Suite di test per la verifica di invarianti topologici nel calcolo dell'arrangement 2D

Dario Di Nardo Matteo Petruzziello

February 2020

Indice

1 Introduzione **2**
1.1 Formula di Eulero 2
1.2 Obbiettivi del progetto 2

2 Progettazione della struttura dei test **3**

3 Test sull'Arrangement 2D **4**
3.1 Test 1: due quadrati tangenti vertice-vertice 4
3.2 Test 2: due quadrati tangenti vertice-lato 5
3.3 Test 3: n poligoni complanari 6

4 Test sull'Arrangement 3D **7**
4.1 Test 4: due cubi concentrici 7
4.2 Test 5: due cubi con un vertice in comune 8
4.3 Test 6: due cubi intersecanti 8

5 Solidi platonici **10**
5.1 Octahedron 10

6 Poliedri non manifold **11**
6.1 Tetramiesaedro/Tetrahemihexahedron 11
6.2 Ottaemiottaedro/Octahemioctahedron 12

7 Conclusioni **12**

1 Introduzione

1.1 Formula di Eulero

In matematica, e più precisamente in topologia algebrica, la **caratteristica di Eulero** (o **numero di Eulero**) è un invariante topologico, un numero che descrive la forma o la struttura di uno spazio topologico indipendentemente da come questo è piegato. È comunemente indicata dalla lettera greca χ.

La caratteristica di Eulero χ è definita per le superfici dei poliedri, secondo la formula

$$\chi = V - E + F \tag{1}$$

dove V, E e F sono rispettivamente il numero dei vertici, spigoli e facce nel poliedro.

Qualsiasi superficie di un poliedro 3D omeomorfo alla sfera ha la caratteristica di Eulero:

$$\chi = V - E + F = 2. \tag{2}$$

Poligoni 2D definiti come grafi planari hanno $\chi = 2$, se si considera la faccia esterna, e $\chi = 1$, se non si considera.

Equazione di Eulero-Poincaré

Una forte generalizzazione dell'equazione di Eulero è l'*equazione di Eulero-Poincaré*, la quale vale per il numero di vertici, spigoli e facce di un solido poliedrale connesso:

$$V - E + F = 2(s - g) + h, \tag{3}$$

dove:

1. s è il numero delle componenti connesse del bordo del corpo solido;
2. g è il genere topologico del solido;
3. h è il numero dei buchi nelle facce del solido.

1.2 Obbiettivi del progetto

Il progetto prevedeva di progettare e implementare una **suite di test** per la verifica di invarianti topologici, come la formula di Eulero, che fosse di supporto alla libreria *LinearAlgebraicRepresentation* (nel seguito della trattazione LAR).

Nella pipeline di decomposizione spaziale per il calcolo della disposizione generata da una raccolta di complessi cellulari, una fase intermedia è la valutazione dell'operatore $\partial 2$ in 3D, dopo la frammentazione indipendente delle 2-celle in 2D e prima del *Topological Gift Wrapping* in 3D. In particolare, la matrice $\partial 2$ viene prodotta unendo una serie di sotto-matrici calcolate indipendentemente l'una dall'altra. La fusione si basa sulla scoperta di classi di equivalenza della relazione di congruenza tra 0-, 1- e 2-celle di oggetti di input decomposti, calcolati in modo indipendente per ogni *2-face* di input. Qui vengono introdotti diversi test locali, al fine di scoprire tutte le coppie congruenti tra le p-celle decomposte in 2D ($0 \leq p \leq 2$), attraverso il calcolo di invarianti topologici locali.

2 Progettazione della struttura dei test

I test sono progettati affinché abbiano tutti una stessa struttura, così da poterli implementare con facilità e poter confrontare i risultati.

- inizializzazione delle strutture dati
- esecuzione dell'algoritmo di Arrangement 2D (o 3D)
- calcolo della caratteristica di Eulero
- visualizzazione dei risultati mediante la libreria *ViewerGL*

Inizializzazione delle strutture dati

Le strutture dati utili ai fini dei test trattati in questo progetto sono inizializzate tramite alcune funzioni per generare poligoni e poliedri semplici (come quadrati, rettangoli, cubi, ecc) e/o composizioni più articolate di questi. Per queste ultime, in alcuni casi, sono utilizzate funzioni per la creazione di complessi cellulari generati randomicamente e alcune delle istanze così ottenute vengono prese in considerazione per essere utilizzate come input ai test, di cui parleremo nelle prossimi paragrafi.

È stato inoltre utilizzato il tipo di dato *Struct* della libreria LAR e la funzione *struct2lar* per l'inizializzazione delle variabili per i *modelli Lar*.

Esecuzione dell'algoritmo di Arrangement

Dopo l'inizializzazione dei dati, viene eseguito l'algoritmo di Arrangement, *planar_ arrangement* o *spatial_ arrangement* a seconda dei casi. Durante l'esecuzione dell'algoritmo sono state inserite nel codice delle stampe di debug che permettono di studiare più attentamente e analizzare il valore della caratteristica di Eulero per ogni 2-faccia σ che viene sottoposta alla suddivisione indipendente nell'Arrangement 2D, all'interno della pipeline. Vengono infatti stampati dei messaggi che visualizzano il numero della 2-cella σ correntemente computata e il rispettivo valore del numero di Eulero calcolato al termine della sua frammentazione 2D. In figura 1 si vede un esempio di queste stampe di debug.

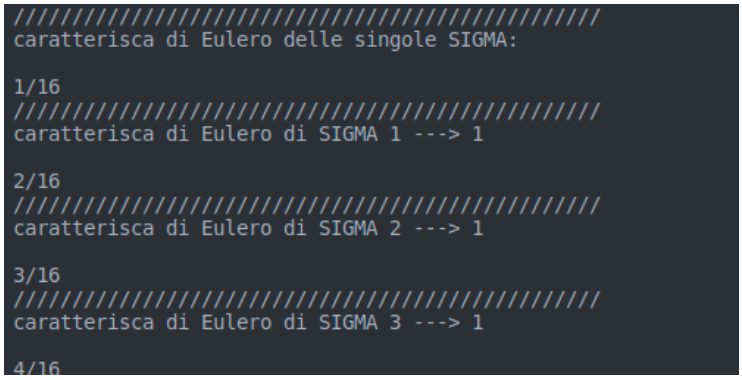


Figura 1: Stampe di debug del calcolo della caratteristica di Eulero durante la frammentazione delle 2-celle.

Verifica della caratteristica di Eulero

Le funzioni per il calcolo della caratteristica di Eulero si trovano nel file *topological_ invariants.jl*.

Per gli oggetti 2- e 3-dimensionali, la caratteristica di Eulero viene calcolata tramite la funzione

```
euler_characteristic(V::Lar.Points, copEV::Lar.ChainOp, copFE::Lar.ChainOp)
```

dove:

- V è la matrice dei vertici, disposti per riga;
- *copEV* e *copFE* sono le matrici di cobordo-0 e cobordo-1, di tipo *ChainOp*.

È stata implementata anche una generalizzazione per il calcolo della caratteristica di Eulero per spazi topologici in n-dimensioni, con $n \geq 1$. Per questi spazi χ viene definita semplicemente come

$$\chi = k_0 - k_1 + k_2 - k_3 + \cdots ,$$

dove k_n è il numero di facce n-dimensionali (vertici e spigoli sono intesi come facce di dimensione 0 e 1).

La funzione implementata è chiamata

```
euler_characteristic_general(V::Lar.Points, cc::Lar.ChainOp...)
```

dove:

- V è la matrice dei vertici, disposti per riga;
- *cc...* è una sequenza di matrici di cobordo di tipo *ChainOp* che definiscono la n-catena.

Visualizzazione dei risultati con *ViewerGL*

Infine, alla fine dei test sono presenti delle chiamate alla libreria *ViewerGL* per visualizzare gli oggetti ottenuti dall'esecuzione delle funzioni precedenti.

3 Test sull'Arrangement 2D

3.1 Test 1: due quadrati tangenti vertice-vertice

Questo test lavora su un complesso cellulare formato da due quadrati tangenti per uno dei loro vertici, come mostrato in figura 2.

Il codice per costruire l'input è mostrato qui di seguito:

```
V, (VV,EV,FV) = Lar.cuboid([1,1], true)
square = (V,EV)
model = Lar.Struct([ square,
                     Lar.r(π), square ])
V,EV = Lar.struct2lar(model)
```

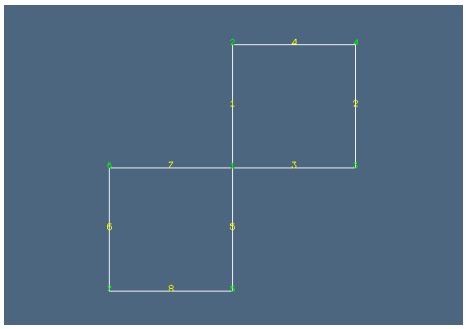


Figura 2: Due quadrati tangenti per uno dei loro vertici.

Al termine dell'arrangiamento 2D la caratteristica di Eulero per l'intero complesso cellulare viene calcolata correttamente, ossia si ottiene $\chi = 1$ (non viene considerata la faccia esterna); viene riscontrato il valore corretto da anche per il numero delle componenti biconnesse. Infatti le componenti biconnesse calcolate alla fine dell'arrangiamento sono due. In figura 3 vengono mostrate, con colori diversi, le componenti biconnesse calcolate dalla funzione *Lar.Arrangement.biconnected_components*.

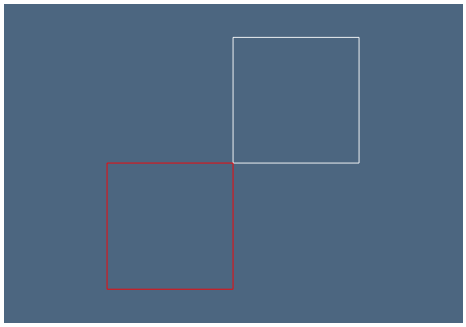


Figura 3: Le componenti biconnesse calcolate.

3.2 Test 2: due quadrati tangenti vertice-lato

Questo test lavora su un complesso cellulare formato da due quadrati, come il precedente, con la differenza che qui il vertice di uno è tangente ad uno spigolo dell'altro, come mostrato in figura 4.

Il codice per costruire l'input è mostrato qui di seguito:

```
V, (VV,EV,FV) = Lar.cuboid([1,1], true)
square = (V,EV)
model = Lar.Struct([ square,
                     Lar.t(0,0.4), Lar.r(3π/4), square ])
V,EV = Lar.struct2lar(model)
```

Al termine dell'arrangiamento 2D, come nel precedente test, la caratteristica di Eulero per l'intero complesso cellulare viene calcolata correttamente,

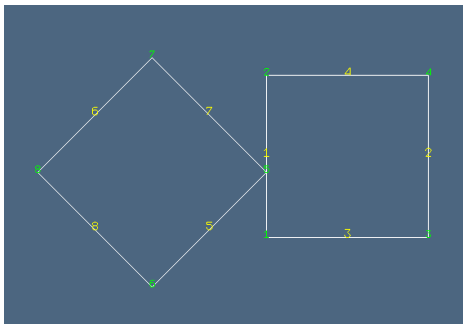


Figura 4: Due quadrati che si tangono vertice-lato.

$\chi = 1$ (non viene considerata la faccia esterna); viene riscontrato il valore corretto anche per il numero delle componenti biconnesse. Infatti le componenti biconnesse calcolate alla fine dell'arrangiamento sono due. In figura 5 vengono mostrate, con colori diversi, le componenti biconnesse calcolate dalla funzione *Lar.Arrangement.biconnected_ components*.

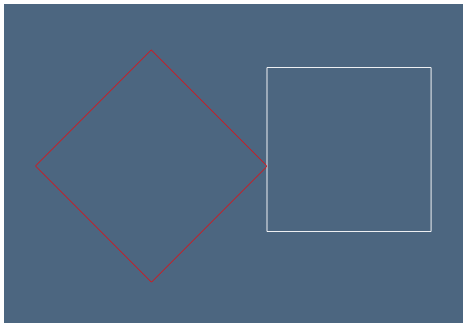


Figura 5: Le componenti biconnesse calcolate.

3.3 Test 3: n poligoni complanari

Questo test esegue l'arrangment su un complesso di 10 poligoni complanari. I dati di input sono stati scelti a partire dalla generazione di 2-complessi simpliciali randomici; dopo molte esecuzioni è stata esportata una istanza significativamente rappresentativa del caso di interesse per questo test. In figura 6 è rappresentato l'input in questione; per i dettagli numerici dei dati si rimanda invece al codice presente nelle repository del progetto.

Dopo l'esecuzione dell'Arrangement si ottengono i seguenti risultati:

```
EULERIAN CHARACTERISTIC-->8, BICON COMPS=8, VERTEX=44, EDGES=48, FACES=12
```

Il risultato è corretto, poiché la caratteristica di Eulero in questo caso è uguale a $1 *$ il numero di componenti biconnesse (concordemente all'equazione di Eulero-Poincaré (3), paragrafo 1.1).

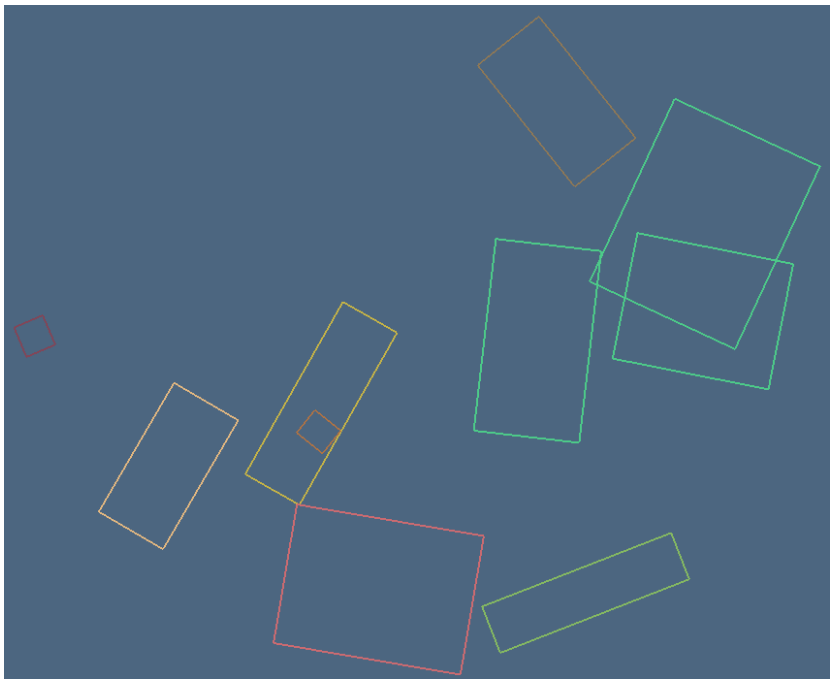


Figura 6: Il 2-complesso simpliciale di poligoni complanari utilizzato in questo test.

4 Test sull'Arrangement 3D

4.1 Test 4: due cubi concentrici

Questo test opera su due cubi concentrici, dove uno è contenuto completamente nell'altro, senza che ci siano intersezioni tra i rispettivi bordi (figura 7).

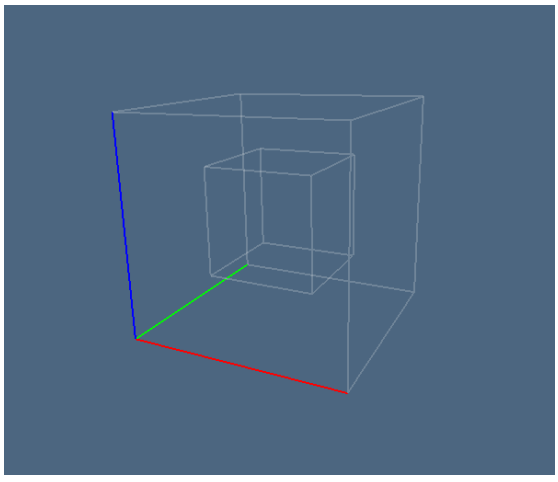


Figura 7: Due cubi concentrici.

Il calcolo della caratteristica di Eulero, dopo l'arrangment 3D, rivela che

il test viene eseguito come se si considerassero i due cubi indipendentemente; nel caso in questione, invece, avremmo voluto che il cubo più piccolo, quello contenuto completamente dentro l'altro, non venisse preso in considerazione in quanto immerso e non essendo visibile nel solido finale.

```
EULERIAN CHARACTERISTIC-->4, BICON COMPS=2, VERTEX=16, EDGES=24, FACES=12
```

Stando alla formula di Eulero-Poincaré (3), il risultato posto in questo modo è corretto in quanto $\chi = 2 * \#biconn = 2 * 2 = 4$; il risultato desiderato invece prevedeva un numero di Eulero pari a 2, poiché i due cubi partecipano allo stessa struttura (dove le componenti 3-dimensionali connesse sono una soltanto).

4.2 Test 5: due cubi con un vertice in comune

Questo test lavora su due cubi inseriti uno dentro l'altro, con entrambi i cubi aventi un vertice nell'origine. L'arrangiamento rivela l'esclusione del vertice in comune presente nell'origine dovuto al bug dell'arrangement 2D.

La non correttezza dell'arrangement è notificata anche dal test scritto per l'HOMEWORK 2, nel quale la matrice di cobordo-1 presenta delle colonne con un numero dispari di elementi uguali ad 1, come mostra la figura 8.

```
////////////////////////////////////////////////////////////////////////////////////////////////////
CHECKING COLUMNS WITH AN ODD NUMBER OF ELEMENTS
WARNING, THE MATRIX CONTAINS 6 COLUMNS WITH AN ODD NUMBER OF ELEMENTS
the execution will sleep for 5 seconds
////////////////////////////////////////////////////////////////////////////////////////////////////
TEST EULERO AFTER SPATIAL ARRANGEMENT
EULERIAN CHARACTERISTIC-->3 BICON COMPS==1 VERTEX = 15 EDGES = 24 FACES = 12
```

Figura 8

4.3 Test 6: due cubi intersecanti

Test A

Questo primo test riproduce una precisa istanza di *randomcubes.jl* dove sono presenti due cubi che si intersecano parzialmente (figura 9).

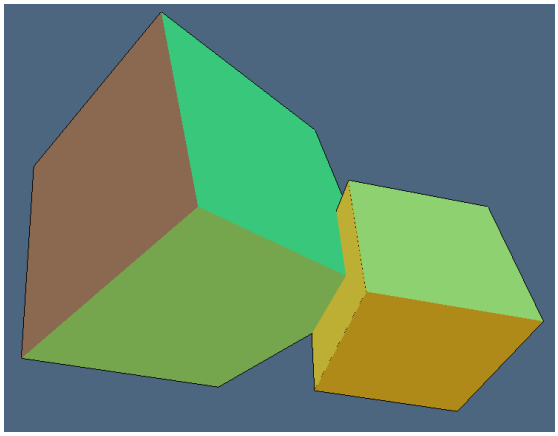


Figura 9: Due cubi che si intersecano.

```
EULERIAN CHARACTERISTIC-->4 BICON COMPS==1 VERTEX=21 EDGES=34 FACES=17
```

In questo caso il test sulla caratteristica di Eulero dovrebbe ritornare il valore di 2, in quanto presente una sola componente biconnessa tridimenzionale. Il motivo del risultato errato sta nel fatto che il risultato dell'arrangiamento considera anche facce, spigoli e vertici interni al poliedro, mentre il test dovrebbe tener conto solo di quelli di bordo. Per tanto il calcolo corretto dovrebbe essere

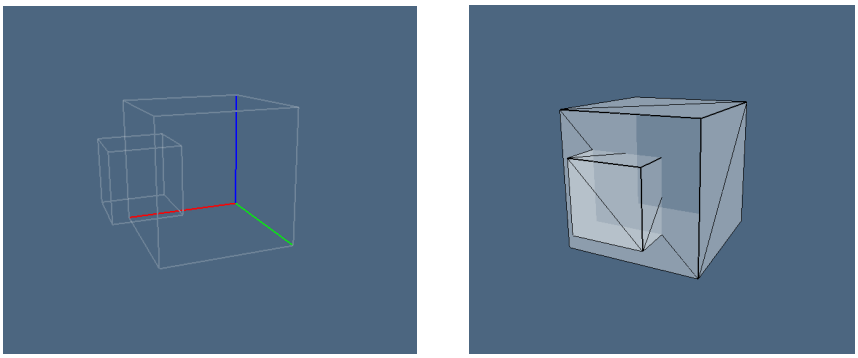
$$V - E + F = 20 - 30 + 12 = 2.$$

Test B

Questo secondo test di cubi intersecanti prevede il complesso simpliciale mostrato in figura 10. L'input è stato costruito ad hoc e il codice relativo alla generazione dei dati è il seguente:

```
V,(VV,EV,FV,CV) = Lar.cuboid([1,1,1],true);
cuboid_big = (V,EV,FV,CV);
V,(VV,EV,FV,CV) = Lar.cuboid([1.25,0.75,0.75],true,[0.75,0.25,0.25]);
cuboid_small = (V,EV,FV,CV);

str = Lar.Struct([ cuboid_big, cuboid_small ]);
V,EV,FV,CV = Lar.struct2lar(str);
```



(a) **(b)**

Figura 10: Altri due cubi che si intersecano.

In questa situazione, con la verifica del numero di Eulero di tutte le 2-face durante l'algoritmo di arrangiamento spaziale, otteniamo un risultato errato relativo alla faccia σ del cubo grande che è attraversata dal cubetto piccolo (identificata come *SIGMA 2* nella stampa di degub in figura 11), mentre $\chi = 1$ per tutte le altre facce. Infatti la caratteristica di Eulero per *SIGMA 2* risulta pari a 2, mentre ci saremmo aspettati che fosse pari a 1, e di conseguenza anche il risultato del calcolo di χ dell'intero complesso simpliciale risulta errato. Questo fatto è dovuto a causa della mancanza all'interno della libreria LAR di una funzione che calcoli le componenti biconnesse sulla matrice delle 2-facce di oggetti 3D. Al momento è presente solamente la funzione che prende in input la matrice EV degli spigoli.

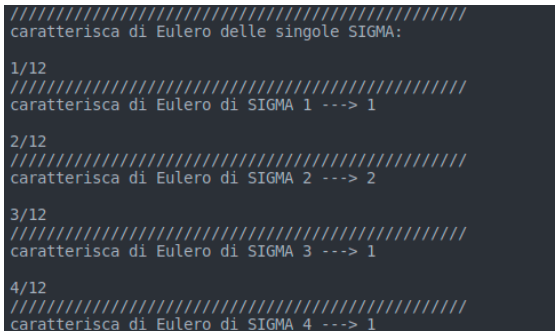


Figura 11: Numero di Eulero errato su una delle 2-facce.

5 Solidi platonici

Nello spazio 3D un solido platonico è un poliedro convesso regolare. Questo è costruito da facce poligonali congruenti e regolari con lo stesso numero di facce che si incontrano a ogni vertice. I solidi platonici sono:

- Tetraedro
- Cubo
- Ottaedro
- Dodecaedro
- Icosaedro

Dato che il tetraedro e il cubo sono già implementati in LAR abbiamo provato a implementare l'Ottaedro, descritto nella sezione seguente.

5.1 Octahedron

L'ottaedro è un poliedro con otto facce triangolari. Ha sei vertici e dodici spigoli, per tanto, come per gli altri solidi platonici la caratteristica di Eulero è uguale a 2.

Abbiamo implementato una funzione che data in ingresso la dimensione del lato della base restituisce la struttura Lar.Struct dell'ottaedro.

```
function octahedron(l)
```

A questo punto abbiamo implementato anche due varianti:

- function octahedron(l,h) dove l è la dimensione del lato della base e h è l'altezza, genera la versione non regolare dell'ottaedro, ovvero una piramide a base quadrata che ha altezza arbitraria, restituita sempre come Lar.Struct.
- function rectangularbase_octahedron(ab,da,h) che restituisce la Lar.Struct di un ottaedro a base rettangolare e altezza arbitraria, con ab e da che sono le dimensioni dei lati del rettangolo e h è l'altezza.

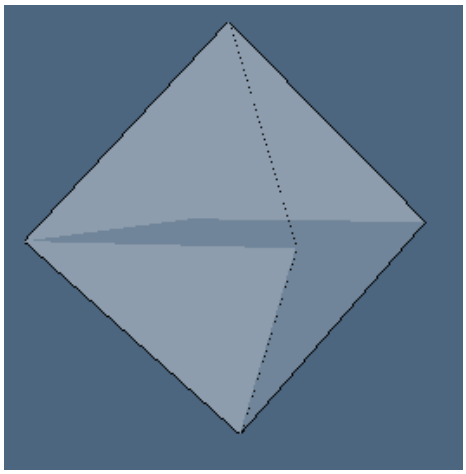


Figura 12: Octahedron

6 Poliedri non manifold

I solidi non manifold sono una rappresentazione di solidi "non reali" sotto forma di componenti reali. Questo tipo di solidi rende maggiormente funzionale e flessibile la modellazione dei solidi. I solidi non manifold possono talvolta essere il risultato di un passaggio di modellazione non corretto o di un risultato inatteso della conversione delle superfici in un solido.

6.1 Tetramiesaedro/Tetrahemihexahedron

Il tetramiesaedro è un solido non manifold avente 6 vertici, 12 spigoli e 7 facce, con caratteristica di eulero uguale a 1.

Siamo riusciti a implementarlo tramite composizione di tetraedri. Il risultato è mostrato nelle figure che seguono.

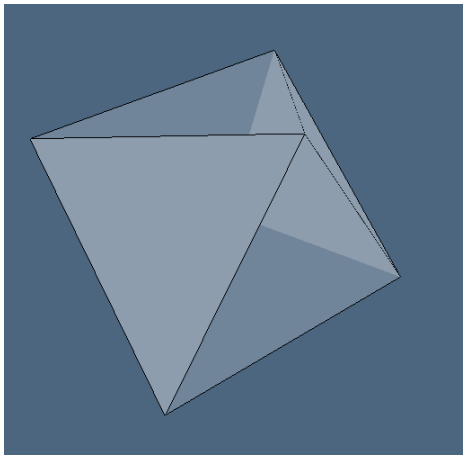


Figura 13: Tetrahemihexahedron

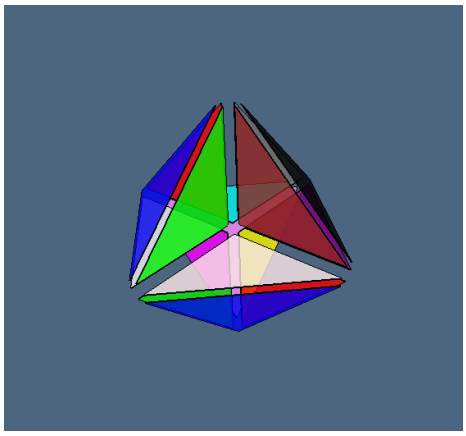


Figura 14: Tetrahemihexahedron esploso

Abbiamo provato ad eseguire *spatial_ arrangement* una volta eliminati i vertici duplicati e abbiamo visto che questo cicla a causa del bug già noto sulla ricostruzione delle 3-celle all'interno della funzione *minimal_ 3cycles*.

6.2 Ottaemiottaedro/Octahemioctahedron

L'ottaemiottaedro è un solido non manifold avente 12 vertici, 24 spigoli e 12 facce, con caratteristica di eulero uguale a 0.

Abbiamo tentato l'implementazione sempre tramite composizione di tetraedri, ottentenendo un poliedro simile al risultato voluto. Il risultato è mostrato nelle figura seguente.

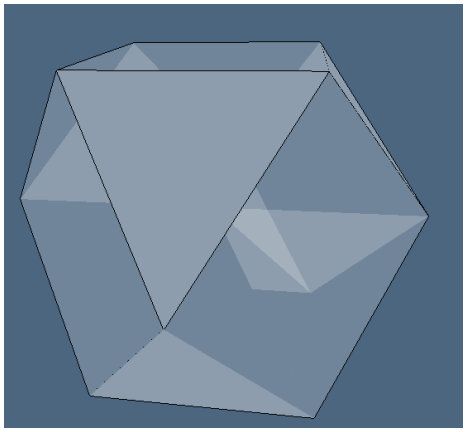


Figura 15: Octahemioctahedron

7 Conclusioni

La suite di test implementata per questo progetto ha permesso di fare una prima analisi basata sullo studio di invarianti topologici, quali la formula di Eulero,

durante l'esecuzione della pipeline di arrangiamento 2D e 3D.

Dai risultati ottenuti si è potuto evincere che in situazioni di complessi cellulari a 2-dimensioni, anche molto articolati, la formula di Eulero viene verificata correttamente sia per la figura nella sua interezza sia per ogni singola 2-faccia.

In uno spazio topologico in 3-dimensioni il discorso è diverso: qui i test sulla caratteristica di Eulero non terminano quasi mai con successo, a causa di bug all'interno dell'implementazione dell'algoritmo di arrangiamento 3D. Un altro fatto che dovrà essere considerato negli sviluppi futuri della libreria LAR è una funzione che calcoli le componenti biconnesse delle 2-facce per Poliedri 3D.

Riferimenti bibliografici

[1] Dario Di Nardo. *Repository github.* `https://github.com/DarioDN/LinearAlgebraicRepresentation.jl`.

[2] Alberto Paoluzzi. *Informatica grafica e cad : geometria, programmazione, modellazione, animazione.* Linguaggi & Programmazione. Hoepli, Milano, 2003.

[3] Matteo Petruzziello. *Repository github.* `https://github.com/petruz93/LinearAlgebraicRepresentation.jl`.

[4] Wikipedia. *Euler Characteristic.* `https://en.wikipedia.org/wiki/Euler_characteristic`.

[5] Wikipedia. *Formula di Eulero per i poliedri.* `https://it.wikipedia.org/wiki/Formula_di_Eulero_per_i_poliedri`.