Progetto Programmazione calcolo scientifico 2025

Kameleddine Haj Mabrouk, Triscari Gabriele

Matematica per l'ingegneria, POLITO

Contenuti

- Documentazione UML
- Triangolazione classe I
- Triangolazione classe II
- Duale poliedro
- Shortest Path

Documentazione UML

Polyheron model

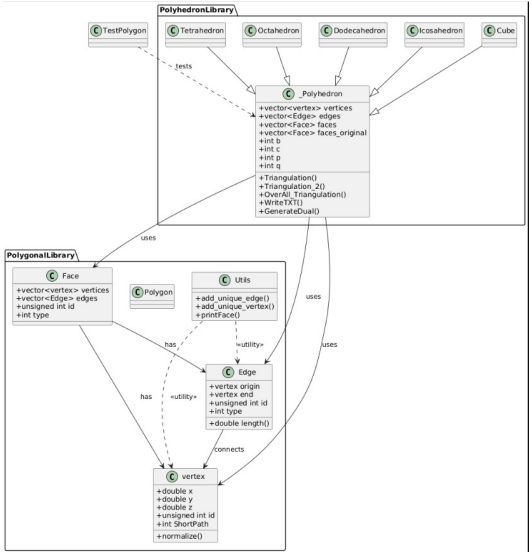
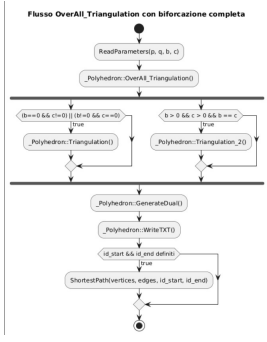


Diagramma di flusso



Triangolazione I

Funzione trianolazione classe I:
La logica alla base della funzione triangolazione classe I è data dalla generazione di tanti layers quanti sono necessari per il riempimento del numero di triangoli necessari per ricoprire ciascuna faccia. Ogni layer ha un numero di triangoli pari $k+1, k \geq 0$ 'upper' ($\triangle$) e $k, k \geq 0$ triangoli lower ($\triangledown$) . Ogni layer($l \geq 0$) ha un numero di vertici i pari $l+1$, equispaziati secondo la formula:

$$v_{jl} = v_{l,start} + (v_{l,end} - v_{l,start}) \frac{j}{l+1}, \quad ; 0 \leq j \leq l+1$$

Out[30]=

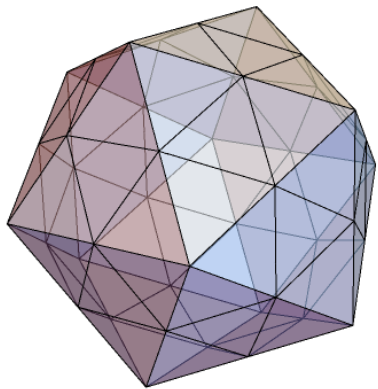


Figure: (p,q,b,c)=(3,5,2,0)

Triangolazione I

Ogni arco e_j invece è data dalla semplice connessione di $v_{l,j} - v_{j+1,l}, \quad 0 \leq j \leq l$
L'analisi della complessità per quanto concerne la procedura complessiva della triangolazione classe I:

- In spazio $\mathcal{O}(F * b^2)$.
- In tempo $\mathcal{O}(F * b^2)$.

Una semplice verifica è che se aumentassi avrei al $b+1$-esimo layer $b+2$ vertici, cioè $b+2$ in più vertici rispetto al trangolazione al layer b-esimo, quindi $\Delta V_{b+1} = b+2 \rightarrow \sum_{0 \leq i \leq b} \Delta V_i = \sum_{i \in [0,b]} (i+2) = \frac{b(b+1)}{2} + 2b = V_b - V_0 \in \mathcal{O}(b^2)$.
Da notare che la tipologia del poliedro(icosaedro, ottaedro, tetraedro) interviene solo in funzione del numero delle facce. In modo simile si ragiona anche per la complessità in spazio, in effetti è necessario creare al $b+1$-layer b archi, quindi la complessità è $O(b^2)$ (a meno del numero di facce, poiché la triangolazione andrebbe effettuata tante quante sono le facce).

Triangolazione Classe II

La funzione `Triangulation_2()` ha lo scopo di effettuare una triangolazione di un poliedro definito da vertici, spigoli e facce già sottoposta a triangolazione di classe 1), producendo una nuova rappresentazione (classe 2),
Vengono copiati i dati dei vertici, spigoli e facce del poliedro di partenza (classe 1) per poterli riutilizzare in modo sicuro.
Si inizializzano nuovi contenitori per la classe 2: vertici (`ver_2`), facce (`faces_2`), spigoli (`edges_2`) e una mappa per evitare duplicati di vertici (`vertex_map_2`).
Si ricopiano i vertici originali nella nuova struttura mantenendo l'univocità tramite la mappa.

Triangolazione Classe II, calcolo dei centri baricentrici

Per ogni faccia triangolare si calcola il baricentro, che viene aggiunto come nuovo vertice nella Classe 2.
Per ogni faccia della Classe 1, si creano tre spigoli che collegano il baricentro ai vertici della faccia. Questi spigoli vengono aggiunti nella Classe 2.
Si costruisce una mappa che associa ogni spigolo originale alle facce a cui appartiene, per distinguere spigoli interni e spigoli di bordo.
Nel caso in cui uno spigolo appartenga a una sola faccia (bordi del poliedro), si calcola il punto medio dello spigolo, lo si aggiunge alla lista dei vertici, e si costruiscono nuovi spigoli e triangoli con il baricentro della faccia.
Se uno spigolo è condiviso da due facce, si costruisce un triangolo che unisce i due baricentri con uno dei vertici dello spigolo. Questo triangolo viene aggiunto alla lista delle nuove facce.
Alla fine del processo vengono aggiornate le stutture di vertices, faces e edges.

Triangolazione Classe II, Complessità computazionale

La funzione `Triangulation_2()` lavora su una mesh composta da n vertici, m spigoli e f facce. Le principali operazioni e i relativi costi sono:

- Inserimento dei vertici in una mappa: $\mathcal{O}(n \log n)$.
- Calcolo dei baricentri e inserimento: $\mathcal{O}(f \log n)$.
- Creazione dei nuovi spigoli (3 per faccia): $\mathcal{O}(f)$.
- Mappatura spigolo-faccia (ricerca lineare su m spigoli): $\mathcal{O}(f \cdot m)$.
- Suddivisione degli spigoli e costruzione facce triangolari: $\mathcal{O}(m \log n)$.

Il costo totale è:

$$\mathcal{O}(n \log n + f \log n + f \cdot m + m \log n)$$

Costruzione del duale

Obiettivo: generare il duale del poliedro..

Nel grafo duale:

- Ogni faccia del poliedro diventa un nodo.
- Due nodi sono collegati da un arco se le rispettive facce condividono un lato (cioè 2 vertici).
- Gli archi del duale vengono rappresentati come segmenti tra i centroidi normalizzati delle facce.

La funzione `GenerateDual()` costruisce il duale a partire dalla mesh triangolata. Ogni faccia viene rappresentata dal proprio centroide, proiettato sulla sfera, e per ogni coppia di facce adiacenti viene creato un arco che le collega. Il risultato è un grafo tridimensionale in cui la struttura viene conservata ma invertita: ciò che erano facce diventano nodi, e le adiacenze tra facce diventano archi.

Duale poliedro

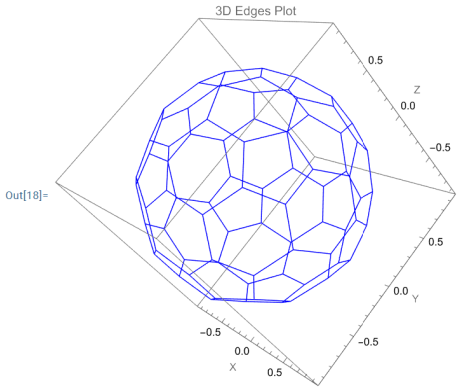


Figure: Duale di (p,q,b,c)=(3,5,2,0)

Calcolo del cammino minimo: `ShortestPath`

Obiettivo: determinare il cammino più breve tra due vertici su un poliedro.

Algoritmo utilizzato: Dijkstra, adatto a grafi con pesi positivi.

Modello di grafo:

- Ogni vertice del poliedro è un nodo del grafo.
- Il grafo è non orientato e costruito dinamicamente a partire dalla lista di archi.

Costo utilizzato:

- Il peso associato a ciascun arco è la distanza euclidea in 3D tra i due vertici collegati.

Shortest Path

Struttura dell'algoritmo:

1. Verifica che gli ID forniti siano validi.
2. Costruisce la lista di adiacenza come `unordered_map<id, lista<(vicino, peso)>`.
3. Inizializza:
   - Vettore `dist[]` con `+inf`
   - Vettore `prev[]` con predecessori
   - Coda di priorità (heap min) per l'estrazione del nodo con distanza minima
4. Applica l'algoritmo di Dijkstra.
5. Ricostruisce il cammino minimo da `id2` a `id1`.
6. Marca i vertici e archi coinvolti con `ShortPath = 1`.
7. Calcola la lunghezza totale del cammino e la stampa.

Complessità computazionale: $\mathcal{O}((V + E) \log V)$ con heap min (V = vertici, E = archi)

Test effettuati

1. **Test sulle triangolazioni:**
   verifica vertici, spigoli, facce e formula di Eulero.

2. **Test sul cammino minimo (`ShortestPath`):**
   - verifica percorso da vertice 0 a id2 impostato.
   - su grafo semplice.
   - controllo correttezza cammino.

3. **Test sulla validità di generazione del duale:**
   - duale del tetraedro.
   - duale dell'ottaedro.

4. **Test su triangolazione di Classe II:**
   - verifica formule teoriche con $b = c$.