione di semispazi illimitati qui descritti (HTAG). Gli HTAG sono memorizzati in una struttura di più alto livello SEMISP che rappresenta l'HTAG o il suo complementare al fine di "ottenere" le componenti naturali descritte dalla struttura dati

```
N_CELL
```

. La DNF, unione di tutte le componenti naturali è rappresentata tramite lista concatenata.

```
typedef struct tipo_trasformazione {
  flt tras[4][4];              /* matrice 4x4 che definisce
                                  la trasformazione di scala
                                  della primitiva istanziata */
  flt inv[4][4];               /* matrice inversa della tras */
} TRASF;
```

```
typedef struct tipo_primitiva {
  MATERIALE *materiale;
  TRASF *trasf;
} PRIMITIVA;

/* lista delle primitive */
typedef struct l_primitiva {
  PRIMITIVA *prim;
  struct l_primitiva *next;
} L_PRIMITIVE;

/* Identificazione dei semispazi naturali nello spazio canonico:
tipo 's': il semispazio ha eq.    X^2 + Y^2 + Z^2 - 1.0 <= 0;
tipo 'c': il semispazio ha eq.    X^2 + Y^2  - 1.0 <= 0;
tipo 'k': il semispazio ha eq.    X^2 + Y^2 - Z^2 + Z - 1/4 <= 0;
tipo 'x'; il semispazio ha eq.    X <= 1/2;
tipo 'y'; il semispazio ha eq.    Y <= 1/2;
tipo 'z'; il semispazio ha eq.    Z <= 1/2;
tipo 'p'; il semispazio ha eq.    X <= 1/2;
tipo 'q'; il semispazio ha eq.    Y <= 1/2;
tipo 'r'; il semispazio ha eq.    Z <= 1/2;
tipo 'w'; il semispazio ha eq. generica AX + BY + CZ + D <= 0; */

typedef struct t_htag{     /* struttura che implementa gli htag */
  char tipo;               /* identifica il tipo di semispazio
                              associato all'htag */
  L_PRIMITIVE *prim_geom;  /* punta alla primitiva dalla quale
                              deriva l'htag */
  int rayflag;             /* e' =1 se il semispazio naturale
                              e'intersecato da almeno un raggio,
                              altrimenti e'=0 */
  int boundflag;           /* e' pari ad 1 se il semispazio e'
                              di frontiera, ossia un parametro
                              d'intersezione tra un raggio e
                              il solido appartiene a tale
                              semispazio, altrimenti e' =0 */
} HTAG;
```

```
typedef struct t_lhtag {    /* lista degli htag */
 HTAG *htag;
 struct t_lhtag *next;
} L_HTAG;

typedef struct semisp{
  HTAG *htag;            /* puntatore ad un semispazio naturale*/
  int id_sp;             /* se id_sp == 0, allora mi riferisco
                            al semispazio  associato all' htag,
                            altrimenti al suo complementare */
}SEMISP;

/* Una componente naturale e' implementata come lista
   di elementi tipo semisp */

typedef struct n_cell{
  SEMISP *semisp;
  struct n_cell *next;
}N_CELL;

/* La struttura tipo dnf (unione delle componenti naturali) e'
   implementata come lista concatenata di n_cell */

typedef struct dnf{
  N_CELL *in_cell;
  struct dnf *next;
}DNF;
```

7.3 Struttura Dati CSG

Lo schema CSG (2.1) è descritto da un albero binario in cui i nodi interni rappresentano le operazioni booleane di Unione, Intersezione e Differenza, mentre i nodi foglia rappresentano le primitive rappresentate da intersezione di semispazi illimitati. Per ogni primitiva è memorizzata l'entità HTAG corrispondente. Per ogni nodo dell'albero è memorizzata la scatola di contenimento (Bounding Box) orientata rispetto al sistema di riferimento.

```
typedef struct tipo_boundbox {
  flt min[4];           /* minimo vertice della bounding box
                           nello spazio di riferimento mondo */
  flt max[4];           /* massimo vertice della bounding
                           box nello spazio di riferimento
                           mondo */
} BOUNDBOX;

typedef struct tipo_csg {  /* implementa i solidi tipo CSG */
  char nome;               /* identifica il tipo di solido:
                              'c', indica il cilindro;
                              's', indica la sfera;
                              'k', indica il cono;
                              '+', indica l'unione;
                              '*', indica l'intersezione;
                              '-', indica la differenza; */
  char tipo;               /* distingue tra solidi tipo
                              primitiva: 'p', e solidi tipo
                              operatore: 'o' */
  struct tipo_csg *sx;     /* puntatore al nodo sinistro */
  struct tipo_csg *dx;     /* puntatore al nodo destro */
  L_PRIMITIVE *primitiva;  /* punta alla primitiva dell'htag
                              curvilineo associato */
  L_PRIMITIVE *p1;         /* punta alla primitiva dell'htag
                              tipo p1 associato */
  L_PRIMITIVE *p2;         /* punta alla primitiva dell'htag
                              tipo p1 associato */
  BOUNDBOX boundbox;       /* questa struttura viene calcolata
                              da int CSG_BoundingBox( ) */
} CSG;
```

7.4 Struttura Dati Campionatore raggi

La struttura dati RAGGIO memorizza i parametri necessari al campionamento Ray-Cast dello schema CSG, per riempire la struttura CLASSIFICAZIONE. Essa è parte della Ray-Rep e viene puntata dalla struttura RRCOLONNA.

```
typedef struct tipo_raggio {
  flt orig[4];              /* [3] = 1.0 */
  flt direz[4];             /* [3] = 0.0 */
  int id;                   /* identificatore del raggio */
} RAGGIO;

#define MAX_INTERSEZIONI 128

typedef struct tipo_classificazione {
  int ni;                                 /* numero di intersezioni tra
                                             raggio e il solido CSG */
  flt parametri[MAX_INTERSEZIONI];        /* parametri[i] = parametro di
                                             intersezione tra il raggio
                                             e il solido ordinati in
                                             ordine crescente */
  char classif[MAX_INTERSEZIONI];         /* quando i >= 1, classif[i] e' la
                                             classificazione dell'intervallo
                                             [parametri[i], parametri[i-1]]
                                             vale 1 se il raggio e' interno
                                             alla primitiva, vale 0 se
                                             l'intervallo associato e'
                                             esterno */
  MATERIALE *materiale[MAX_INTERSEZIONI]; /* il puntatore al materiale
                                             in cui si propaga il
                                             segmento di raggio */
  CSG *primitiva[MAX_INTERSEZIONI];       /* puntatore alla primitiva
                                             intersecata relativa al
                                             parametro i-esimo */
  HTAG *semispazi[MAX_INTERSEZIONI];      /* puntatore al semispazio
                                             contenente l'i-esimo
                                             parametro d'intersezione */
} CLASSIFICAZIONE;
```

7.5 Struttura Dati Ray Rep

Lo schema Ray-Rep è rappresentato dalla struttura dati L_RRRIGA, che memorizza come lista concatenata di righe di campionamento. Per ottenere la Ray-Rep di un solido campione

mandiamo una griglia di raggi che soddisfa, ad esempio, le condizioni I viste nella sezione **??**. Tale griglia è costruita come una matrice di raggi, tutti con la stessa direzione, e ottenuta partendo da un'origine e spostandolo in direzione x (di riga) di una quantità costante pari a sx e in direzione y (di colonna) di una quantità costante pari a sy.

```
typedef struct rr_colonna{       /* struttura associata ad un
                                    raggio della griglia */
  flt origine[4];                /* e' il vettore che identifica
                                    l'origine del raggio */
  CLASSIFICAZIONE *class;        /* deriva dalla classificazione
                                    raggio-solido */
} RRCOLONNA;

typedef struct l_rr_colonna{     /* mettiamo i raggi appartenenti
                                    ad una stessa riga in una lista */
  RRCOLONNA *col;                /* rappresenta il raggio di una
                                    colonna */
  struct l_rr_colonna *next;     /* punta al raggio nella colonna
                                    successiva */
} L_RRCOLONNA;

typedef struct rr_riga{          /* struttura che rappresenta una
                                    riga di raggi della griglia */
   L_RRCOLONNA *head;            /* lista di raggi appartenenti
                                    ad una stessa riga*/
} RRRIGA;

typedef struct l_rr_riga{
   RRRIGA *riga;                  /* riga di raggi */
   struct l_rr_riga *next;        /* puntatore alla riga successiva */
} L_RRRIGA;
```

7.6 Lista degli HTags di un solido CSG

```
.....
static L_HTAG *testa_lhtag = NULL;      /* puntatore al primo elemento
```

```
                                        della lista degli htag */

L_HTAG *GetTestaLHTag( )
{  return( testa_lhtag );
} /* GetTestaLHTag() */

L_HTAG *InsertHTag(HTAG *htag, char *nomef) /* inserisce un htag nella lista */
{
 ....

} /* InsertHTag() */

/* La funzione che segue prende come argomento albero CSG,
   trova le sue primitive, e per ogniuna di queste mette
   gli htag corrispondenti in una lista concatenata */


static void listaHtag(CSG *solido)
{
  HTAG *htag;
  if(solido->tipo == 'o')
    {
      listaHtag(solido->dx);
      listaHtag(solido->sx);
    }
  else
    { /* il primo htag di una primitiva che si inserisce nella lista
         e' quello di tipo curvilineo */
      htag = NewHtag("lista:curve");
      ....
      htag->tipo = solido->nome;
      ....
      if( solido->nome != 's' )    /* se la primitiva e' del tipo:
                                      'c' o 'k', inseriamo nella
                                      lista anche gli altri due
                                      semispazi */
    {  /* semispazi di tipo 'z'*/
```

```
            htag = NewHtag("lista:p1");
        htag->tipo = 'z';
        ....
            ....
        solido->p1 = solido->primitiva;
        ....
            ....
            /* semispazio di tipo 'r'*/
            htag = NewHtag("lista:p2");
            htag->tipo = 'r';
            ....
            ....
        solido->p2 = solido->primitiva;
    }
    }
} /* listaHtag() */

L_HTAG *CreaListaHTag( CSG *solido )
{
  listaHtag( solido );
  return( GetTestaLHTag() );
}
```

7.7 Ray Rep di un solido CSG

La direzione del raggio di campionamento è costante, mentre l'origine del raggio viene iterativamente modificata per coprire la griglia di campionamento. Il convertitore utilizza un algoritmo RayCast [11] per riempire la struttura CLASSIFICAZIONE descritta in A.1.4.

```
void CSG_RAYREP(CSG *solido,
            flt point_of_view[], flt dir_of_view[],
        int nriga, int ncolonna,
         flt sx, flt sy)
{
  int id = 0;
  int i,j = 0;
  RAGGIO ray;
```

```
CLASSIFICAZIONE *clas = NULL;
RRCOLONNA *col;
RRRIGA *rriga;
ray.direz[0] = dir_of_view[0];
ray.direz[1] = dir_of_view[1];
ray.direz[2] = dir_of_view[2];
ray.direz[3] = dir_of_view[3];
VecNormalizza(ray.direz);
ray.orig[3] = 1.0;
for( i = 0; i < nriga ; i++ )
  {
    testa_lcol = NULL;
    for( j = 0; j < ncolonna; j++ )
  { id++;
        ray.id = id;
    ray.orig[0] = point_of_view[0] + (flt) j*sx;
    ray.orig[1] = point_of_view[1] ;
    ray.orig[2] = point_of_view[2] + (flt) i*sy;
    if( clas == NULL )
      clas = New_Class("CSG_RAYREP");
    CSG_ClassificaRaggio(&ray, solido, clas, ID_RAGGIO);
        col = NewColonna("CSG_RAYREP");
    col->origine[0] = ray.orig[0];
    col->origine[1] = ray.orig[1];
    col->origine[2] = ray.orig[2];
    col->origine[3] = ray.orig[3];

        if(TestIntersezione(clas, ID_RAGGIO))
      {
        col->class = clas;
        clas = NULL;
      }
          else   col->class = NULL;
     InsertColonna(col, "CSG_RAYREP");
   }
      rriga = NewRiga("CSG_RAYREP");
      rriga->head = testa_lcol;
```

```
        InsertRiga(rriga, "CSG_RAYREP");
    }
}

L_RRRIGA *GetTestaRayrep(void)
{
  return(testa_lriga);
} /* GetTestaRayrep() */

L_RRRIGA *CreaRayrep(CSG *solido,
                     flt point_of_view[4], flt dir_of_view[4],
             int riga, int colonna,
             flt sx,flt sy)
{
  CSG_RAYREP(solido,
             point_of_view, dir_of_view,
             riga, colonna,
             sx, sy);
  return( GetTestaRayrep() );
} /* CreaRayrep */
```

7.8 Esempio di conversione CSG - Ray Rep

Il solido campione, descritto nel paragrafo 1 (Figura 2 è rappresentato con lo schema CSG mediante la differenza ($CSG_Differenza$) di due gusci composti da due sfere ed un cilindro. Il solido CSG viene convertito nella Ray-Rep corrispondente utilizzando una struttura Raggio (A.1.4) che memorizza tutte le informazioni necessarie al campionamento.

```
main()
{   L_HTAG *head = NULL;          //aggiunta
 L_RRRIGA *primo = NULL;

 RAGGIO point_view;

 CLASSIFICAZIONE class;
    CSG *solido0, *solido1, *solido2, *solido3, *solido4, *solido5;
          *solido6, *solido7, *solido8, *solido9, *solidospess;
```

```
 /* distanze sono in mm */

 flt raggio_ext = 10.0;
 flt altezza_cyl = 20.0;
 flt off_set  = 3.0;
 flt raggio_int = raggio_ext - off_set;


 flt dist = 10.0;


 flt lsx = 0.5;
 flt lsz = 0.254;

/* NUMERO DI RAGGI CAMPIONE DA VALUTARE */

 int riga = (altezza_cyl+2.*raggio_ext)/lsz+1;
 int colonna = 2.*raggio_ext/lsx+1;

/* PUNTO IN BASSO A DESTRA */

 point_view.orig[0] = -raggio_ext-dist;
 point_view.orig[1] = -raggio_ext - lsx;
 point_view.orig[2] = -0.5*altezza_cyl-raggio_ext;
 point_view.orig[3] = 1.0;

/* DIREZIONE DI SCANSIONE */

 point_view.direz[0] = 1.0;
 point_view.direz[1] = 0.0;
 point_view.direz[2] = 0.0;
 point_view.direz[3] = 0.0;

 point_view.id = 0;

/* INSTANZIA IL SOLIDO CAMPIONE IN CSG */
```

```
 solido0 = CSG_Cilindro(raggio_ext, altezza_cyl, "rosso");
 solido4 = CSG_Sfera(raggio_ext, "rosso");
 CSG_Trasla(solido4, 0.0, 0.0, 0.5*altezza_cyl);
 solido2 = CSG_Unione( solido0, solido4);
 solido3 = CSG_Sfera(raggio_ext, "rosso");
 CSG_Trasla(solido3, 0.0, 0.0, -0.5*altezza_cyl);
 solido1 = CSG_Unione(solido2, solido3);

 solido5 = CSG_Cilindro(raggio_int, altezza_cyl, "rosso");
 solido6 = CSG_Sfera(raggio_int, "rosso");
 CSG_Trasla(solido6, 0.0, 0.0, 0.5*altezza_cyl);
 solido7 = CSG_Unione(solido5, solido6);
 solido8 = CSG_Sfera(raggio_int, "rosso");
 CSG_Trasla(solido8, 0.0, 0.0, -0.5*altezza_cyl);
 solido9 = CSG_Unione(solido7, solido8);

 solidospess = solido0;
 solidospess = CSG_Differenza(solido1, solido9);

 /* CONVERSIONE DA CSG VERSO RAYREP */
  head = CreaListaHTag(solidospess);
 printf("\nlista htag 1\n");
 stampa_htag(head);

 primo = CreaRayrep(solidospess,
            point_view.orig, point_view.direz,
            riga, colonna,
            lsx , lsz);


 salva_rayrep("rr_capsula", primo, point_view.direz);
 return;

}
```