Semplificazione di Modelli Geometrici in Memoria Secondaria

David ***Canino***
canino.david@gmail.com
Università degli Studi di Genova
30 Ottobre 2007

Relatori
prof. Paola ***Magillo***
dott. Davide ***Sobrero***

Correlatore
prof. Patrizia ***Boccacci***

Introduzione

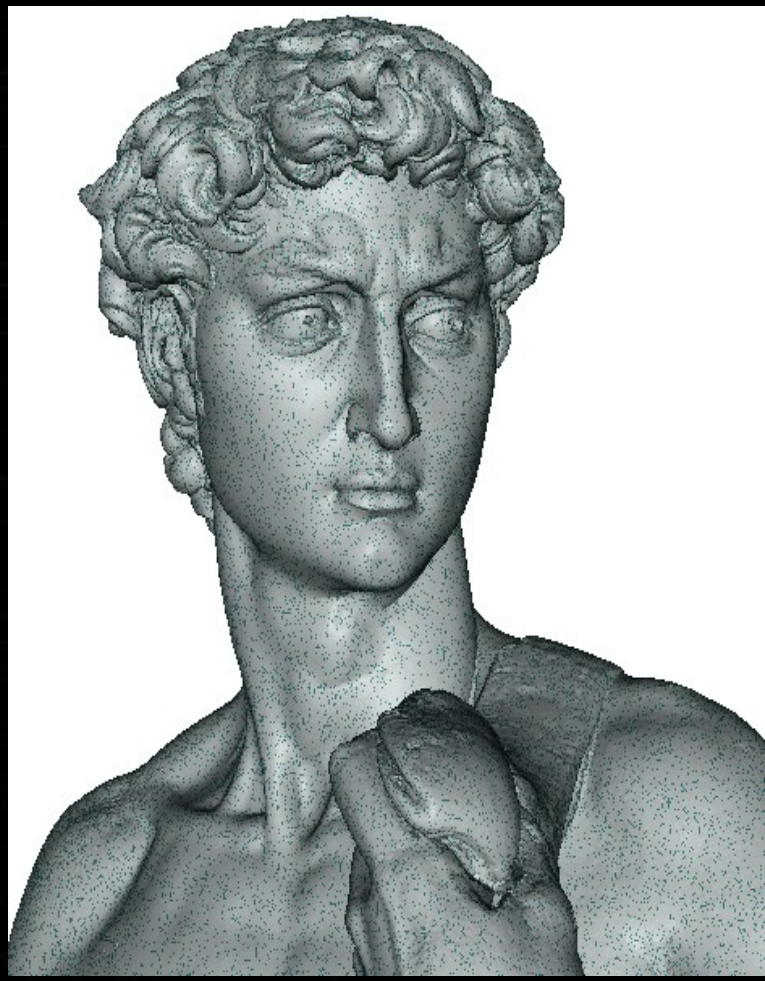
- Nelle applicazioni si richiede di ***rappresentare*** oggetti del mondo reale in maniera tale da poter essere gestiti in un ***calcolatore*** attraverso un ***modello geometrico***
- La rappresentazione sicuramente più nota in letteratura sfrutta i ***complessi simpliciali***
- Possiamo limitare la nostra analisi al solo caso ***euclideo***
- Un ***k-simplesso*** euclideo è la combinazione convessa di k+1 punti linearmente indipendenti
- Un ***complesso simpliciale euclideo*** è un insieme finito H di simplessi euclidei che soddisfa le seguenti proprietà:
  - se γ è un simplesso di H e β una faccia di γ allora anche β è un simplesso di H
  - se α e β sono due simplessi di H, la loro intersezione o è vuota o è una faccia di entrambi i simplessi

Introduzione (2)

- In letteratura troviamo due importanti esempi di complessi simpliciali:
  - le ***triangolazioni*** in cui ogni simplesso è un triangolo: sono utilizzate per la rappresentazione di ***superfici***
  - le ***griglie di tetraedri*** in cui ogni simplesso è un tetraedro: sono utilizzate per la rappresentazione di ***volumi***
- Nella nostra ricerca è molto importante il concetto di ***risoluzione*** o ***livello di dettaglio*** di un complesso simpliciale euclideo, definito come la densità dei suoi simplessi

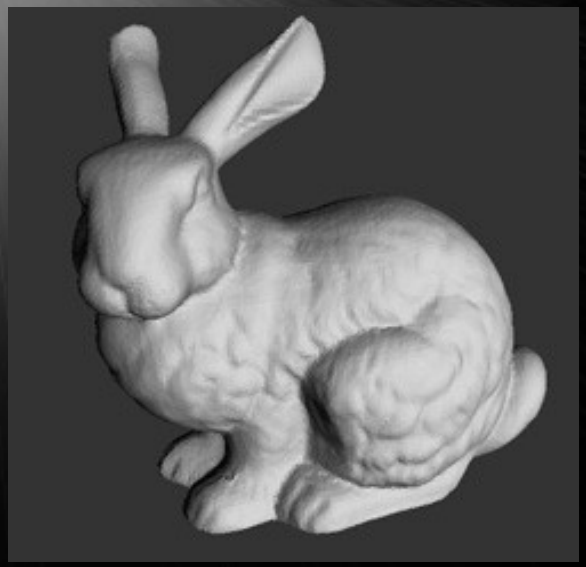
Problema

- Il ***livello di dettaglio*** di una mesh può essere elevato: ciò implica un'alta ***occupazione spaziale*** del complesso simpliciale
- La dimensione della mesh può ***eccedere*** la quantità di ***memoria primaria*** disponibile in un calcolatore
- Pertanto ***NON*** si possono gestire più i vari modelli geometrici in un calcolatore.
- Possiamo ricordare i modelli digitali delle opere di Michelangelo, generati nell'ambito del progetto ***Digital Michelangelo***, presso la ***Stanford University*** all'indirizzo Web ***http://graphics.stanford.edu/projects/mich/***



La semplificazione

- Una soluzione di questo problema è dato dall'uso delle tecniche di ***semplificazione***, le quali ***riducono*** il livello di dettaglio e quindi l'***occupazione*** spaziale del complesso simpliciale
- L'idea è di partire da un complesso simpliciale ad ***alto*** livello di dettaglio ed ottenerne una versione ***semplificata*** applicando degli operatori locali di ***modifica***



Il dilemma

- Solitamente le tecniche di semplificazione richiedono:
  - il ***caricamento*** dell'intera mesh in ***memoria primaria***
  - l'***applicazione*** di una certa ***modifica*** locale alla mesh in modo da ridurne il livello di dettaglio
- Le ***modifiche*** vengono ***applicate*** alla mesh finchè non viene raggiunto il livello di dettaglio voluto
- Ma la dimensione di un complesso simpliciale può ***eccedere*** la quantità di memoria primaria, pertanto queste tecniche ***NON*** sono applicabili
- Pertanto è necessario definire delle tecniche di ***semplificazione*** in ***memoria secondaria*** mantenendo la mappa poligonale in un supporto di memorizzazione: queste tecniche sono note anche come ***semplificazione out-of-core***

Semplificazione out-of-core

- Questi metodi si possono suddividere in varie tipologie:
  - ***clustering di tipo spaziale:*** si basa sulla suddivisione in ***cluster*** dei punti della mesh. Ogni cluster viene poi sostituito da un ***punto rappresentativo***, scelto secondo un certo criterio. Ad esempio:
    - P. Lindstrom, 2000
    - P. Lindstrom and T. Silva, 2001
  - ***streaming mesh:*** si basa su una ***codifica*** della mesh, adatta all'invio su uno ***stream***. Ad esempio:
    - M. Isenburg e altri, 2003
    - M. Isenburg e P. Lindstrom, 2005
  - ***decomposizione*** della mesh attraverso un ***indice spaziale***, la quale verrà approfondita in questa ricerca. Ad esempio:
    - H. Prince, 2000
    - P. Cignoni e altri, 2003

La decomposizione della mesh

- Si basa sulla suddivisione della mesh attraverso un ***indice spaziale***
  - Un ***indice spaziale*** è una struttura dati gerarchica ad ***albero***, la quale ***suddivide*** ricorsivamente un certo dominio, fino ad arrivare a delle celle atomiche (dette ***foglie*** dell'albero), alle quali associamo i dati da memorizzare.
- Gli ***indici spaziali*** vengono mantenuti su ***disco*** ed i dati memorizzati vengono ordinati in modo tale che quelli ***spazialmente vicini*** siano nello ***stesso blocco***: in questo modo si limita il numero di accessi al disco
- Questa proprietà verrà garantita raggruppando i nodi in ***cluster***, ognuno dei quali costituisce l'***unità*** di memorizzazione ***fondamentale*** che andrà scritta sul ***disco***
- Nel nostro caso vogliamo gestire un ***complesso simpliciale*** quindi nelle ***foglie*** ne verranno memorizzati i ***simplessi***

La semplificazione di una mesh

- Il punto di ***partenza*** della nostra analisi è l'algoritmo di ***semplificazione*** introdotto in ***Cignoni e altri, 2003***
- Per poter ***semplificare*** una mesh secondo questo ***approccio*** dobbiamo applicare questo ***schema*** di funzionamento:

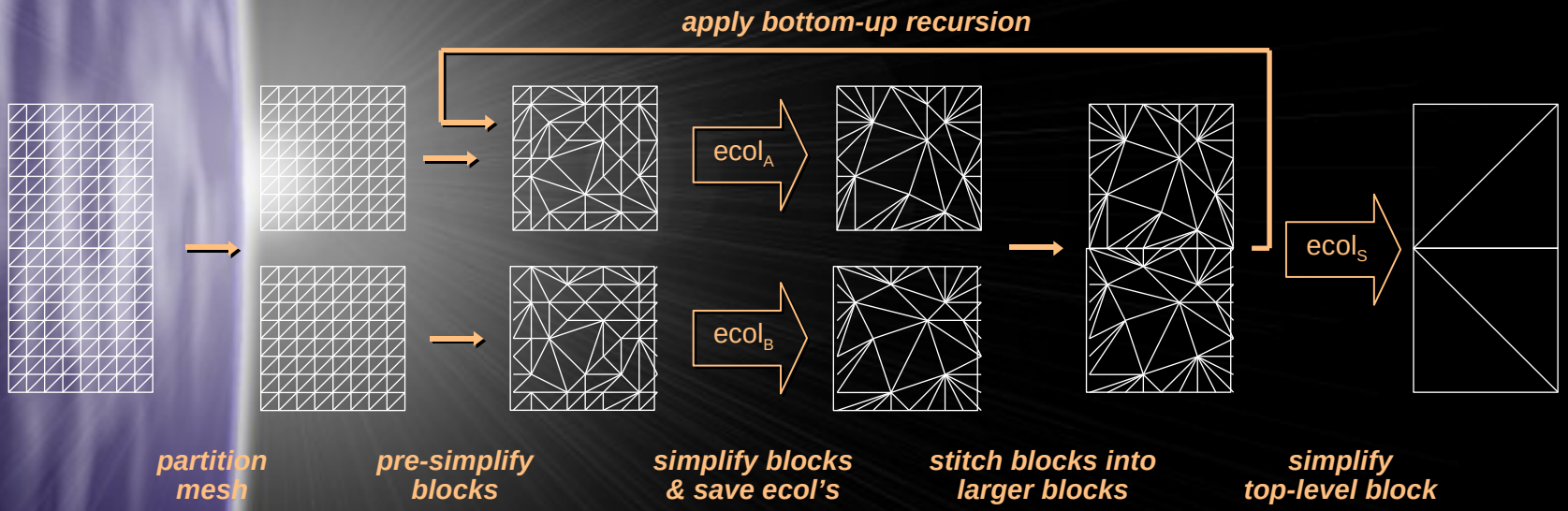


figure courtesy of Hugues Hoppe

dove assumiamo che:

- la mesh venga ***suddivisa*** attraverso un generico indice spaziale
- ogni ***foglia*** contenga una ***porzione*** di mesh semplificabile in ***RAM***

Il contributo di questa ricerca

- Il nostro obiettivo è quello di poter variare il tipo di ***indice spaziale***, il tipo di ***complesso simpliciale*** e l'algoritmo di ***semplificazione*** nello schema appena introdotto
- In realtà possiamo osservare che in letteratura sono stati ***già*** sviluppati:
  - svariati ***indici*** spaziali – *H. Samet, 2006*
  - vari algoritmi di ***semplificazione*** per complessi simpliciali
- ***NON*** è disponibile un framework per l'***indicizzazione*** spaziale di mesh in grado di ***adattarsi*** facilmente alle varie ***esigenze*** dell'utente.
- In questa ricerca si propone una possibile ***soluzione***, introducendo il framework ***OMSM*** (dall'espressione inglese *Objects Management in Secondary Memory*) per la gestione di un complesso simpliciale in memoria secondaria.

La memorizzazione di dati spaziali

- In letteratura sono state sviluppate varie architetture per la gestione di ***dati spaziali*** in memoria secondaria, considerando questi aspetti:
  - l'uso di ***indici*** spaziali per facilitare le operazioni sui dati
  - la ***suddivisione*** dei nodi dell'indice spaziale in ***cluster*** in base ad una certa politica: per ***cluster*** si intende un gruppo di nodi, i quali possono essere considerati un'unica entità
  - la ***gestione*** dinamica dei ***cluster*** in memoria secondaria
- Questi aspetti possono essere considerati ***indipendenti*** fra loro e le tecniche utilizzate per la loro gestione possono essere combinate in maniera ***ortogonale***

La memorizzazione di dati spaziali (2)

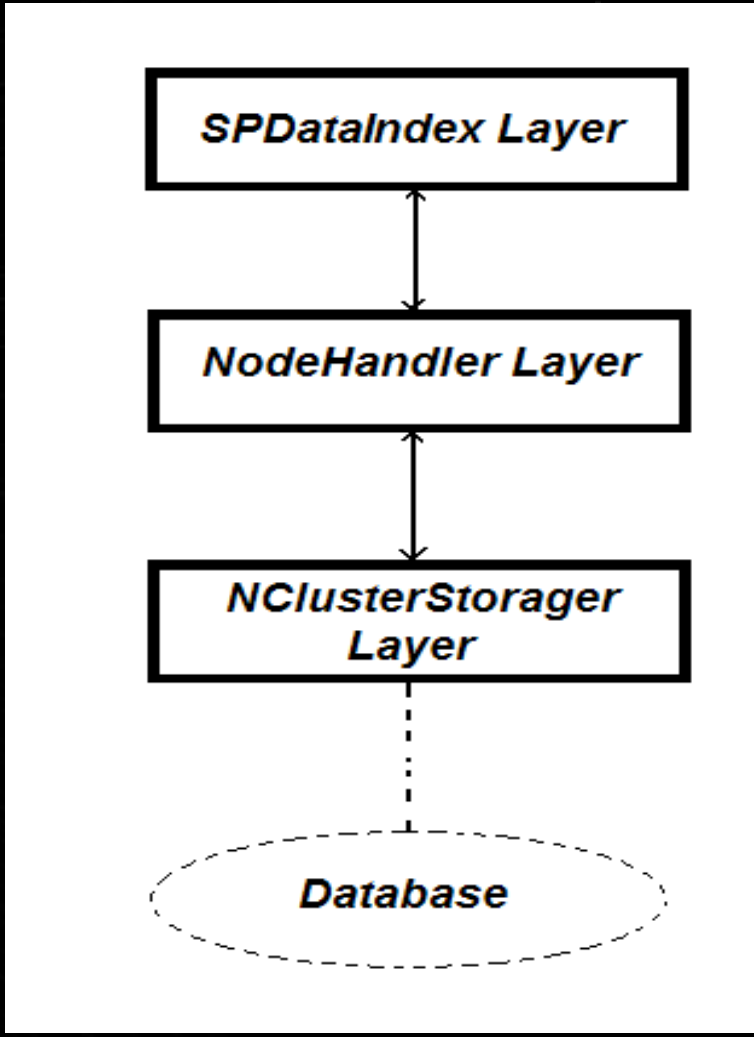
- La maggior parte delle architetture di memorizzazione
  - prevedono una serie di ***scelte fissate*** a priori, soprattutto per quanto riguarda gli ultimi due aspetti.
  - permettono solamente di ***cambiare*** l'indice spaziale da usare
  - gestiscono solamente ***dati bidimensionali***
- Ad esempio possiamo ricordare il database spaziale ***Oracle Spatial***, quello ***IBM DB2 Spatial Extender*** e quello ***GiST (Hellerstein, 1995)***
- Pertanto i framework esistenti per la memorizzazione di dati spaziali:
  - ***NON*** si adattano facilmente rispetto alle esigenze dell'utente ed hanno una ***struttura monolitica,*** difficilmente modificabile
  - ***NON*** possono essere utilizzati per la ***decomposizione*** di complessi simpliciali e quindi per la loro semplificazione

Il framework *OMSM*

- E' stato introdotto in questa tesi per gestire in maniera ***dinamica*** ***grosse*** quantità di dati geometrici in memoria secondaria
- Può ***integrare*** fra loro le diverse tecniche sviluppate in letteratura, garantendo la massima ***flessibilità*** possibile nella risoluzione di questo problema
- Ha una struttura ***modulare*** e facilmente ***estendibile,*** adattandosi alle varie ***esigenze*** di memorizzazione dell'utente
- E' in grado di memorizzare un certo insieme di ***entità geometriche*** di ***dimensione*** topologica ***diversa***, ma immerse nello ***stesso spazio*** metrico euclideo

Il framework *OMSM (2)*

- Ha una struttura ***multi-livello***, la quale può essere facilmente ***estesa***
- ***NON*** viene assunto l'utilizzo di una particolare tecnica per ogni livello, ma solamente che ogni livello sia in grado di offrire dei ***servizi*** in relazione al suo ruolo
- Il funzionamento di un livello può essere considerato ***indipendente*** dagli altri
- L'utente può ***scegliere*** l'implementazione di ogni livello, ottenendo vari ***comportamenti*** del framework
- Ogni ***livello*** ha un suo ***modello*** dei dati, ***indipendente*** dagli altri



Il livello *SPDataIndex*

- Permette all'utente di ***interagire*** con i dati memorizzati, ***nascondendo*** i dettagli implementativi del framework
- Il ***modello*** dei dati di questo livello è un ***generico*** oggetto geometrico
- Ogni oggetto geometrico viene indicizzato attraverso il suo ***punto rappresentativo***, il quale ne descrive le proprietà, secondo un certo principio (ad esempio il baricentro dell'oggetto)
- In questo livello possiamo ***variare*** il tipo di ***indice*** spaziale da usare
- Viene mantenuto in ***memoria*** solamente il nodo ***radice*** di un indice spaziale: gli altri nodi verranno caricati dinamicamente
- L'***efficienza*** delle primitive dipende dal tipo di ***indice*** spaziale

Il livello *NodeHandler*

- Si occupa della gestione dei ***nodi*** dell'indice spaziale e di quello di configurazione (detto ***Super-Nodo***), a seconda delle richieste del livello ***SPDataIndex:*** entrambi verranno indicati come ***nodo OMSM***
- Il ***modello*** dei dati di questo livello è il ***generico*** nodo ***OMSM***
- Suddivide i nodi secondo una qualsiasi politica di ***clustering*** per ***minimizzare*** il numero di operazioni di ***I/O*** da eseguire sul database
- La ***suddivisione*** in cluster può influire sull'***efficienza*** del livello
- Per ***minimizzare*** il numero di ***accessi*** al database dei cluster viene mantenuta una ***cache*** in modo tale da memorizzare i cluster più frequentemente utilizzati, sfruttando la politica di sostituzione ***LRU*** (dall'inglese ***Least Recently Used***)

Il livello *NClusterStorager*

- Si occupa della ***gestione*** a basso livello dei ***cluster*** di nodi ***OMSM***, operando su un supporto di memorizzazione
- Le ***tecniche*** utilizzate in questo livello dipendono dalla ***dislocazione fisica*** dei cluster, ***modificabile*** a seconda delle varie esigenze
- Il ***modello*** dei dati gestiti da questo livello è definito dalla ***coppia*** formata dal ***codice identificativo*** del cluster e dalla ***sequenza di byte*** che lo rappresenta, mantenuta a ***finale piccolo*** (ordine ***little-endian***)
- In questo modo è possibile:
  - ***tralasciare*** i dettagli implementativi del cluster da gestire
  - ***applicare*** delle trasformazioni sulla sequenza di byte, ad esempio possiamo cifrarla o comprimerla, a seconda delle varie esigenze

La libreria *OMSM*

- Contiene l'ampia parte ***implementativa*** della nostra ricerca
- E' stata realizzata in ***C++*** ed è compatibile con lo standard ***POSIX***
- E' supportata dalle piattaforme ***GNU/Linux*** e ***Microsoft Windows***
- Per dimostrare la ***fattibilità*** della soluzione proposta, è stato realizzato un ***prototipo*** del framework per la gestione di ***triangolazioni***:
  - gli indici spaziali disponibili sono il ***K-d tree***, il ***Quadtree*** e le loro versioni ibride cioè le strutture ***Hybrid K-d trie*** e ***Hybrid Quadtrie***
  - la politica di clustering è quella ***singola***, secondo cui un cluster può contenere un ***solo*** nodo
  - i cluster vengono memorizzati nel database di tipo embedded ***Oracle Berkeley DB***, molto noto in letteratura
- Questo prototipo ha uno scopo dimostrativo e potrà essere facilmente ***esteso*** in futuro, soprattutto negli ultimi due livelli

Configurare il framework *OMSM*

- Possiamo modificare alcune proprietà del framework ***OMSM***, le quali riguardano la sua struttura come:
  - il tipo di ***indice spaziale*** da utilizzare
  - la politica di ***clustering*** dei nodi
  - la ***dislocazione*** fisica dei dati
  - un ***parametro ausiliario*** per la memorizzazione dei cluster, il cui significato varia a seconda della dislocazione fisica dei dati
  - il ***numero massimo dei livelli*** in un indice spaziale, usato per variare la capacità di un nodo, ove previsto
- Le impostazioni sono mantenute in un ***nodo speciale*** di un database di tipo ***OMSM***, detto ***Super-Nodo***
- Per ***facilitare*** il processo di configurazione viene fornito il programma ***Omsmconf***, dotato di un'interfaccia grafica realizzata usando il toolkit ***FLTK***, disponibile all'indirizzo Web ***http://www.fltk.org***

Conclusioni

- Il lavoro di ricerca svolto in questa tesi si inquadra nella risoluzione del problema della ***semplificazione*** di un complesso simpliciale euclideo in memoria secondaria
- Il nostro scopo è quello di ***generalizzare*** la tecnica di semplificazione descritta in ***Cignoni e altri, 2003*** rispetto:
  - al tipo di ***indice spaziale*** utilizzato per la decomposizione
  - all'algoritmo di ***semplificazione*** iterativa utilizzato
  - al tipo di ***complesso simpliciale*** in input
- In questa tesi abbiamo definito il framework ***OMSM*** (dall'inglese ***Objects Management in Secondary Memory***) in grado di ***decomporre*** un complesso simpliciale euclideo
- Questa architettura si caratterizza per l'elevato grado di ***modularità*** e di ***flessibilità*** in modo da adattarsi alle varie ***esigenze*** di memorizzazione