Progetto Programmazione e Calcolo Scientifico 2024

Luca Agostino, Tommaso Basile, Andrea Cigna

18 giugno 2024

Sommario

Il progetto si propone di sviluppare un software avanzato in linguaggio C++ per l'analisi e la segmentazione di fratture in strutture poligonali. L'obiettivo principale è quello di individuare le tracce di fratture presenti in una data geometria poligonale e suddividere le fratture in sottopoligoni distinti sulla base delle tracce trovate. Il sistema utilizzerà le strutture dati più appropriate in termini di memoria allocata ed efficienza computazionale. Una volta identificate tutte le tracce presenti, le fratture verranno utilizzate per segmentare la geometria originale in più sottopoligoni, permettendo un'analisi dettagliata e la possibilità di eseguire ulteriori studi su ciascun frammento. Questo progetto troverà applicazione in diversi campi, inclusi l'ingegneria civile, la geologia e la scienza dei materiali, offrendo uno strumento efficace per la valutazione delle strutture fratturate e il miglioramento della loro gestione e riparazione.

Indice

1 Individuazione delle tracce

Per determinare tutte le tracce che appartengono ad una frattura, ovvero le intersezioni che si generano con altri poligoni appartenente allo stesso set di dati, procederemo con l'individuazione dei piani che contengono le fratture in modo da ottenere ottenere la retta di intersezione. A questo punto, se la questa retta interseca entrambi i poligoni, verranno generati quattro punti e in base alla loro posizione spaziale sarà possibile capire se la traccia è passante o meno per ognuno

dei poligoni analizzati. In tutti gli altri casi, la traccia non viene generata. Ripeteremo questo procedimento per ogni coppia di fratture sufficientemente vicine.

1.1 Piano passante per una frattura

Date N fratture con n_i vertici ciascuna, denominiamo con $\mathbf{P}_j^{(i)}$ il vertice j della frattura i. I range di valori sono, quindi, $i = 0, \dots, N-1$ e $j = 0, \dots, n_i - 1$. Pertanto, per ogni frattura i, abbiamo una n_i-upla $(\mathbf{P}_0^{(i)}, \dots, \mathbf{P}_{n_i-1}^{(i)})$. Cerchiamo l'equazione cartesiana del piano π contenente la frattura, dando per assunto che effettivamente i vertici forniti appartengono tutti ad uno stesso piano e che, quindi, corrispondano effettivamente a un poligono nello spazio tridimensionale. Per semplicità di notazione omettiamo nelle trattazioni successive l'apice (i), sottointendendo che ci stiamo riferendo alla traccia i-esima. Vogliamo arrivare a una scrittura di π del tipo

$$ax + by + cz = d.$$

Se denotiamo con $\mathbf{AB} = (x_B - x_A, y_B - y_A, z_B - z_A)$ il vettore congiungente $\mathbf{A}$ a $\mathbf{B}$, un generico punto $\mathbf{P} \in \mathbb{R}^3$ appartiene al piano passante per i punti $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2$ se vale che

$$(\mathbf{P}_0\mathbf{P}_1 \wedge \mathbf{P}_0\mathbf{P}_2) \cdot \mathbf{P}_0\mathbf{P} = 0, \tag{1}$$

cioè il vettore che congiunge $\mathbf{P}_0$ a $\mathbf{P}$ deve essere ortogonale al vettore normale al piano (detto giacitura) $\mathbf{P}_0\mathbf{P}_1 \wedge \mathbf{P}_0\mathbf{P}_2$. La (1) equivale a porre nullo il determinante della matrice avente per righe le componenti dei 3 vettori rispetto alla base canonica di $\mathbb{R}^3$:

$$\det \begin{pmatrix} x - x_0 & y - y_0 & z - z_0 \\ x_1 - x_0 & y_1 - y_0 & z_1 - z_0 \\ x_2 - x_0 & y_2 - y_0 & z_2 - z_0 \end{pmatrix} = 0.$$

Sviluppando il determinante con la regola di Laplace rispetto alla prima riga otteniamo un'equazione del tipo

$$a\,(x - x_0) + b\,(y - y_0) + c\,(z - z_0) = 0, \tag{2}$$

dove

$$\begin{aligned} a &= \det \begin{pmatrix} y_1 - y_0 & z_1 - z_0 \\ y_2 - y_0 & z_2 - z_0 \end{pmatrix}, \\ b &= -\det \begin{pmatrix} x_1 - x_0 & z_1 - z_0 \\ x_2 - x_0 & z_2 - z_0 \end{pmatrix}, \\ c &= \det \begin{pmatrix} x_1 - x_0 & y_1 - y_0 \\ x_2 - x_0 & y_2 - y_0 \end{pmatrix}. \end{aligned}$$

Esplicitiamo i prodotti a primo membro dell'equazione (2)

$$\begin{aligned} &a\,(x - x_0) + b\,(y - y_0) + c\,(z - z_0) = 0, \\ &ax - ax_0 + by - by_0 + cz - cz_0 = 0, \\ &ax + by + cz - ax_0 - by_0 - cz_0 = 0. \end{aligned}$$

Chiamiamo

$$d = -ax_0 - by_0 - cz_0.$$

Così facendo abbiamo ricavato l'equazione cartesiana del piano cercato

$$ax + by + cz = d,$$

con a, b, c non tutti nulli, non essendo tutti e tre i punti allineati.

1.2 Retta di intersezione tra piani

Date due fratture diverei i e j, grazie a quanto detto sopra, riusciamo a trovare l'equazione cartesiana dei due piani contententi le fratture. Nel caso in cui i due piani non siano paralleli, ci proponiamo di ricavare l'equazione in forma parametrica della retta r che si genera dall'intersezione dei due piani; siamo, cioè, in cerca di una scrittura del tipo

$$\mathbf{OP} = \mathbf{OQ} + t\mathbf{v},$$

con t parametro reale e $\mathbf{v}$ la direzione della retta. Scriviamo in un sistema le equazioni per i due piani in questo modo:

$$\begin{cases} a_1x + b_1y + c_1z = d_1 \\ a_2x + b_2y + c_2z = d_2 \end{cases}$$

dove i pedici indicano il piano rappresentato. Per i due piani definiamo con

$$\mathbf{n}_1 = \mathbf{P}_0\mathbf{P}_1 \wedge \mathbf{P}_0\mathbf{P}_2, \quad \mathbf{n}_2 = \mathbf{P}'_0\mathbf{P}'_1 \wedge \mathbf{P}'_0\mathbf{P}'_2$$

le due giaciture. La direzione individuata dalla retta di intersezione è data dal vettore normale a entrambe le giaciture dei piani:

$$\mathbf{v} = \mathbf{n}_1 \wedge \mathbf{n}_2.$$

Al fine di trovare $\mathbf{Q}$, se $c_1 \neq 0$ oppure $c_2 \neq 0$, battezziamo z come parametro libero e, ponendolo pari a 0, risolviamo il sistema lineare

$$\begin{cases} a_1x + b_1y = d_1 \\ a_2x + b_2y = d_2 \end{cases}$$

trovando $\mathbf{OQ} = (\overline{x}, \overline{y}, 0)$. Allora la retta di intersezione r è data in forma parametrica da

$$\begin{cases} x = \overline{x} + tv_x \\ y = \overline{y} + tv_y \\ z = tv_z. \end{cases}$$

Se la retta di intersezione tra i due piani è parallela al piano xy, scegliamo un altro parametro libero (ad esempio x): il nostro obbiettivo è solamente avere un punto di partenza per essa.

1.3 Punti di intersezione

Per ogni coppia di vertici consecutivi $\mathbf{P}_j, \mathbf{P}_{j+1}$ scriviamo la retta passante per entrambi r_j attraverso l'ascissa curvilinea s in questo modo:

$$\mathbf{OP} = \mathbf{OP}_j + s\mathbf{P}_j\mathbf{P}_{j+1},$$

pertanto solo per valori di s in $(0, 1)$ otteniamo un punto del lato della frattura. Consideriamo adesso due fratture, attraverso i punti 1 e 2 troviamo i piani che li contengono e la retta di intersezione (se esiste) tra le due.

Per ogni coppia di vertici consecutivi della prima frattura intersechiamo la retta r di intersezione e la retta r_j:

$$\begin{cases} \mathbf{OP} = \mathbf{OQ} + t\mathbf{v} \\ \mathbf{OP} = \mathbf{OP}j + s\mathbf{P}_j\mathbf{P}_{j+1} \end{cases}$$

per cui abbiamo

$$\mathbf{OQ} + t\mathbf{v} = \mathbf{OP}_j + s\mathbf{P}_j\mathbf{P}_{j+1},$$

ossia un sistema di 3 equazioni nelle incognite t, s. Riscriviamolo in tal modo:

$$\begin{cases} v_x t - (x_{j+1} - x_j)s = x_j - \overline{x} \\ v_y t - (y_{j+1} - y_j)s = y_j - \overline{y} \\ v_z t - (z_{j+1} - z_j)s = z_j - \overline{z} \end{cases} \tag{3}$$

Potrebbe succedere che il sistema non ammetta soluzioni, in tal caso le due rette non si intersecano, cioè la traccia non attraversa il lato $\mathbf{P}j\mathbf{P}j + 1$. Altrimenti, in caso di sistema determinato otterremo la coppia (s, t). Se $s \in [0, 1]$ otteniamo effettivamente che il punto di intersezione giace sul lato $\mathbf{P}j\mathbf{P}j + 1$ della prima frattura. Iteriamo il processo per tutti gli n_i vertici della frattura e troveremo due punti $\mathbf{Q}_1$ e $\mathbf{Q}_2$, trovati attraverso l'intersezione tra r e due lati distinti della frattura.

Alla stessa maniera, effettuiamo questi calcoli per la seconda frattura, trovando o intersezione nulla per ogni lato o due altri due punti $\mathbf{Q}_3$ e $\mathbf{Q}_4$.

Nel caso in cui troviamo i 4 punti di intersezione, associamo ad ognuno il valore di t corrispondente trovato nel sistema (5) e costruiamo due intervalli dell'asse reale ordinando i valori dei parametri: otteniamo allora (a, b) e (c, d), dove $a, b \in \{t_1, t_2\}$ e $c, d \in \{t_3, t_4\}$. Se poniamo $a = \min\{t_1, t_2\}$, $b = \max\{t_1, t_2\}$, $c = \min\{t_3, t_4\}$ e $d = \max\{t_3, t_4\}$, attraverso una funzione che calcoli l'intersezione tra questi due intervalli, verifichiamo che

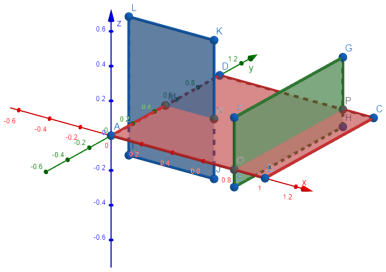

Figura 1: Esempio con 3 fratture e 2 tracce

(i) se $(a, b) \cap (c, d) = (a, c)$, allora si tratta di una traccia non passante e i suoi vertici sono $\mathbf{Q}_1$ e $\mathbf{Q}_3$;

(ii) se $(a, b) \cap (c, d) = (a, b)$, allora si tratta di una traccia passante per il poligono 1 e i suoi vertici sono $\mathbf{Q}_1$ e $\mathbf{Q}_2$;

(iii) se $(a,b) \cap (c,d) = (b,d)$, allora si tratta di una traccia non passante e i suoi vertici sono $\mathbf{Q}_2$ e $\mathbf{Q}_4$;

(iv) se $(a,b) \cap (c,d) = (c,d)$, allora si tratta di una traccia passante per il poligono 2 e i suoi vertici sono $\mathbf{Q}_3$ e $\mathbf{Q}_4$.

Attraverso questo algoritmo abbiamo trovato i vertici delle tracce, il loro valore di **Tips** e anche le intersezioni dei prolungamenti di esse sui poligoni, il che risulterà molto utile in seguito.

2 Determinazione dei sotto-poligoni generati per ogni frattura

L'idea alla base della determinazione dei sotto-poligoni è la seguente: analizzando una frattura alla volta, esaminiamo tutte le tracce per ordine di lunghezza e valore di **Tips** e dividiamo il poligono in cui si trova la traccia in due sotto-poligoni. Il procedimento si ferma non appena la traccia non è più passante per il sotto-poligono che stiamo analizzando, in questo caso dividiamo un'ultima volta il poligono e passeremo alla traccia successiva.

L'approccio che abbiamo utilizzato per implementare la determinazione dei sotto-poligoni è iterativo e può essere sintetizzato come segue.

2.1 Individuazione del punto iniziale

Per ogni frattura dovremo creare una cella 2D iniziale che rappresenti il poligono di partenza, quindi verranno memorizzati tutti i vertici e i lati della frattura. Ad ogni vertice corrispondono le rispettive coordinate, mentre ad ogni lato sono assegnati gli identificativi dei vertici che lo generano.

Per ogni frattura $j = 1, \dots, N$, consideriamo, una per volta, le tracce che le appartengono. Queste avranno una coppia di vertici $(\mathbf{T}_1, \mathbf{T}_2)$, quindi selezioniamo la cella 1D più vicina al punto $\mathbf{T}_1$ attraverso i valori dell'ascissa curvilinea s più prossimi a 0; tale cella verrà selezionata all'interno della struttura attualmente presente, individuata dalla coppia $(\mathbf{P}_j, \mathbf{P}_{j+1})$. Il punto situato alla sinistra di $\mathbf{T}_1$ così individuato sarà il punto iniziale della suddivisione dei sotto-poligoni. Consideriamo il sistema

$$\begin{cases} \mathbf{OP} = \mathbf{OT}_1 + s\mathbf{T}_1\mathbf{T}_2 \\ \mathbf{OP} = \mathbf{OP}_j + t\mathbf{P}_j\mathbf{P}_{j+1} \end{cases} \tag{4}$$

Risolviamolo per determinare l'intersezione tra la traccia e ogni lato. Per cui abbiamo

$$\mathbf{OT}_1 + s\mathbf{T}_1\mathbf{T}_2 = \mathbf{OP}_j + t\mathbf{P}_j\mathbf{P}_{j+1},$$

ossia un sistema di 3 equazioni nelle incognite t, s. Riscriviamolo in tal modo:

$$\begin{cases} (\tilde{x}_2 - \tilde{x}_1)s - (x_{j+1} - x_j)t = x_j - \tilde{x}_1 \\ (\tilde{y}_2 - \tilde{y}_1)s - (y_{j+1} - y_j)t = y_j - \tilde{y}_1 \\ (\tilde{z}_2 - \tilde{z}_1)s - (z_{j+1} - z_j)t = z_j - \tilde{z}_1 \end{cases} \tag{5}$$

Potrebbe succedere che il sistema non abbia soluzione, in tal caso le rette sono parallele. Altrimenti in caso di sistema determinato otterremo la coppia (s, t). Se $t \in [0, 1]$ otteniamo effettivamente che il punto di intersezione giace sul lato $\mathbf{P}_j\mathbf{P}_{j+1}$, inoltre se $s \in [0, 1]$ allora il punto si troverà sulla traccia stessa. Se, invece, s non è mai compreso nell'intervallo $[0, 1]$, significa che la traccia è interamente contenuta nella cella 2D. Ripetiamo tale procedura per tutte le celle 1D relative alla frattura di riferimento e scegliamo di volta in volta il valore di s più vicino a 0 (nel caso in cui sia effettivamente uguale a 0 avremo che il punto iniziale della traccia giace su un lato). Quindi, memorizziamo in `idPreviousPoint` e in `idSuccessivePoint` i vertici del lato di partenza che è relativo al valore di s più vicno a 0, in `idIntersectionEdge` l'identificativo del lato $\mathbf{P}_j\mathbf{P}_{j+1}$ trovato, così come in `idIntersectionPoint` il punto di intersezione stesso individuato attraverso la risoluzione del sistema sopra citato.

Adesso dividiamo la cella 1D che contiene il punto di intersezione in due nuove celle 1D, queste verranno memorizzate con codice identificativo e vertici nella struttura `PolygonalMesh`. Inoltre, ad ogni cella 1D creata verranno assegnati:

1. un booleano `isOn1D` per indicare se è attivo (*true*), cioè fa parte della struttura corrente, oppure se non lo è ($false$);
2. i codici identificativi delle fratture adiacenti al lato, al massimo due.

Quindi la cella 1D appena divisa assumerà il booleano $false$ poiché esce dalla struttura corrente, mentre le due nuove celle 1D create acquisiscono l'attivazione e l'adiacenza alla cella 2D che stiamo analizzando. Anche per le celle 2D, che saranno sottoposte a suddivisioni, associamo un booleano `isOn2D`, per stabilire se la cella 2D appartenga ancora o no alla mesh.

2.2 Creazione dei sotto-poligoni

Prima di procedere con la suddivisione del poligono, dovremo fare attenzione a due casi:

(i) se la traccia è interamente contenuta nel poligono allora dovremo suddividerlo in due sotto-poligoni (aggiornando le celle 2D adiacenti al lato di intersezione) e passare alla traccia successiva;

(ii) se la traccia continua su un altro poligono al di fuori di quello che stiamo analizzando, dovremo ripetere l'operazione di suddivisione anche per il poligono successivo, il cui identificativo si trova nel dato dei vicini al lato di intersezione.

Per creare due nuove celle 2D, che rappresenteranno i due sotto-poligoni, è necessario assegnare alle celle i rispettivi lati creati, attivarli attraverso il booleano `isOn1D` e spegnere il poligono da cui provengono ponendo il campo `isOn2D` della cella al valore *false*.

Partendo dal vertice `idSuccessivePoint`, e spostandoci in senso antiorario sul poligono da suddividere, assegneremo vertici e lati ai sotto-poligoni in base alla seguente casistica. Un lato e un vertice saranno inseriti:

1. nel primo sotto-poligono, se il punto di intersezione con la retta della traccia non è ancora stato raggiunto;

2. nel secondo sotto-poligono, se il punto di intersezione con la retta della traccia è già stato superato.

Per verificare la condizione di intersezione ripetiamo, per tutti i lati, il sistema (4). Quindi, se esiste l'intersezione e il parametro $t \in [0, 1]$, allora il secondo estremo della traccia sta sul lato considerato che verrà diviso e disattivato, inoltre verrano memorizzati:

- `idNewIntersectionPoint` il secondo punto di suddivisione del poligono;
- `idNewPreviousPoint` il punto finale del lato contenente l'intersezione;
- `idNewSuccessivePoint` il punto di partenza del lato contenente l'intersezione;

Dopo aver creato i due sotto-lati ed averli memorizzati correttamente, è possibile continuare l'assegnazione dei lati del poligono originale nel secondo sotto-poligono. Il passaggio da un poligono all'altro può essere segnalato da un booleano che si attiva non appena troviamo l'intersezione. Per ultimo aggiungiamo ad entrambi i sotto-poligoni il lato che li divide con l'accortezza di assegnare, come vicini a questo lato, le due nuove celle create. A questo punto, se la traccia iniziale non era interamente contenuta nel poligono, vengono assegnati:

- `idIntersectionPoint` a `idNewIntersectionPoint`;
- `idPreviousPoint` a `idNewPreviousPoint`;
- `idSuccessivePoint` a `idNewSuccessivePoint`;

in modo tale che il nuovo punto di partenza per la suddivisione del poligono adiacente sia il secondo punto di intersezione della prima suddivisione.

Questo procedimento viene iterato per ogni traccia del poligono originale finché il punto di intersezione della traccia con i lati del poligono, ottenuto mediante il sistema (4), ha l'ascissa curvilinea $s \in [0, 1]$. Non appena troveremo un valore al di fuori di questo intervallo, concluderemo la divisione del poligono in cui è presente la traccia, aggiorneremo il poligono adiacente e passeremo alla traccia successiva, fino ad esaurirle completamente.

L'aggiornamento del poligono adiacente va fatto solo nel caso in cui questo esista, e consiste nella sostituzione dell'unico lato che risulterà spento, ovvero il lato adiacente, con i due nuovi sotto-lati generati dall'ultima suddivisione e con l'aggiunta dell'ultimo punto di intersezione.

2.3 Esempio

Di seguito considereremo un esempio chiarificatore. Le notazioni fin qui adottate saranno semplificate ulteriormente al fine di favorire al meglio l'esposizione. Consideriamo una sola frattura (Figura 1) identificata da 4 vertici: **A**, **B**, **C**, **D**.

Su questa frattura consideriamo due tracce: la più lunga **OP** e la più corta **RS**. Una volta trasferite le informazioni della frattura nella struttura `PolygonalMesh`, partiamo per ordine di lunghezza e scegliamo la traccia

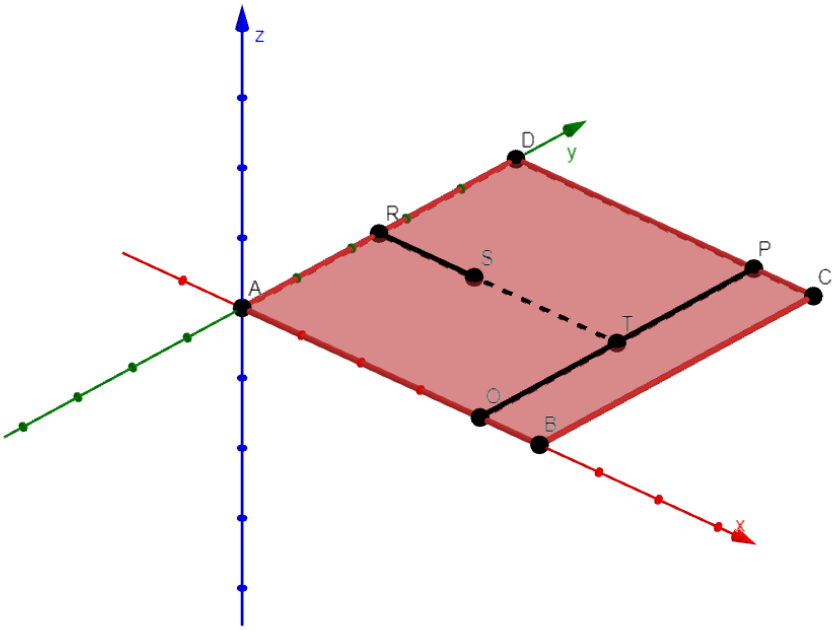

Figura 2: Esempio

OP. Per ogni lato di **ABCD**, intersechiamo, usando il sistema (4), la traccia con esso: questo sistema avrà una soluzione con $t \in [0,1]$ solo per i lati **AB** e **CD**. Scegliamo come `idIntersectionEdge` l'identificativo di **AB** e aggiungiamo alla mesh il punto **O** con un nuovo identificativo (sarà 4, in quanto i primi 4 vertici avranno rispettivamente 0,1,2,3). `idPreviousPoint`, `idIntersectionPoint` e `idSuccessivePoint` saranno, rispettivamente, 0, l'identificativo di **A**, 4 e 1, l'id di **B**. Procediamo allo spegnimento di **AB** e inseriamo nella struttura le nuove sotto-celle 1D **AO** e **OB**, attivandole e inserendo nella struttura di adiacenza gli identificativi dei due nuovi sotto-poligoni che andremo a costruire (saranno quelli di **OBCP** e **OPDA**). Prima di operare le intersezioni con tutti i lati, provvediamo ad inserire al primo sotto-poligono che genereremo il lato **OB**. Ora, ciclando sui lati di **ABCD** a partire da `idSuccessivePoint` (ossia **B**), intersechiamo la retta contenente la traccia con ognuno di essi:

- i vertici **B**, **C** e il lato **BC** apparterrano al primo sottopoligono che genereremo attraverso il taglio. Aggiorniamo, come sempre, anche la struttura di adiacenza;
- il lato **CD** genererà una cella 0D (il punto **P**) e altre due sotto-celle 1D, alla stessa maniera di **AB**: perciò dovremo procedere all'inserimento del vertice **C** nel primo sotto-poligono, allo spegnimento del lato **CD**, all'aggiornamento delle adiacenze dei due nuovi sotto-lati generati **CP** e **PD** e all'assegnazione di essi a **OBCP** e **OPDA**. A questo punto, ogni lato che andremo ad aggiungere apparterrà al secondo sotto-poligono.

- Continuiamo l'inserimento di vertici e lati nel secondo sotto-poligono fino all'esaurimento dei lati, in questo caso fino al lato **DA**.

Ci resta da trattare il lato **AB**. Abbiamo già provveduto ad inserire **OB** al primo sotto-poligono; ora possiamo inserire i vertici **A**, **O** e il lato **AO** al primo sotto-poligono. Provvediamo, infine, ad aggiungere il lato **OP** ai due sotto-poligoni generati, aggiungendo ad esso le rispettive informazioni di adiacenza (i suoi vicini saranno entrambi i nuovi sotto-poligoni). Possiamo adesso procedere allo spegnimento del booleano relativo alla presenza della cella 2D **ABCD**, poichè è stata suddivisa e non più esistente.

Allo stesso modo, per la traccia **RS**, troveremo come punto iniziale il punto **R** e procederemo identicamente. Dovremo avere l'accortezza di notare che il poligono da suddividere sarà **OPDA** e che il poligono **OBCP** dovrà essere modificato, in quanto gli verrà aggiunto il nuovo vertice **T** e i nuovi lati **PT** e **TO**.

3 Documentazione UML

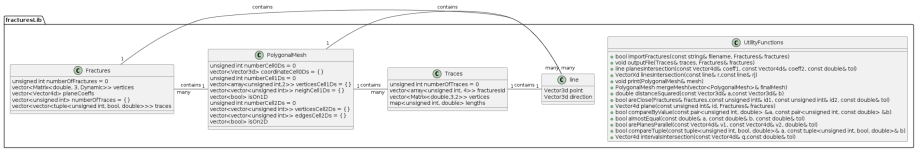

Figura 3: Documentazione UML

L'immagine rappresenta le strutture logiche e le funzioni utilizzate all'interno del progetto e i relativi collegamenti tra di esse.