

**SUPSI**

# Nuvole di punti ad alta densità: visualizzazione ed interazione

---

Studente/i

**Sala Nicholas**

Relatore

**Landoni Fabio**

---

Correlatore

**Guidi Roberto**

---

Committente

**Landoni Fabio**

---

Corso di laurea

**Ingegneria informatica**

Modulo

**M00009 Progetto di diploma**

---

Anno

**2019**

---

Data

**10 settembre 2019**

STUDENTSUPSI



Questo lavoro viene sottomesso  
a parziale requisito  
per il conseguimento del

**BACHELOR IN INGEGNERIA INFORMATICA**

SCUOLA UNIVERSITARIA PROFESSIONALE  
DELLA SVIZZERA ITALIANA  
DIPARTIMENTO TECNOLOGIE INNOVATIVE  
MANNO, SVIZZERA

STUDENTSUPSI



# Indice

<b>Abstract</b>	<b>1</b>
<b>Introduzione</b>	<b>3</b>
<b>1 Motivazione e contesto</b>	<b>5</b>
1.1 Contesto . . . . .	5
1.1.1 Nuvola di punti . . . . .	5
1.1.2 Nuvola di punti ad alta densità . . . . .	7
1.1.3 Classificazione . . . . .	7
1.2 Motivazioni . . . . .	8
<b>2 Problema</b>	<b>9</b>
2.1 Efficente interazione con nuvole ad alta densità . . . . .	9
2.2 Risposta grafica all'interazione con le classificazioni . . . . .	9
2.3 Visualizzazione parallela di due nuvole . . . . .	10
2.4 Vincoli e requisiti . . . . .	10
2.5 Obiettivi di progetto . . . . .	10
<b>3 Stato dell'arte</b>	<b>11</b>
3.1 Acquisizione di nuvole di punti . . . . .	11
3.2 Formati nuvole di punti . . . . .	11
3.3 Algoritmi e strutture per una efficiente gestione di nuvole di punti . . . . .	13
3.3.1 Soglia al numero di punti . . . . .	13
3.3.2 Culling . . . . .	13
3.3.2.1 Frustum Culling . . . . .	14
3.3.2.2 Backface Culling . . . . .	14
3.3.3 Level Of Details (LOD) . . . . .	14
<b>4 Approccio al problema</b>	<b>19</b>
4.1 Studio dello stato dell'arte . . . . .	19
4.2 Metodo iterativo . . . . .	19
4.3 Framework di sviluppo . . . . .	21

4.4 Strumenti tecnologici . . . . .	22
4.4.1 Piattaforma . . . . .	22
4.4.2 Sistema software per il controllo di versione . . . . .	22
<b>5 Implementazione</b>	<b>25</b>
5.1 Scelta libreria . . . . .	25
5.1.1 Individuazione . . . . .	25
5.1.2 Prima iterazione . . . . .	27
5.1.3 Seconda iterazione . . . . .	28
5.1.3.1 Parametri misurati . . . . .	28
5.1.3.2 Nuvole di punti utilizzate per le prove . . . . .	29
5.1.3.3 Computer utilizzati . . . . .	29
5.1.3.4 Strumenti di supporto . . . . .	30
5.1.3.5 Risultati seconda iterazione . . . . .	30
5.1.3.6 Seconda selezione . . . . .	32
5.1.4 Terza iterazione . . . . .	33
5.1.4.1 Open3D . . . . .	33
5.1.4.2 BA_Pointcloud . . . . .	34
5.1.5 Scelta conclusiva . . . . .	34
5.2 Implementazione prototipo finale . . . . .	40
5.2.1 Convertitore . . . . .	40
5.2.1.1 Struttura dei nodi . . . . .	42
5.2.1.2 Suddivisione dell'ambiente tridimensionale . . . . .	42
5.2.1.3 Identificazione nodi . . . . .	42
5.2.1.4 Salvataggio delle meta-informationi . . . . .	44
5.2.2 Visualizzatore . . . . .	46
5.2.2.1 Loader . . . . .	46
5.2.2.2 Traversal . . . . .	48
5.2.2.3 Visualizer . . . . .	51
5.2.3 Modifiche apportate ad Open3D . . . . .	52
<b>6 Risultati</b>	<b>55</b>
6.0.1 Visualizzatore con meccanismo di LOD . . . . .	55
6.0.2 "Rilevatore" di classi . . . . .	57
<b>7 Conclusione</b>	<b>59</b>
7.1 Sviluppi futuri . . . . .	59
7.2 Competenze acquisite . . . . .	60
7.3 Ringraziamenti . . . . .	60

<b>A Scheda di progetto</b>	<b>63</b>
<b>B Email</b>	<b>65</b>
<b>C Esempi formati nuvole di punti</b>	<b>69</b>
C.1 XYZ . . . . .	69
C.2 PTS . . . . .	69
C.3 PTX . . . . .	69
C.4 PLY . . . . .	70
<b>D Risultati prove librerie (seconda iterazione)</b>	<b>71</b>
D.1 Legenda parametri . . . . .	71
D.2 Misurazioni ottenute . . . . .	71



# Abstract

A team at University of Applied Sciences and Arts of Southern Switzerland is developing a system for the classification of high-density point clouds. To verify the goodness of the classification algorithm a tool to visualize the point clouds is needed. Moreover, the tool must provide the possibility to fluidly interact with the cloud (move, zoom,...). Due to the large size of the point clouds and their high density, the implementation of such a tool is not a trivial task. In this document the steps taken to develop a prototype are presented. A short description of the state of the art will be provided. The libraries taken into account, the experiments made with them and the obtained results will be described in detail. After presenting the criteria used to choose the library for the final prototype, it will be explained how the prototype has been implemented and the adjustments done to the chosen library. In the final part of this document possible future developments will be exposed

\* \* \*

Un team presso la Scuola Universitaria Professionale della Svizzera Italiana sta realizzando un sistema per la classificazione di nuvole di punti ad alta densità. Per valutare l'accuratezza dell'algoritmo di classificazione è necessario realizzare uno strumento in grado di visualizzare le nuvole di punti. Oltre alla visualizzazione, questo strumento dovrà permettere l'interazione, in maniera fluida, con le nuvole (muoverle, zoom, ...). A seguito della dimensione delle nuvole e della loro elevata densità, la realizzazione di un'applicazione in grado di soddisfare le esigenze appena esposte risulta un'operazione tutt'altro che banale. In questo documento verranno presentati tutti i passi fatti per la creazione di un prototipo. Dopo una breve esposizione dello stato dell'arte, saranno descritte in dettaglio le librerie prese in considerazione, gli esperimenti fatti e i risultati ottenuti. Dopo aver presentato i criteri utilizzati per la scelta della libreria da usare nel prototipo finale, verrà spiegato come quest'ultimo è stato implementato e quali modifiche sono state apportate alla libreria scelta. Nella parte finale di questo documento saranno esposti alcuni possibili sviluppi futuri.



# Introduzione

Nel Capitolo 1 verrà descritto il contesto in cui questo progetto di diploma è inserito. Verranno inoltre spiegate le motivazioni sulle quali esso è basato. Il lettore sarà così introdotto nell'ambito in cui si è svolto il presente lavoro.

Il Capitolo 2 espone il problema che viene affrontato con questo progetto. Si avrà quindi una chiara visione di quali sono le difficoltà da superare. Saranno descritti anche i requisiti che il risultato di questo lavoro di diploma deve rispettare.

Lo stato dell'arte riferito al contesto del progetto, verrà descritto nel Capitolo 3. Sarà possibile trovare l'elenco degli algoritmi maggiormente utilizzati in questo ambito. Il lettore potrà capire come attualmente problemi simili vengono gestiti e risolti.

Nel Capitolo 4 verrà spiegato il tipo di approccio che si è scelto di adottare. Si potrà conoscere il metodo adottato in ogni fase del lavoro. Questo capitolo rappresenta il "come" si è deciso di operare.

Le analisi, le scelte e l'implementazione fatte vengono esposte nel Capitolo 5. Lo sviluppo della parte software viene delineato in ogni sua fase e caratteristica.

Nel Capitolo 6 verrà presentato e analizzato il risultato ottenuto nelle sue componenti fondamentali.

Gli sviluppi futuri applicabili al risultato ottenuto verrano delineati nel Capitolo 7. Il lettore potrà trovare inoltre un'analisi di tutte le competenze che il progetto di diploma ha permesso di acquisire.



# Capitolo 1

## Motivazione e contesto

Il presente progetto di diploma opera in un contesto ben definito che verrà descritto qui di seguito. Inoltre esistono delle motivazioni che hanno portato alla creazione di questo lavoro di diploma, anch'esse saranno spiegate. Questo consentirà di capire qual'è l'ambito in cui si opera.

### 1.1 Contesto

Il contesto all'interno del quale si innesta questo progetto di diploma è quello delle nuvole di punti ([Sezione 1.1.1]). Nello specifico esiste un progetto più ampio che la Scuola Universitaria Professionale della Svizzera Italiana (SUPSI) sta svolgendo in collaborazione con un'azienda della svizzera tedesca che opera nel campo dell'acquisizione di nuvole di punti. L'Institute for Information Systems and Networking (ISIN) e il Dalle Molle Institute for Artificial Intelligence (IDSIA) stanno realizzando per il progetto citato in precedenza un sistema per lo stoccaggio di nuvole di punti, la loro classificazione ([Sezione 1.1.3]), il salvataggio delle classificazioni e l'accesso a queste nuvole classificate. ISIN si occuperà dello stoccaggio e dell'accesso, mentre IDSIA preparerà l'algoritmo di classificazione.

Questo lavoro di diploma si inserisce quindi in questo sistema, cercando di offrire uno strumento adatto alla visualizzazione delle nuvole di punti, attraverso il quale si possa valutare la qualità delle classificazioni prodotte. In Figura 1.1 viene illustrato uno schema rappresentante le fasi attraversate da una nuvola. Il lavoro di diploma si inserisce quindi nella fase 5. Successivamente verranno analizzate le parti principali che compongono il contesto.

#### 1.1.1 Nuvola di punti

Una nuvola di punti è un insieme di punti caratterizzati dalla loro posizione in un sistema di coordinate e da eventuali valori aggiuntivi ad essi associati (quale ad esempio il colore).

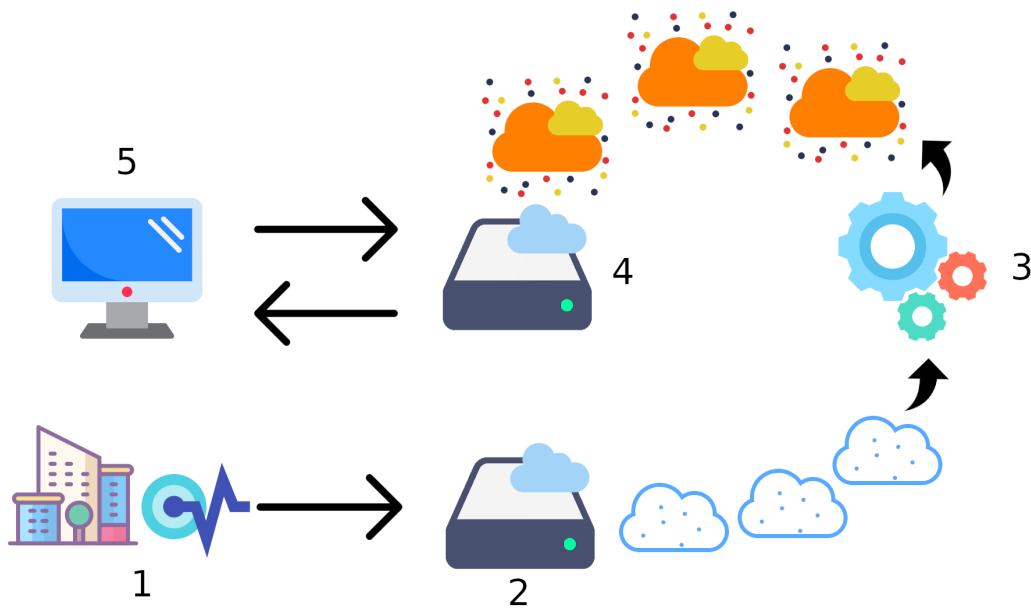


Figura 1.1: Diagramma rappresentante le fasi attraversate da una nuvola di punti. Fasi: 1 acquisizione dall'ambiente reale, 2 stoccaggio, 3 classificazione attraverso l'algoritmo di classificazione, 4 salvataggio della nuvola classificata, 5 visualizzazione della nuvola classificata.

Servono per rappresentare strutture tridimensionali come oggetti ([Figura 1.2]) o superfici in rilievo. I loro impieghi possono essere svariati, per esempio:

1. creazione di modelli Computer-Aided Design (CAD)<sup>1</sup> tridimensionali.
2. controllo qualità dei prodotti: nell'industria per verificare se i pezzi realizzati soddisfano le specifiche e le tolleranze del progetto.
3. diagnostica per immagini in campo medico.
4. creazione percorso di saldatura per robot industriali.

Spesso le nuvole di punti servono da partenza per la creazione di superfici come le mesh poligonali<sup>2</sup> utilizzate in grafica computerizzata e nella modellazione CAD. Inoltre una nuvola di punti permette di poter interagire con essa (muovendole, ruotandole ...) in un ambiente 3D, capacità che una semplice foto non possiede.

Il numero di punti in una nuvola è una parametro rilevante come descritto di seguito.

<sup>1</sup>Utilizzo di tecnologie software per supportare l'attività di progettazione di manufatti sia virtuali che reali.

<sup>2</sup>Reticolo che definisce un oggetto nello spazio composto da vertici spigoli e facce.

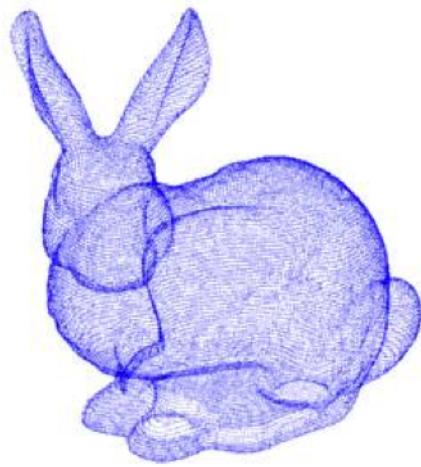


Figura 1.2: Esempio di una nuvola di punti rappresentante un coniglio. (Fonte: [1])

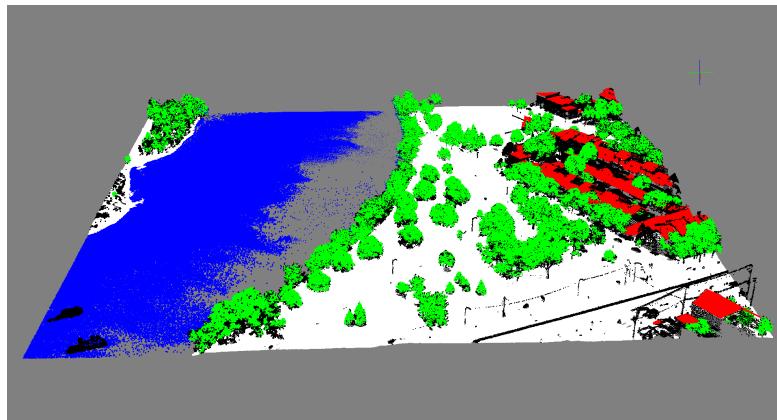


Figura 1.3: Esempio di classificazione su una nuvola di punti. (Fonte: [2])

### 1.1.2 Nuvola di punti ad alta densità

Con il termine nuvola di punti ad alta densità si intende una nuvola di punti con un numero di punti per unità di area molto elevato. Come vedremo nella Sezione 2.1 la grande quantità di punti rende difficile la loro gestione attraverso un software di visualizzazione.

### 1.1.3 Classificazione

Il termine classificazione viene utilizzato per varie attività che si possono ricondurre alla gestione delle conoscenze [3]. Nello specifico in questo progetto, si intende la divisione o la distribuzione in classi<sup>3</sup> dei punti presenti in una nuvola di punti, al fine di poter individuare gli oggetti che la compongono. Ad esempio come si può osservare in Figura 1.3 la

<sup>3</sup>Categorie attraverso le quali è possibile suddividere qualcosa.

classificazione ha categorizzato i punti in: alberi (verde), case (rosso), fiume (blu), terreno (bianco). Solitamente questo lavoro viene effettuato da un algoritmo di classificazione. Con il termine algoritmi di classificazione si intendono degli algoritmi in grado di individuare le classi all'interno di dati forniti, in questo contesto all'interno di una nuvola di punti.

## 1.2 Motivazioni

Nel sistema in realizzazione per il progetto ISIN-IDSIA di cui si parlava all'inizio di questo capitolo, in cui questo lavoro di diploma si inserisce, manca uno strumento che faciliti la valutazione dell'accuratezza delle classificazioni prodotte dall'algoritmo di IDSIA. Questa è la motivazione principale che ha portato alla creazione di questo progetto di diploma. Dato che la maggior parte delle nuvole di punti usate nel progetto sono ad alta densità ([Sezione 1.1.2]) nasce l'esigenza di dover visualizzare e valutare le classificazioni di questo tipo di nuvole che rappresentano oggetti di difficile gestione vista la loro dimensione in termini di punti.

# Capitolo 2

## Problema

In questo capitolo vedremo quali sono i principali problemi con cui si è confrontati quando si devono visualizzare nuvole di punti ad alta densità. Inoltre poter offrire le funzionalità per valutare la bontà dell'algoritmo di classificazione rappresenta un altro problema. Le difficoltà che andranno superate vengono descritte nel dettaglio successivamente. I vincoli e i requisiti presenti saranno delineati. La scheda di progetto dove è possibile trovare la descrizione originale viene allegata nell'Appendice A.

### 2.1 Efficiente interazione con nuvole ad alta densità

Un problema nasce quando si tratta di interagire in modo fluido con nuvole di punti ad alta densità ([Sezione 1.1.2]). Spesso le dimensioni di queste nuvole non consentono il loro completo caricamento in Random Access Memory (RAM) e, anche se questo caricamento fosse possibile, gestire per ogni frame<sup>1</sup> generato dal motore grafico<sup>2</sup> un immenso numero di punti renderebbe l'esperienza utente insoddisfacente proprio per l'elevata computazione necessaria tra la creazione di un frame e l'altro. Il valore dei Frame Per Second (FPS) non sarebbe adeguato per ottenere un applicativo fluido.

### 2.2 Risposta grafica all'interazione con le classificazioni

Un altro problema da risolvere è dare la possibilità, attraverso delle risposte grafiche, di individuare le classificazioni ([Sezione 1.1.3]) trovate precedentemente nella nuvola. Così che si possa verificare la qualità dell'algoritmo di classificazione utilizzato verificando la coerenza tra punti evidenziati e classe trovata. Per esempio: selezionando un punto appartenente ad un albero, si potrebbe verificare che la classe identificata dall'algoritmo di classificazione ([Sezione 1.1.3]) per il punto in questione sia effettivamente un albero.

<sup>1</sup> Immagini individuali in una sequenza di immagini.

<sup>2</sup> Nucleo software in grado di gestire e visualizzare sullo schermo oggetti di grafica tridimensionale.

## 2.3 Visualizzazione parallela di due nuvole

Sempre al fine di valutare la bontà degli algoritmi di classificazione ([Sezione 1.1.3]), si desidera poter visualizzare due nuvole in contemporanea in modo da poter trovare visivamente le differenze tra le classificazioni di algoritmi diversi o semplicemente di avere due tipi di visualizzazione diversi della stessa nuvola. Per esempio: se si volessero trovare le differenze tra due versioni dello stesso algoritmo di classificazione si potrebbero visualizzare contemporaneamente le nuvole classificate con versioni differenti.

## 2.4 Vincoli e requisiti

Lo sviluppo software del progetto di diploma presenta dei vincoli. Inanzitutto come sistema operativo per l'uso del prototipo finale viene imposto un ambiente linux. Questo perchè il sistema operativo utilizzato da IDSIA (istituto con il compito di classificare le nuvole) è una distribuzione linux. Quindi il software creato dovrà essere compatibile con tali sistemi. Inoltre il prototipo che si andrà a realizzare dovrà essere basato su una libreria open-source. Questo requisito è anche un aiuto allo sviluppo, in quanto sarà possibile andare a leggere ed eventualmente modificare, licenza permettendo, il codice sorgente delle librerie qualora ve ne fosse la necessità.

## 2.5 Obiettivi di progetto

A seguito di quanto sin qui esposto, il seguente lavoro si propone di realizzare un prototipo che consenta la visualizzazione e l'interazione (muovere, zoom, ...) con nuvole di punti ad alta densità. Il prototipo dovrà essere in grado di offrire una buona esperienza utente anche con grandi nuvole ad elevata densità di punti.

Per fare ciò si dovranno intraprendere i seguenti passi:

- Identificare alcune librerie che possano essere di supporto alla risoluzione dei problemi sopra esposti.
- Valutare le funzionalità e comprendere il comportamento delle librerie prese in esame. Per fare ciò sarà necessario definire un sistema di valutazione oggettivo delle stesse.
- Individuare la libreria maggiormente idonea sulla quale basare la realizzazione del prototipo.
- Sviluppare il prototipo ed eventualmente adattare la libreria scelta al fine di meglio soddisfare le esigenze di progetto.

Il tutto dovrà essere fatto rispettando i vincoli e i requisiti summenzionati ([Sezione 2.4]).

# Capitolo 3

## Stato dell'arte

Nel seguente capitolo verrà descritto lo stato dell'arte relativo al contesto ([Sezione 1.1]). Durante il corso degli anni sono state introdotte varie tecniche con lo scopo di gestire al meglio le nuvole di punti. Parte di queste tecniche sono generali ottimizzazioni che vengono fatte oggigiorno nella grafica computerizzata. Mentre certe trovano una particolare applicazione proprio nel campo delle nuvole di punti. Oltre a tali tecniche verranno esposti i formati in cui le nuvole di punti vengono salvate, non prima però di parlare della loro acquisizione.

### 3.1 Acquisizione di nuvole di punti

Le nuvole di punti ([Sezione 1.1.1]) vengono catturate attraverso l'utilizzo di scanner laser ([Figura 3.1]), quindi attraverso dei raggi laser che vengono proiettati sulla superficie da scansionare, riflessi, e analizzati per ottenere le informazioni del punto associato. Una delle tecniche più utilizzate per questa scansione è detta Light Detection and Ranging (LiDAR) che consiste nel telerilevamento<sup>1</sup>. Questo metodo viene particolarmente utilizzata per analizzare le informazioni della superficie terrestre. Infatti questi scanner muniti di speciali ricevitori GPS sono spesso montati ad un elicottero o ad un aeroplano, proprio con lo scopo di catturare parti della superficie terrestre in nuvole di punti.

### 3.2 Formati nuvole di punti

Non esiste un formato standard per il salvataggio di nuvole di punti, vi sono più formati suddivisi principalmente in due grandi categorie: formato binario e formato testuale. I formati binari permettono di risparmiare memoria, mentre i formati testuali occupano più memoria ma sono leggibili dall'umano. Per degli esempi dettagliati sui formati testuali si consulti l'Appendice C. In seguito vengono elencati i formati principali. Le fonti di queste informazioni

<sup>1</sup> Disciplina tecnico-scientifica che permette di ricavare informazioni sull'ambiente e su oggetti posti a distanza da un sensore mediante misure di radiazione elettromagnetica. (Fonte: [4])



Figura 3.1: Leica RTC 360 LiDAR scanner. (Fonte: [5])

sono : [6] [7].

Formati testuali:

- XYZ: Formato non standard, composto semplicemente da una linea contenente valori XYZ per ogni punto. Non essendo uno standard lo si può trovare anche in altre forme (per esempio con l'aggiunta anche dei valori rgb) ([Appendice C.1]).
- PTS: Formato composto da una prima linea indicante il numero di punti contenuti all'interno del file, e successivamente per ogni punto una linea indicante valori XYZ, intensità<sup>2</sup>, RGB ([Appendice C.2]).
- PTX: Composto da un'intestazione contenente informazioni come numero di colonne, numero di righe, posizione scanner e matrice di trasformazione che consente di trasformare più scansioni in un unico sistema di coordinate. Questo formato prevede infatti più scansioni. Successivamente all'intestazione i punti vengono salvati nel classico formato XYZIRGB, ossia le 3 coordinate spaziali (XYZ), l'intensità (I) e il colore (RGB) ([Appendice C.3]).
- PLY: Formato ideato nel laboratorio di grafica di Standford, disponibile in una versione binaria e in una versione ASCII. PLY salva le nuvole di punti come una collezione di poligoni secondo una struttura che prevede un'intestazione, una lista di vertici e una lista di facce ([Appendice C.4]).

Formati binari:

- LAS: Formato binario, utilizzato per gestire dati provenienti da macchine che utilizzano LiDAR ([Sezione 3.1]) come metodo di scansione.

---

<sup>2</sup>Valore della luce riflessa dalla superficie.

```

1 if self.points.GetNumberOfPoints() < self.maxNumPoints:
2     self.points.InsertNextPoint(point[:])
3 else:
4     r = random.randint(0, self.maxNumPoints)
5     self.points.SetPoint(r, point[:])
6

```

Figura 3.2: Esempio di implementazione di una soglia al numero di punti. (Codice adattato da [8].)

- LAZ: Versione compressa di LAS.

### 3.3 Algoritmi e strutture per una efficiente gestione di nuvole di punti

Oggi giorno ottenere nuvole di punti altamente dense è un'operazione rapida e relativamente semplice. Per questo motivo diventano necessarie tecniche in grado di gestire queste enormi quantità di dati che altrimenti porterebbero ad esperienze utente insoddisfacenti in termini di visualizzazione. Per poter interagire (visualizzare, muovere, ruotare, scalare, ...) in modo fluido con nuvole di punti ad alta densità ([Sezione 1.1.2]) negli anni sono stati sviluppati degli algoritmi in grado di facilitare la fase di visualizzazione.

#### 3.3.1 Soglia al numero di punti

Una delle tecniche più triviali è quella di imporre una soglia al numero di punti che vengono visualizzati. Oltre a questa soglia il numero di punti rimane costante poiché all'inserimento di un nuovo punto, quest'ultimo viene salvato al posto di uno scelto in modo casuale tra quelli aggiunti precedentemente. Questo consente di mantenere il numero di punti sotto un certo valore massimo. La Figura 3.2 mostra un esempio d'implementazione di questa tecnica.

#### 3.3.2 Culling

Nella visualizzazione 3D il termine Culling descrive la rimozione di oggetti di qualsiasi tipo che non contribuiscono all'immagine finale. Questo consente di far lavorare il motore grafico unicamente sugli oggetti che verranno visualizzati, evitando di sprecare risorse su ciò che non sarà visibile nell'immagine finale. Questa tecnica contribuisce molto nel contesto del progetto ([Capitolo 1]) proprio perchè consente di rimuovere dati non necessari. Per esempio se ci stiamo focalizzando su un dettaglio specifico della nuvola di punti e la maggioranza degli altri punti non vengono visualizzati, tali punti potranno non essere caricati per la visualizzazione e facilitarne quindi la computazione. Esistono vari tipi di culling. In seguito vengono descritti i più rilevanti.

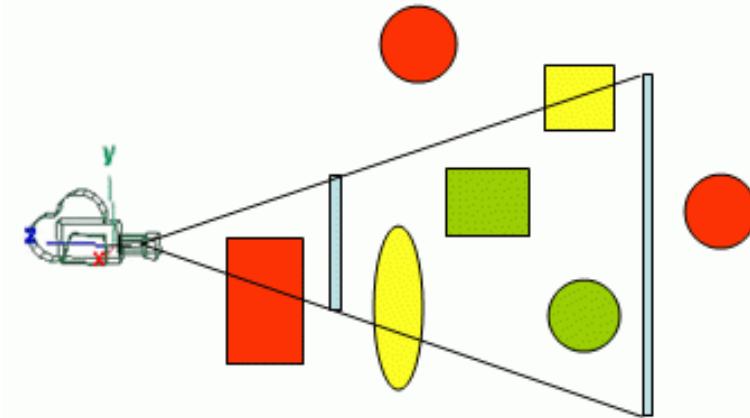


Figura 3.3: Immagine illustrante il funzionamento del frustum culling. Gli oggetti in rosso non vengono visualizzati in quanto fuori dalla piramide di visualizzazione. Perciò è inutile aggiungerli alla lista di oggetti in visualizzazione. (Fonte: [9])

### 3.3.2.1 Frustum Culling

Nella sua forma più semplice verifica che ogni oggetto rappresentante la scena da visualizzare sia contenuto (completamente o parzialmente) nella piramide di visualizzazione<sup>3</sup>. Se l'oggetto analizzato non è contenuto, può essere scartato, facendo risparmiare risorse computazionali. In Figura 3.3 possiamo notare in rosso gli oggetti che non sono contenuti nella piramide di visualizzazione e quindi rimossi dalla scena.

### 3.3.2.2 Backface Culling

In una scena di oggetti 3D composti da una serie di poligoni, le facce di questi poligoni che non sono orientate (parzialmente o direttamente) verso la camera, non sono visibili. La tecnica di Backface Culling prevede quindi di rimuovere queste facce semplificando così gli oggetti che saranno computazionalmente più semplici da visualizzare. Questa procedura può essere visualizzata in Figura 3.4. Ipotizzando che la camera sia di fronte all'oggetto 3D, notiamo a destra come il backface culling andrà a rimuovere la faccia che non è visibile.

### 3.3.3 Level Of Details (LOD)

Con il termine Level of Details (LOD) vengono indicate tutte le tecniche atte a modificare la complessità<sup>4</sup> di un oggetto 3D con lo scopo di facilitarne la visualizzazione. La complessità viene modificata secondo fattori che indicano come modificarla senza perdere qualità grafica. Queste tecniche risultano particolarmente utili nel contesto del progetto di diploma

<sup>3</sup>Piramide tronca formata dai 6 piani che contengono la scena 3D.

<sup>4</sup>Numero di primitive di cui è composto, in questo caso numero di punti.

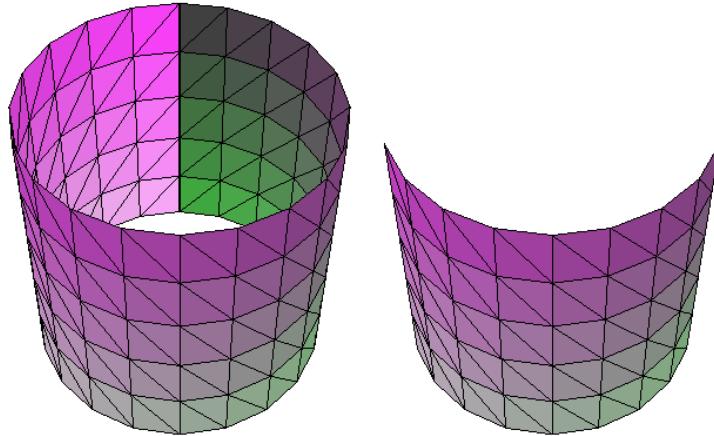


Figura 3.4: Immagine illustrante il funzionamento del backface culling. Possiamo notare lo stesso oggetto: a sinistra senza backface culling, a destra con il culling. La camera si trova di fronte a questo oggetto e secondo la visione della camera l'oggetto non ha subito modifiche. (Fonte: [10])

([Capitolo 1]). Infatti esse consentono di semplificare l'oggetto 3D in visualizzazione e nel caso di nuvole di punti ad alta densità ([Sezione 1.1.2]) questa semplificazione rappresenta un aspetto vitale per favorire delle buone prestazioni in visualizzazione. Di seguito vediamo alcune di queste tecniche che più si prestano ad un'applicazione sulle nuvole di punti. Tutte le informazioni qui riportate sono un riassunto di quanto esposto in [11].

- Octree: albero dove ogni nodo ha al massimo 8 figli. Al raggiungimento del massimo numero di figli, si espande la gerarchia aggiungendo il nodo ad un livello inferiore. Questa struttura dati risulta comoda per partizionare un ambiente tridimensionale come è possibile osservare in Figura 3.5. Relativamente al contesto del progetto di diploma, una struttura ad Octree potrebbe essere utile per convertire una nuvola da un file unico ad una gerarchia di file. Immaginiamo che in ogni nodo dell'octree ci sia una porzione di punti e che tutti i nodi uniti formano la nuvola originale. In questo modo il compito di un visualizzatore sarebbe facilitato, in quanto si potrebbero caricare solo parti della nuvola. Il minor numero di punti da visualizzare comporterebbe una computazione necessaria minore.
- Nested Octree: gerarchia per gestire nuvole di punti ad alta densità che possono essere visualizzate ed esplorate interattivamente. Si tratta di un Octree esterno che descrive il traversamento della gerarchia e un octree interno che viene rappresentato da tanti octree uno per ogni nodo di quello esterno. La profondità massima degli octree interni è limitata (per esempio 8 livelli). Entrambi gli octree suddividono lo spazio regolarmente ad ogni livello. Rispetto ad un normale octree, consente di avere

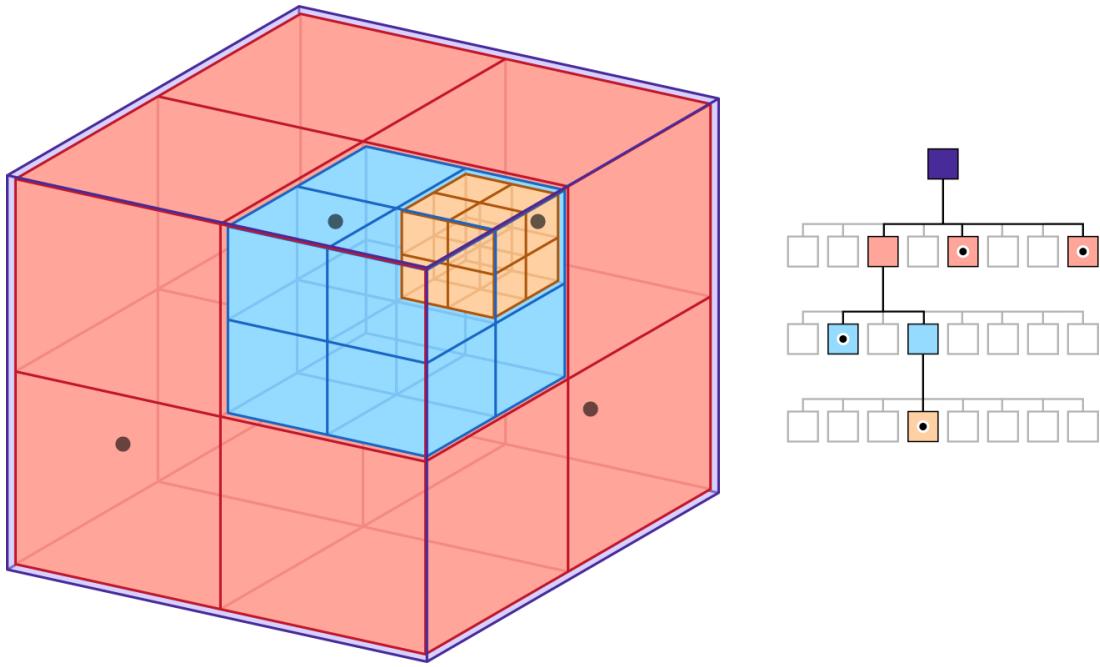


Figura 3.5: Illustrazione di un Octree. A sinistra possiamo notare come sia possibile il partizionamento di un ambiente 3D. Mentre a destra la relativa rappresentazione ad albero gestibile programmaticamente. (Fonte [12])

un maggior numero di LOD, in quanto in ogni nodo della gerarchia sono presenti più livelli di dettaglio interni.

- Layered Point Clouds (LPC): gerarchia che in ogni nodo salva una nuvola di almeno  $M$  punti. Per il nodo radice gli  $M$  punti vengono scelti in modo casuale e rimossi dalla nuvola di input, mentre successivamente la nuvola viene divisa in due parti. Ognuna delle due parti agisce da nuvola input per la continuazione del proprio ramo nella gerarchia. Rispetto ad un Octree o un Nested Octree, LPC consente la creazione di un minor numero di livelli di dettaglio. Questa caratteristica potrebbe fare la differenza se si desidera implementare un visualizzatore performante.
- QSplat: gerarchia che suddivide lo spazio 3D in sfere. L'albero relativo è quindi un albero in cui ogni nodo rappresenta una sfera. I punti vengono salvati nei nodi foglia, un nodo padre possiede due nodi figli e le sfere dei nodi figli sono completamente racchiuse in quella del nodo padre. Il nodo padre salva i valori medi<sup>5</sup> dei valori dei figli. Oltre alla diversa popolazione di ogni livello di dettaglio, come avviene per LPC, rispetto alle tecniche citate in precedenza, in certi livelli di dettaglio QSplat non vi-

<sup>5</sup>Vengono creati nuovi punti ottenuti dalla media aritmetica tra coppie di punti e tra i loro colori.

sualizzerà punti veri e propri della nuvola, ma punti finti ottenuti unendo coppie di punti.



## Capitolo 4

# Approccio al problema

Trattandosi di un progetto con una forte componente di ricerca si è scelto di procedere con sperimentazioni basate su prototipazione iterativa. Dopo un necessario studio dello stato dell'arte è stato possibile iniziare questo processo iterativo che verrà descritto in seguito. Il risultato di questo processo è un prototipo quanto più vicino a ciò che il progetto di diploma intende realizzare. L'esecuzione di tale processo è stata facilitata dall'uso di strumenti tecnologici di supporto.

### 4.1 Studio dello stato dell'arte

Il tipo di progetto ha richiesto un lavoro di documentazione al fine di comprendere chiaramente quali algoritmi vengono utilizzati oggi giorno per la gestione di nuvole di punti ad alta densità ([Sezione 1.1.2]), la loro visualizzazione e l'interazione con esse. A questo scopo sono state consultate le banche dati disponibili<sup>1</sup>, ricercando paper riguardanti il problema, in particolare ACM Digital Library<sup>2</sup> e IEEE/IET Electronic Library<sup>3</sup>. Oltre ad una comprensione dello stato dell'arte sulle nuvole di punti, la lettura di questi documenti ha contribuito all'individuazione delle librerie utilizzabili per i futuri prototipi. Una volta individuate le librerie è stato possibile iniziare la serie di iterazioni previste.

### 4.2 Metodo iterativo

Come precedentemente accennato si è scelto di applicare una prototipazione iterativa. In Figura 4.1 possiamo osservare il diagramma rappresentante questo processo iterativo. Abbiamo come input delle librerie precedentemente individuate e candidate per la valutazione. Mentre come output otteniamo il prototipo finale. Il processo iterativo tra input ed output è

<sup>1</sup><http://www.supsi.ch/biblioteca/banche-dati.html>.

<sup>2</sup><https://dl.acm.org/>.

<sup>3</sup><https://www.ieee.org/publications/subscriptions/iel-overview-pricing.html>.

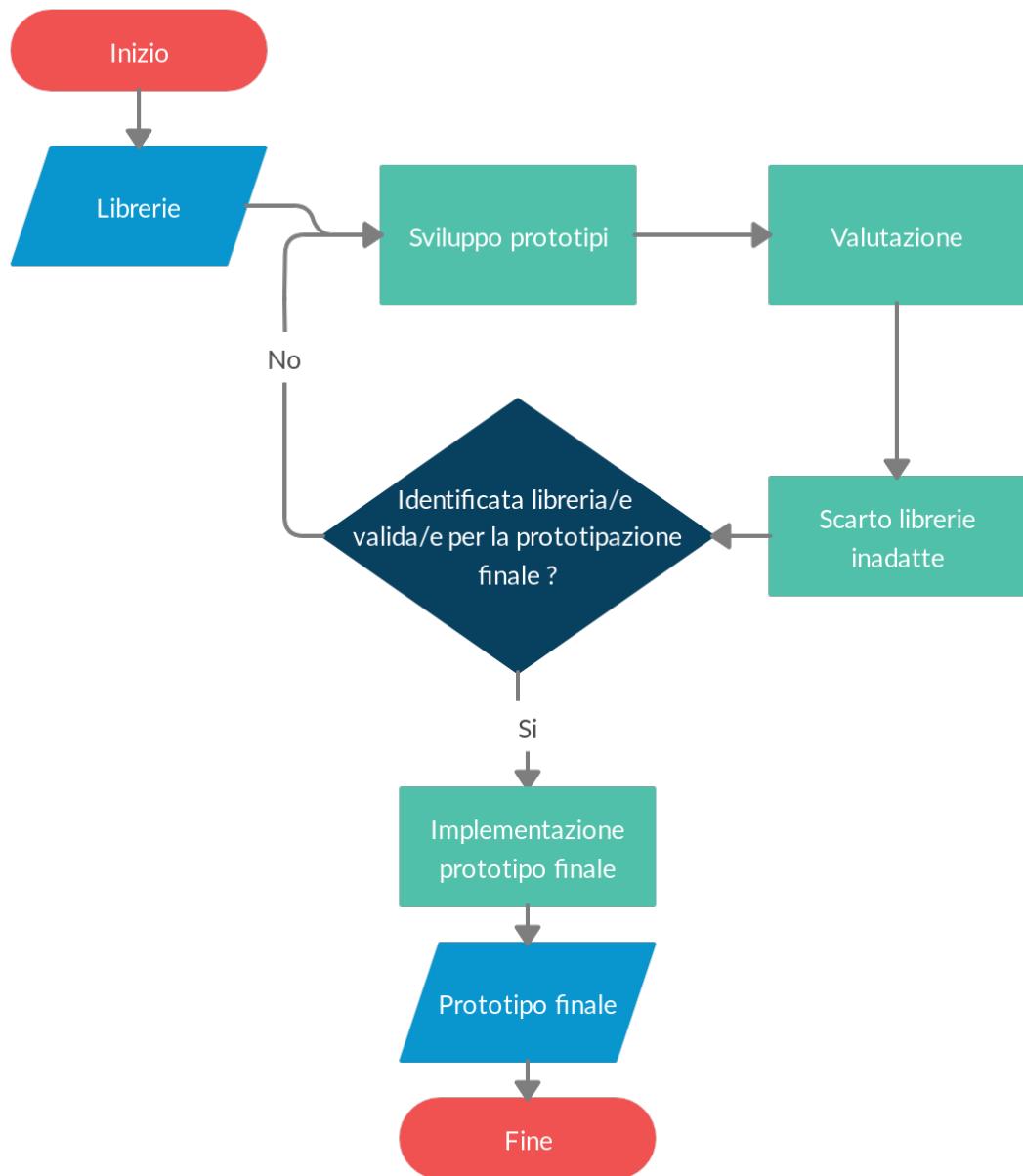


Figura 4.1: Illustrazione del processo iterativo adottato come approccio al problema.

composto da varie fasi che vengono di seguito descritte.

- Creazione prototipi: in questa fase per ogni libreria viene creato un prototipo. Per esempio durante la prima iterazione veniva richiesto semplicemente di visualizzare una nuvola di punti ([Sezione 1.1.1]) e poter interagire con essa. Non riuscire con una determinata libreria a creare un prototipo, significa che la libreria sarà scartata in quanto inadatta. Naturalmente la complessità e i requisiti di ogni prototipo vengono aumentati ad ogni iterazione.
- Valutazione: questa fase prevede la valutazione dei prototipi creati. Può essere fatta in due modi:
  1. semplicemente valutando se i prototipi creati rispettano i requisiti imposti dall'iterazione (Prime iterazioni<sup>4</sup>).
  2. introducendo uno schema di valutazione composto da parametri misurati per ogni prototipo (Iterazioni avanzate<sup>5</sup>). Grazie a questo schema ogni libreria può essere valutata nel dettaglio.
- Scarto librerie inadatte: le librerie che in seguito alle valutazioni sono risultate inadatte, vengono scartate. Successivamente in base al numero di prototipi finali che si desidera realizzare viene scelto se continuare con il processo iterativo di selezione delle librerie, oppure se passare all'implementazione finale.

La definizione del linguaggio di programmazione da utilizzare è stata fatta durante il corso di questo processo, definendo il linguaggio alla fine del processo iterativo. Questo è dovuto al fatto che le varie librerie prese in considerazione supportano linguaggi di programmazione differenti (C++, python, C#).

La fine di questo processo porta quindi all'implementazione del prototipo finale. Questa fase utilizza come base di partenza il prototipo realizzato con la libreria scelta.

In seguito verranno descritti gli strumenti utilizzati che hanno permesso di portare a termine tale processo.

### 4.3 Framework di sviluppo

Al fine di gestire il ciclo di sviluppo del software è stata adottata una metodologia agile<sup>6</sup> nello specifico una versione "light" del framework Scrum. Scrum è un framework di processo di

<sup>4</sup> Iterazioni iniziali in cui i requisiti imposti ai prototipi creati rappresentano funzionalità basilari.

<sup>5</sup> Successivamente alle prime iterazioni vi sono delle iterazioni avanzate. Ossia iterazioni in cui sono introdotte tecniche di valutazione con una struttura complessa e dettagliata.

<sup>6</sup> Metodi di sviluppo del software focalizzati sull'obiettivo di consegnare al cliente in tempi brevi e frequentemente software funzionante e di qualità ([13]).

tipo agile utilizzato dai primi anni novanta per gestire lo sviluppo di prodotti complessi. Esso prevede di dividere un progetto in blocchi rapidi di lavoro detti Sprint ([14]). Per questo progetto di diploma è stato deciso di lavorare con Sprint di 2 settimane. Ad inizio progetto è stato creato il backlog di prodotto con user stories e features che sono state aggiornate man mano con l'evolversi del progetto. Ogni Sprint è stato tenuto sotto controllo grazie a degli incontri periodici settimanali tra relatore e studente.

## 4.4 Strumenti tecnologici

Con lo scopo gestire al meglio il progetto di diploma sono state scelte delle tecnologie di supporto. Questi strumenti hanno consentito un controllo delle versioni dei prototipi creati, e una pianificazione delle iterazioni. Viene premesso che non sono state adottate tecniche per l'automatizzazione del deployment. Questo perchè lavorando a vari prototipi con tecnologie diverse e linguaggi diversi, avrebbe richiesto un investimento di tempo troppo elevato rispetto ai benefici.

### 4.4.1 Piattaforma

Si è scelto TI-EDU Scource Code Management (SCM)<sup>7</sup> come piattaforma per le gestione del progetto di diploma. Questo strumento basato su Redmine<sup>8</sup> fornisce gli strumenti necessari all'utilizzo del framework Scrum ([Sezione 4.3]). È stato quindi possibile preparare e controllare l'andamento degli Sprint ([Figura 4.2]) attraverso SCM. La piattaforma inoltre mette a disposizione una wiki utile per descrivere il progetto. La wiki è stata utilizzata per salvare gli appunti inerenti lo stato dell'arte, annotare il procedimento del processo iterativo e le valutazioni per ogni libreria.

### 4.4.2 Sistema software per il controllo di versione

Per il controllo delle versioni è stato scelto l'utilizzo di git<sup>9</sup>. Sono stati utilizzati due rami: master e dev. Lo sviluppo è stato effettuato su dev. Ciclicamente al raggiungimento di uno stato intermedio di consistenza è stata eseguita l'operazione di merge nel ramo master ([Figura 4.3]).

---

<sup>7</sup>Piattaforma messa a disposizione da SUPSI agli studenti.

<sup>8</sup>Applicazione web open source per la gestione di progetti.

<sup>9</sup>Software di controllo versione distribuito, creato da Linus Torvalds.

**Sprint board**

Sprint: SALA005

Start date: 19/07/0030

End date: 19/08/0013

Filter by assignee: any

Product backlog items	New	In Progress	Resolved	Feedback	Testing
#12812: Individuazione classi					
#12811: Selezione punti con nuvola dinamica					
#12813: Individuazione classi con nuvola dinamica					

Figura 4.2: Interfaccia grafica della piattaforma utilizzata per il controllo degli Sprint.

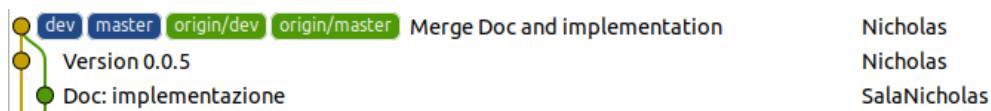


Figura 4.3: Esempio di merge nel ramo master al raggiungimento di uno stato di consistenza estratto da git.



# Capitolo 5

# Implementazione

Successivamente alla definizione del tipo di approccio, è iniziata la sua applicazione vera e propria. L'implementazione del progetto di diploma è divisa principalmente in due parti: implementazione prototipi, implementazione del prototipo finale. La prima parte riguarda l'implementazione di tutti i prototipi ed è relativa alle valutazioni delle librerie. Mentre la seconda parte è l'implementazione del prototipo finale vero e proprio. Questo capitolo descrive entrambe le fasi. Viene quindi spiegata la scelta della libreria, l'esecuzione del processo iterativo adottato ([Sezione 4.2]) e le implementazioni svolte per realizzare il prototipo finale.

## 5.1 Scelta libreria

La fase di scelta delle librerie è stata necessaria per la creazione dei prototipi che hanno consentito di valutare e scartare le librerie individuate secondo gli obiettivi prefissati ([Sezione 2.5]).

### 5.1.1 Individuazione

Grazie ad un precedente studio dello stato dell'arte ([Sezione 4.1]) sono state individuate delle librerie riguardanti il contesto ([Capitolo 1]) trattato. Fondamentali per questa ricerca sono stati i requisiti ([Sezione 2.4]) del progetto di diploma. Vengono in seguito elencate le librerie candidate:

- Open3D[16] 0.7.0.0: libreria multipiattaforma per lo sviluppo di software che si occupa di dati 3D. Open3D è stata sviluppata con un set di dipendenze minimo. Il backend è stato sviluppato in C++ e viene esposto anche attraverso un'interfaccia frontend python. Il modulo Geometry possiede una classe PointCloud che astrae una nuvola di punti attraverso una lista di punti, ognuno con coordinate e colori. Mentre le classi Octree e KDTree consentono ricerche e suddivisioni con algoritmi ottimizzati.

Repository: <https://github.com/intel-isl/Open3D>

Licenza: MIT

- Potree[17] 1.6: Libreria basata su webGL<sup>1</sup> per la visualizzazione di nuvole di punti ad alta densità ([Sezione 1.1.2]) su browser sviluppata presso l'University of Technology di Vienna<sup>2</sup>. Per la sua esecuzione necessita solamente di un browser con supporto a webGL e di un webserver. Caratteristica fondamentale di questa libreria è il formato con il quale vengono salvate le nuvole di punti. Questo formato prevede una gerarchia di file basata su Octree ([Sezione 3.3.3]) e consente di poter visualizzare nuvole di punti giganti evitando di caricare contemporaneamente tutti i punti in memoria.  
 Repository: <https://github.com/potree/potree>  
 Licenza: BSD/MIT
- BA\_Pointcloud[18] 1.2: Tesi di bachelor sviluppata da Simon Maximilian Fraiss presso l'University of Technology di Vienna. Software basato sul formato utilizzato da Potree per la gestione di nuvole di punti ad alta densità e sul motore grafico Unity<sup>3</sup> per la loro visualizzazione. Questa combinazione consente di avere alte prestazioni anche con nuvole molto dense.  
 Repository: [https://github.com/SFraissTU/BA\\_PointCloud](https://github.com/SFraissTU/BA_PointCloud)  
 Licenza: BSD 2-Clause
- VTK[19] 8.2.0: Libreria multipiattaforma nata nel 1994 e in sviluppo tutt'oggi. Rende disponibili funzionalità come: image processing, grafica 3D, visualizzazione di volumi. Attualmente alla versione 8.2.0.  
 Repository: <https://gitlab.kitware.com/vtk/vtk>  
 Licenza: OSI-approved BSD 3-clause
- PCL[20] 1.9.1: Libreria multipiattaforma sulla quale si è basata la creazione di Open3D. Attualmente PCL si trova alla versione 1.9.1. Tra le sue dipendenze è presente VTK, utilizzata per la visualizzazione di punti 3D.  
 Repository: <https://github.com/PointCloudLibrary/pcl>  
 Licenza: BSD
- CGAL[21] 4.13.1: Libreria che mette a disposizione efficienti algoritmi geometrici. Essa è usata in varie aree che necessitano di computazioni geometriche quali ad esempio: biologia molecolare, medicina e robotica. Riguardo alle nuvole di punti possiede funzionalità come: classificazione ([Sezione 1.1.3]), ricostruzione di superfici e stima delle distanze.  
 Repository: <https://github.com/CGAL/cgal>  
 Licenza: LGPL/GPL

<sup>1</sup>Libreria che fornisce attraverso un contesto html un'API di grafica 3D per browser.

<sup>2</sup><https://www.tuwien.at/en/>.

<sup>3</sup><https://unity.com/>.

- 3DTK 1.2: Libreria multipiattaforma con algoritmi e metodi per il processamento di nuvole di punti. Supporta vari formati di nuvole di punti e possiede un veloce visualizzatore.

Repository: <https://sourceforge.net/p/slam6d/code/HEAD/tree/trunk/>

Licenza: GPLv3

- PPTK 0.1.0: Pacchetto python per la visualizzazione e il processamento di nuvole di punti. Come meccanismo di dettaglio utilizza un Octree mentre i punti sono salvati tramite un KDTree.

Repository: <https://github.com/heremaps/pptk>

Licenza: MIT

### 5.1.2 Prima iterazione

Seguendo l'approccio descritto nella Sezione 4.2, successivamente all'individuazione delle librerie, è avvenuta la prima iterazione. Questo ha comportato la creazione dei primi prototipi e la valutazione delle librerie secondo quanto creato. Due requisiti hanno fatto la differenza:

1. presenza di api per la programmazione: senza delle api non sarebbe stato possibile utilizzare la libreria importandola in un altro programma.
2. visualizzare una nuvola di punti e interagire con essa.

Il fallimento o il successo nella creazione di questo primo prototipo è stato determinante. A seguito di quanto si è riuscito a realizzare e delle valutazioni fatte, è stato deciso di scartare le seguenti librerie:

- CGAL: Durante il tentativo di costruire un visualizzatore, si è notato che CGAL non presenta un modulo di visualizzazione ma solamente algoritmi per il processamento di nuvole.
- 3DTK: Dopo aver risolto vari errori in fase di compilazione, grazie anche all'aiuto tramite email degli sviluppatori ([Appendice B]), si è notato che 3DTK prevede solamente un utilizzo tramite linea di comando, non esistono api per la programmazione.
- PPTK: Dopo aver installato il pacchetto python e seguito le istruzioni sulla repository per la creazione di un visualizzatore, si è notato che il codice presenta un errore in esecuzione. Sono stati fatti vari tentativi per risolvere questo problema, ognuno senza successo. Sembrerebbe che la libreria si blocchi durante l'apertura di un socket con un altro processo. In Figura 5.1 è riportato il codice utilizzato che presenta questo problema.

```

1 import numpy as np
2 import pptk
3
4 v = pptk.viewer(x)
5 v.set(point_size=0.01)
6

```

Figura 5.1: Codice python utilizzato per la creazione del primo prototipo con PPTK. (Fonte [22])

### 5.1.3 Seconda iterazione

Ogni libreria che ha superato la prima selezione è stata sottoposta ad ulteriori sperimentazioni, come previsto dall'approccio iterativo. Viene introdotto in questa iterazione uno schema di valutazione basato sull'uso di metriche e parametri misurati oggettivamente. Come descritto nel wiki del Khronos Group<sup>4</sup> il tentativo di valutare un'applicazione OpenGL interattiva richiede una metrica stabilità per ottenere le prestazioni dell'applicazione. È una pratica comune misurare i Frame Per Second (FPS)<sup>5</sup> e questo valore viene considerato come velocità di render da molti nuovi programmati. Essendo questo valore non lineare nel tempo, tuttavia non sempre è il miglior modo di misurare le prestazioni. Il Frame-Time (inverso degli fps) rappresenta un'alternativa valida. Esso indica il tempo in cui ogni frame appare sullo schermo: più il frame-time è piccolo più la fase di visualizzazione è veloce e di conseguenza si ottiene un applicativo fluido. Per poter valutare in modo corretto un'applicazione OpenGL è necessario quindi scegliere i parametri adatti.

#### 5.1.3.1 Parametri misurati

Sono stati scelti i seguenti parametri di valutazione per ogni libreria:

- tempo di caricamento del file comprese eventuali conversioni<sup>6</sup>
- tempo di caricamento del file senza conversione
- fps durante lo zoom
- fps durante il movimento (con mouse) della nuvola
- frame-time durante lo zoom
- frame-time durante il movimento (con mouse) della nuvola
- quantità di ram utilizzata dal processo
- percentuale di utilizzo della cpu durante lo zoom
- tempo di selezione dei punti scelti (tempo tra click e selezione effettiva)
- meccanismo di caricamento della nuvola
- interazione programmatica

<sup>4</sup><https://www.khronos.org/opengl/wiki/Performance>.

<sup>5</sup>Numeri di fotogrammi prodotti da un'applicativo in un secondo.

<sup>6</sup>Conversioni del file di input necessarie per adattarlo alla struttura supportata dalla libreria in valutazione.

### 5.1.3.2 Nuvole di punti utilizzate per le prove

Con lo scopo di meglio comprendere le prestazioni delle librerie, le prove si sono basate su nuvole di dimensioni diverse (piccola, media, grande):

- whole.xyz, 602.5 MB, 13'173'867 punti
- birdfountain\_station1\_xyz\_intensity\_rgb.txt, 1.5 GB, 40'133'912 punti
- castleblatten\_station5\_xyz\_intensity\_rgb.txt, 2.8 GB, 49'152'311 punti

Si è scelto l'uso di tre file diversi perchè si desidera valutare la variazione delle prestazioni al variare della dimensione della nuvola. Nuvola media e grande utilizzate per le prove sono messe a disposizione dall'ETH di Zurigo [23]. Sono state scelte queste nuvole dato che vengono appositamente distribuite per benchmark. Vista la loro densità esse rappresentano un ottimo campione rappresentativo per le misurazioni delle prestazioni in esame in questo lavoro.

### 5.1.3.3 Computer utilizzati

Di seguito vengono descritte le specifiche dei computer usati per le prove:

- HP Envy 17  
Intel Core i7-4700MQ  
NVIDIA GeForce GT740M  
15,6 GiB ram  
Ubuntu 18.04.2 LTS
- Lenovo thinkpad t470s  
Intel Core i7-7600U  
Intel HD Graphics 620 (Kaby Lake GT2)  
19.5 GiB ram  
Fedora 30

È stato deciso di utilizzare due portatili differenti. Infatti il primo è dotato di una scheda grafica dedicata e più performante rispetto al secondo. L'utilizzo di questi due pc ha portato a capire quanto una scheda grafica potesse influire sulle prestazioni delle librerie. Com'è possibile leggere nell'Appendice D.2 dopo aver valutato il file medio si è capito che le prestazioni erano simili in entrambi i PC. È stato quindi deciso di proseguire le misurazioni utilizzando solo il primo computer.



Figura 5.2: Misura dei parametri attraverso gli strumenti riportati in Sezione 5.1.3.4.

#### 5.1.3.4 Strumenti di supporto

Come è possibile notare in Figura 5.2, al fine di misurare i parametri elencati è stato necessario l'utilizzo di strumenti provenienti da terze parti:

- glxosd<sup>7</sup>: misurazione degli fps.
- fps extension<sup>8</sup>: misurazione degli fps per librerie browser-based.
- stacer<sup>9</sup>: misurazione utilizzo della memoria e cpu.

#### 5.1.3.5 Risultati seconda iterazione

I risultati ottenuti in questa seconda iterazione consentono di avere un quadro generale sulle prestazioni di ogni libreria. Per tutte le misurazioni di dettaglio si invita il lettore a consultare le appendici ([Appendice D]). Di seguito viene presentato un riassunto, con i parametri più significativi, delle prove e delle misurazioni svolte.

1. Tempo di caricamento: In Figura 5.3 viene evidenziato come le librerie più veloci nel caricamento delle nuvole siano Potree ([Sezione 5.1.1]) e BA\_Pointcloud ([Sezione 5.1.1]).

<sup>7</sup><https://glxosd.nickguletskii.com/>.

<sup>8</sup><https://chrome.google.com/webstore/detail/fps-extension/gdkkmimldhefhmmmlalioafomdlahcog>.

<sup>9</sup><https://github.com/oguzhaninan/Stacer>.

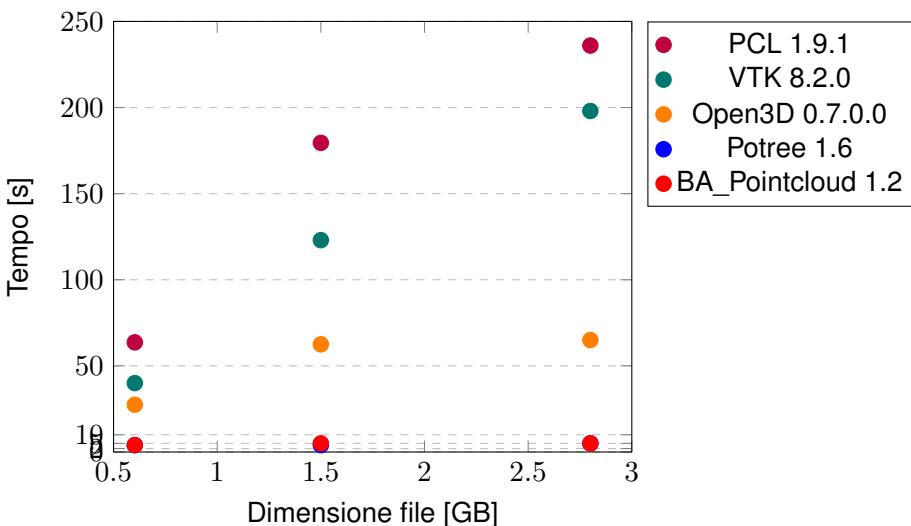


Figura 5.3: Tempo caricamento nuvola senza conversione.

Il risultato è facilmente spiegabile: esse sono le uniche che utilizzano dei meccanismi di LOD ([Sezione 3.3.3]). Esse infatti inizialmente, invece di caricare tutti i punti della nuvola, caricano solamente il nodo radice di una gerarchia basata su un Octree ([Sezione 3.3.3]).

2. Frame-time: Nella Figura 5.4 osserviamo che il frame-time in generale tende a crescere con il crescere della dimensione della nuvola. Infatti all'aumentare del numero di punti da visualizzare, aumenta la computazione necessaria per visualizzarli. È interessante notare come VTK ([Sezione 5.1.1]) mantenga un frame-time costante tra nuvola media e grande. Questo perchè la libreria implementa il meccanismo della soglia al numero di punti, come descritto nella Sezione 3.3.1. Le librerie con un frame-time minore, e quindi con una maggior fluidità nella visualizzazione, sono quelle che utilizzano dei meccanismi di LOD (Potree, BA\_Pointcloud). Particolare è anche il comportamento di PCL. Essa infatti durante lo zoom o il movimento, se la nuvola è molto densa, tenta di diminuire il numero di punti visualizzati per qualche istante. Questo comportamento porta ad avere buone prestazioni durante le interazioni con la nuvola.
  3. RAM utilizzata: In Figura 5.5 possiamo osservare che tra le librerie che finora hanno ottenuto prestazioni simili, vi è una differenza sull'utilizzo della memoria. Infatti BA\_Pointcloud ottimizza l'uso della memoria rispetto a Potree. Questo è probabilmente dovuto ad una differente implementazione dei meccanismi di LOD. Si può notare come, per i casi misurati, Open3D abbia un utilizzo della memoria proporzionale

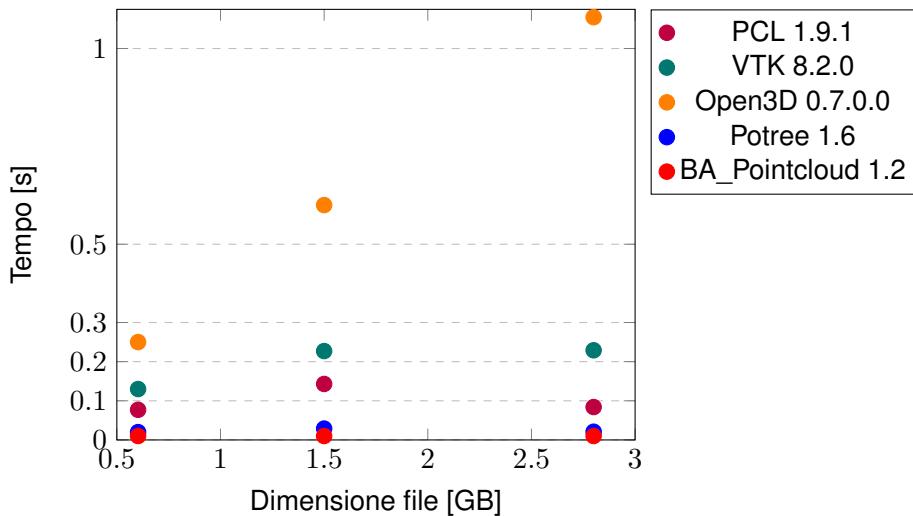


Figura 5.4: Frame-Time durante lo zoom.

alla dimensione della nuvola. Mentre PCL è la libreria che utilizza il maggior quantitativo di RAM.

### 5.1.3.6 Seconda selezione

I risultati elencati hanno permesso di dare delle valutazioni e quindi di passare alla fase di eliminazione delle librerie inadatte. Le seguenti librerie sono state scartate:

- PCL: Uno dei difetti riscontrati in PCL, è la necessità di convertire i file. Infatti essa necessita di un header nel file della nuvola. Anche i colori vanno convertiti: PCL è in grado di leggere solamente un valore `uint32_t`<sup>10</sup> quale informazione del colore. In Figura 5.6 è possibile osservare un tentativo fatto con colori non convertiti. Dover convertire tutti i file delle nuvole rappresenta quindi un punto a sfavore per PCL. Altri importanti motivi che hanno portato a scartare PCL sono l'utilizzo della memoria non ottimizzato ([Figura 5.5]) e i tempi di caricamento elevati ([Figura 5.3]).
- VTK: Nativamente la libreria non possiede un modulo specifico per la gestione di nuvole di punti. Ne è stato realizzato uno per le prove ma non può competere con implementazioni come quella di Open3D. Essendo quindi Open3D più specializzata nel contesto di questo progetto di diploma ([Capitolo 1]), VTK è stata scartata. Open3D potrebbe essere considerata l'evoluzione specializzata sulle nuvole di punti di VTK. Infatti Open3D è stata inizialmente inspirata da PCL, che utilizza VTK.

<sup>10</sup>[http://docs.pointclouds.org/1.7.0/structpcl\\_1\\_1\\_point\\_x\\_y\\_z\\_r\\_g\\_b.html](http://docs.pointclouds.org/1.7.0/structpcl_1_1_point_x_y_z_r_g_b.html).

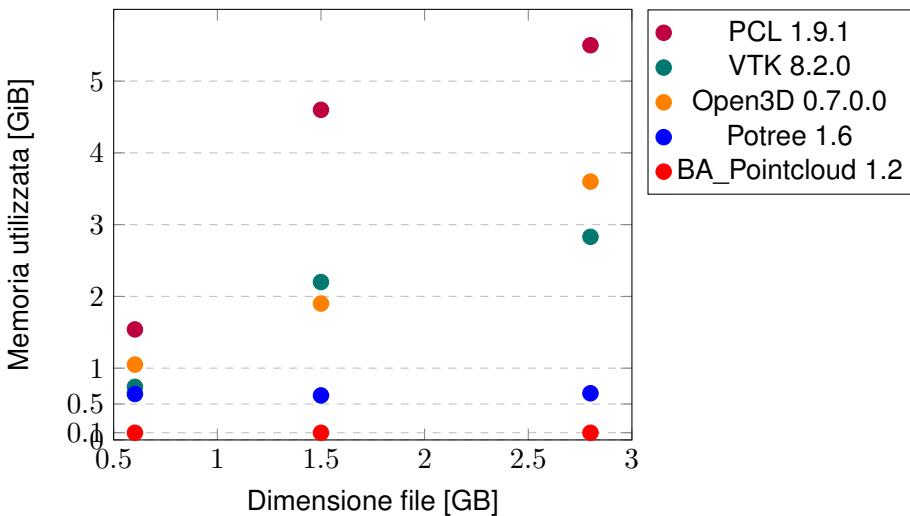


Figura 5.5: Ram utilizzata dopo il caricamento del file.

- **Potree:** Durante le prove è stato notato che Potree, nel momento in cui il LOD ([Sezione 3.3.3]) diminuisce, non rilascia la memoria precedentemente allocata: dopo aver navigato all'interno nella nuvola e caricato in dettaglio vari punti della stessa ([Figura 5.8]), lo zoom viene fatto tornare allo stato iniziale ([Figura 5.9]) ma la memoria occupata non diminuisce. Teoricamente invece Potree dovrebbe scaricare dalla memoria tutti i punti riferiti a LOD non più necessari e tornare ad un utilizzo della memoria simile a quello di partenza ([Figura 5.7]). Questo comportamento potrebbe avere conseguenze spievoli soprattutto nella visualizzazione di nuvole molto dense. Un altro fattore che ha portato allo scarto di Potree è la differenza di prestazioni in confronto a BA\_Pointcloud. Dai risultati ottenuti infatti si può notare come BA\_Pointcloud abbia un frame-time migliore ([Figura 5.4]) e una gestione della memoria più ottimizzata ([Figura 5.5]).

#### 5.1.4 Terza iterazione

A seguito delle scelte fatte durante la seconda iterazione la scelta della libreria sulla quale basare il prototipo finale è stata ristretta a due possibilità: Open3D ([Sezione 5.1.1]) e BA\_Pointcloud ([Sezione 5.1.1]). Coerentemente al processo iterativo scelto come approccio, è stata fatta un'ulteriore iterazione. In particolare per questa fase bisognava capire se ciò che mancava alle due librerie per il raggiungimento degli obiettivi del progetto ([Sezione 2.5]) fosse fattibile. È iniziata quindi una terza fase di creazione prototipi.

##### 5.1.4.1 Open3D

Le sperimentazioni su Open3D si sono orientate principalmente al capire se fosse possibile collegare al visualizzatore un meccanismo di LOD ([Sezione 3.3.3]), così da migliorare le

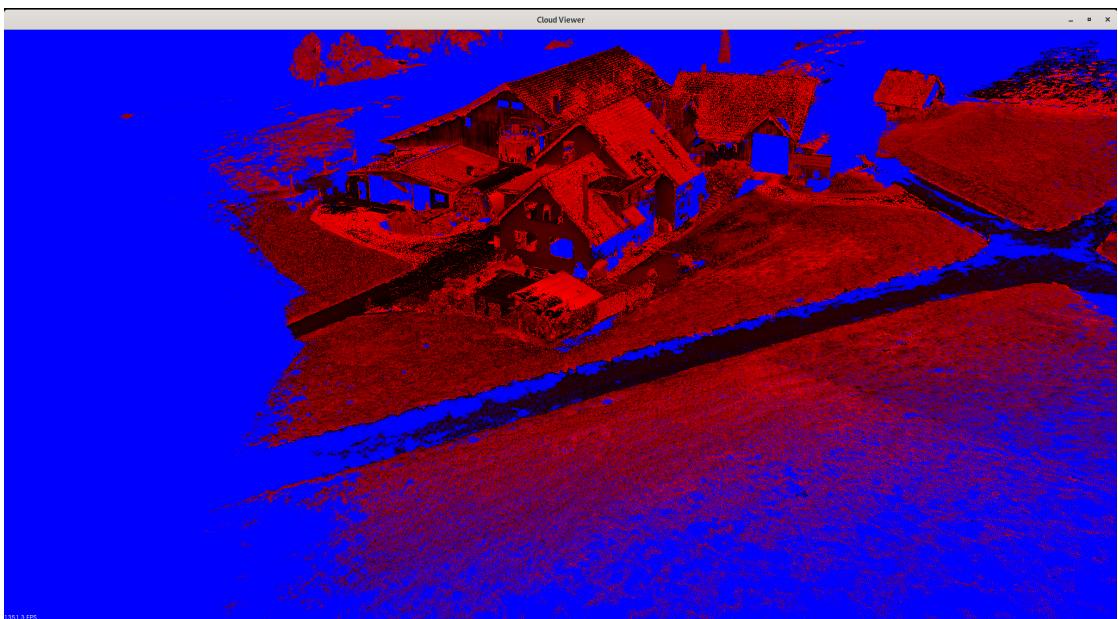


Figura 5.6: Tentativo di apertura di una nuvola con PCL senza convertire i colori nel formato richiesto dalla libreria.

prestazioni anche con nuvole di punti ad alta densità ([Sezione 1.1.2]). Questo ha comportato (come si può notare in Figura 5.10) un tentativo andato a buon fine di creare una nuvola dinamica. Per nuvola dinamica si intende una nuvola i cui punti vengono modificati in fase di esecuzione dell'applicativo. Il successo di questo tentativo ha dimostrato la possibilità di collegare ad open3D un meccanismo di LOD. Si rende attento il lettore che per ottenere questo risultato è stato necessario apportare modifiche e aggiunte al codice sorgente di Open3D.

#### 5.1.4.2 BA\_Pointcloud

Attraverso le sperimentazioni su BA\_Pointcloud si è cercato di capire se fosse possibile l'operazione di selezione di un punto della nuvola. È stato possibile registrare l'operazione di click su di un gameobject<sup>11</sup> ma non su un singolo punto. Un tentativo invece andato a buon fine, come si può notare in Figura 5.11, è stato quello di avere contemporaneamente due visualizzazioni differenti della stessa nuvola.

#### 5.1.5 Scelta conclusiva

Le sperimentazioni finali su Open3D e BA\_Pointcloud hanno portato a meglio comprendere quali parti mancassero ad ogni libreria per il raggiungimento dell'obiettivo del progetto

---

<sup>11</sup>Oggetto fondamentale in Unity usato come contenitore per componenti. BA\_Pointcloud lo utilizza come contenitore per punti.

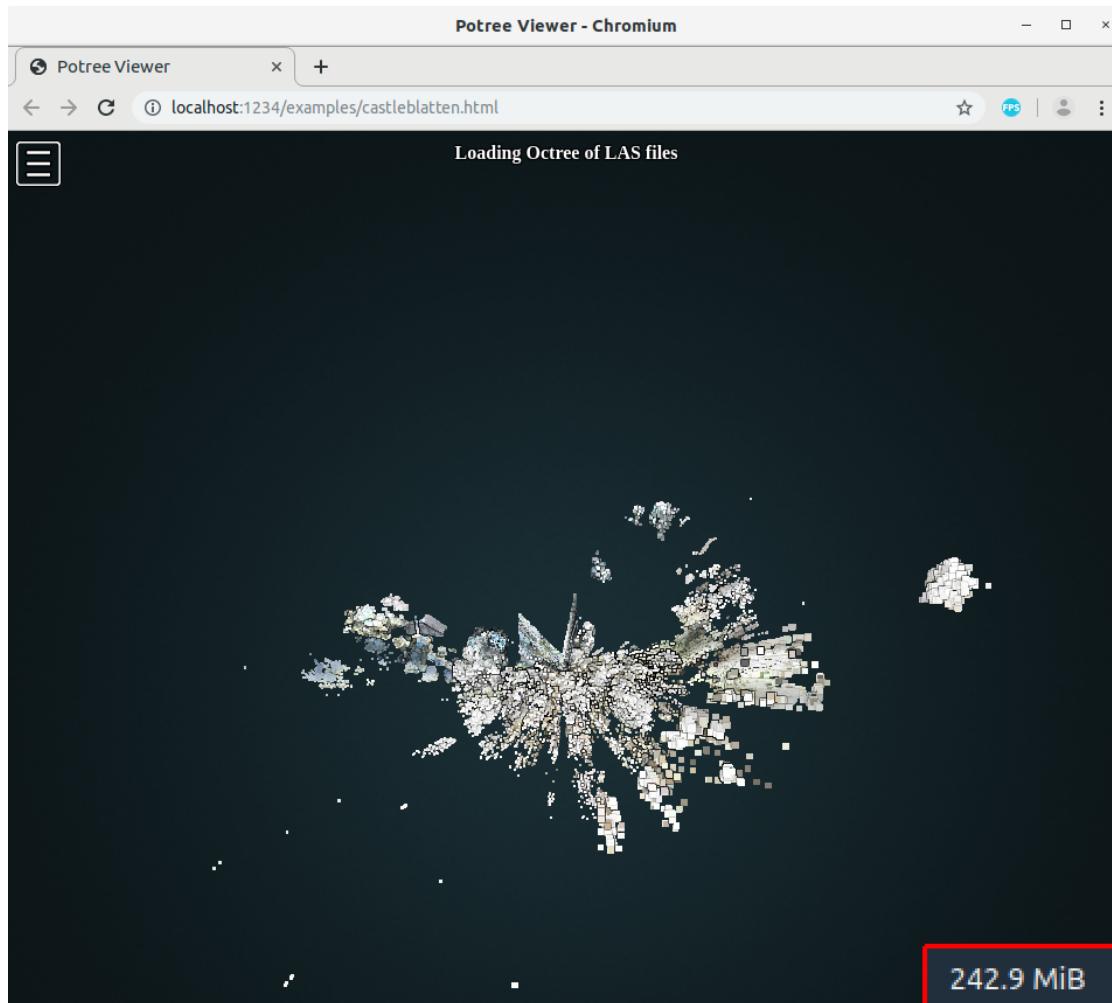


Figura 5.7: Potree: situazione iniziale, memoria a 242.9 MiB.

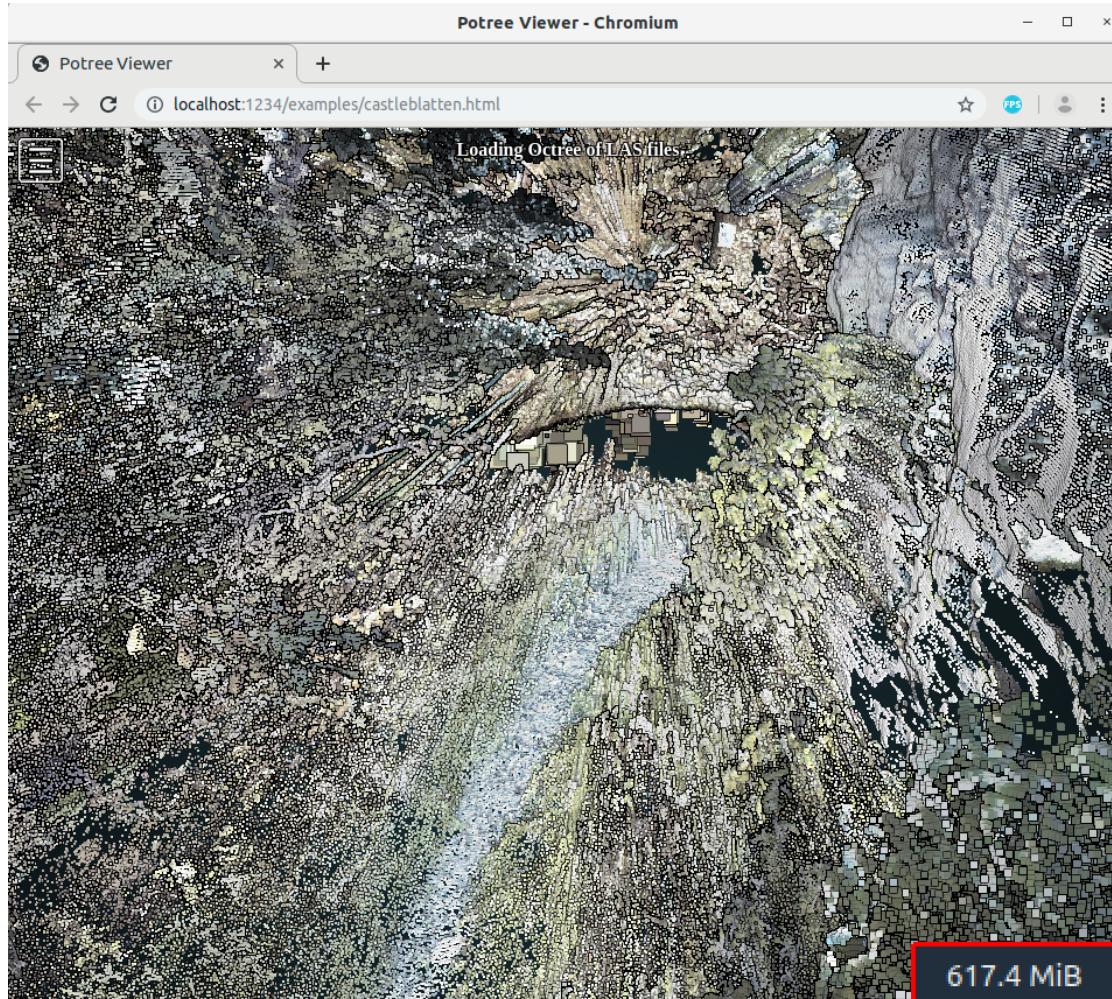


Figura 5.8: Potree: durante la navigazione nella nuvola, memoria a 617.4 MiB.

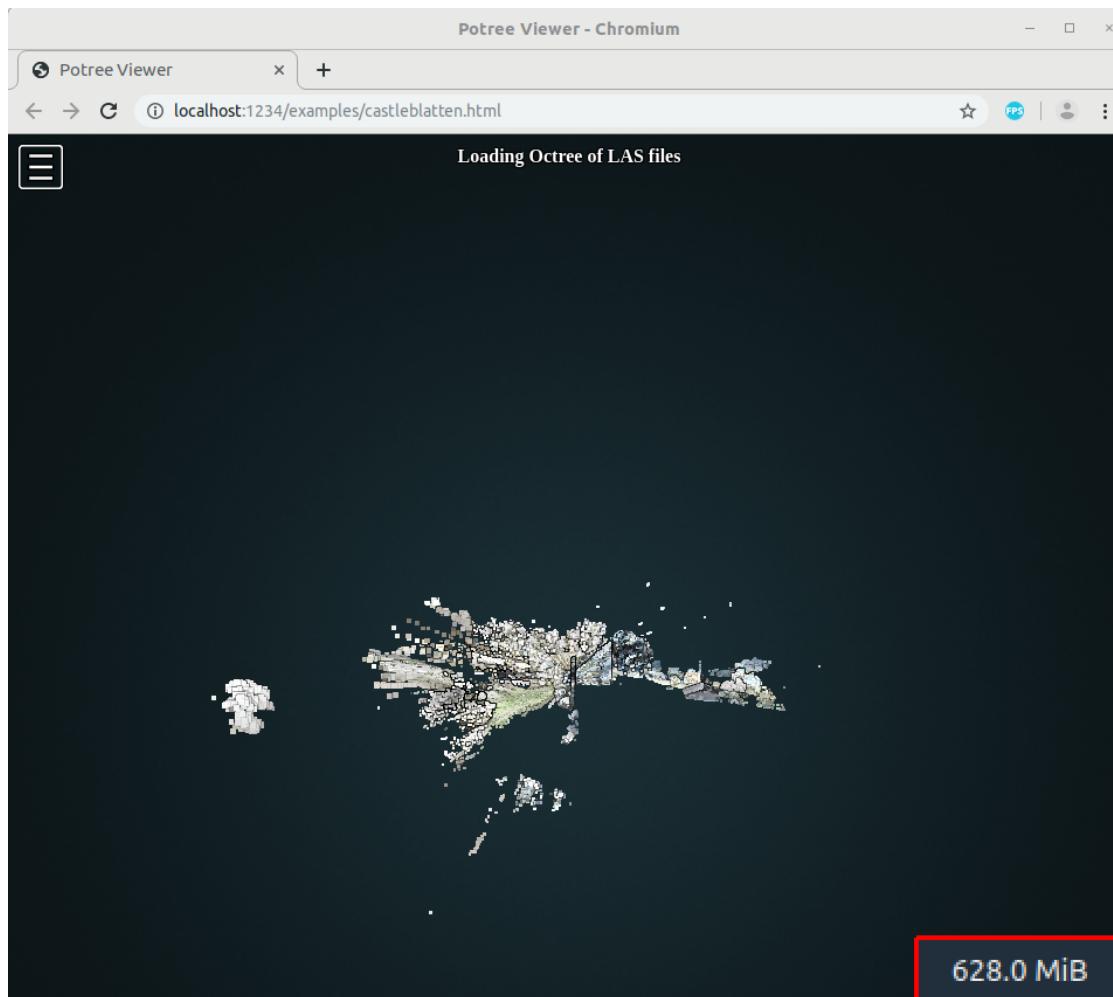


Figura 5.9: Potree: dopo la navigazione con lo stesso LOD di partenza, memoria a 628 MiB.

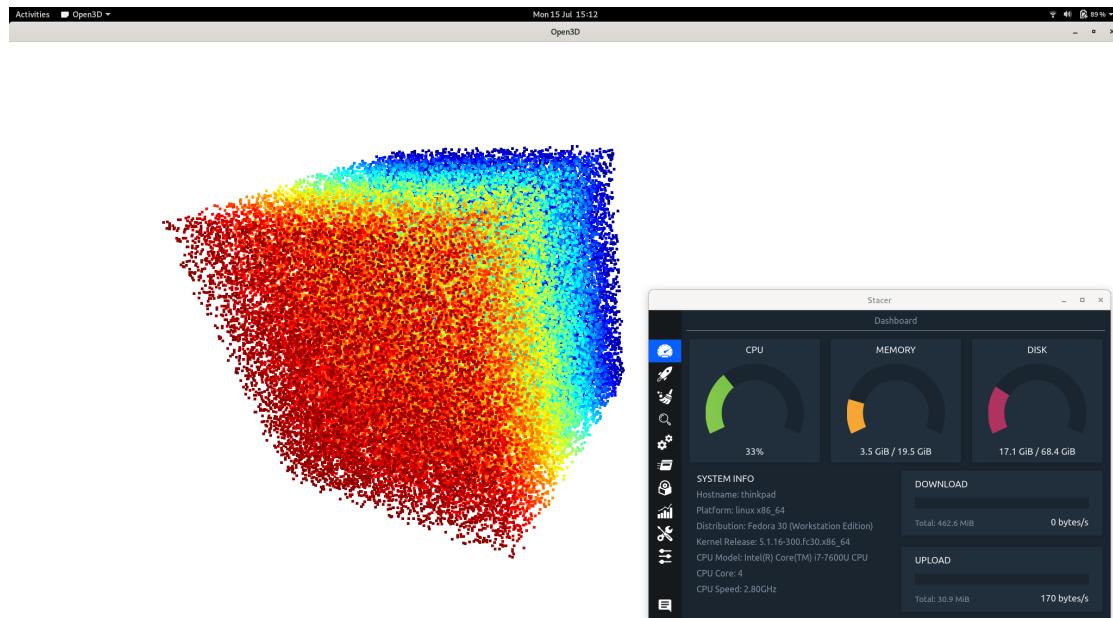


Figura 5.10: Open3D: tentativo di creazione di una nuvola dinamica.

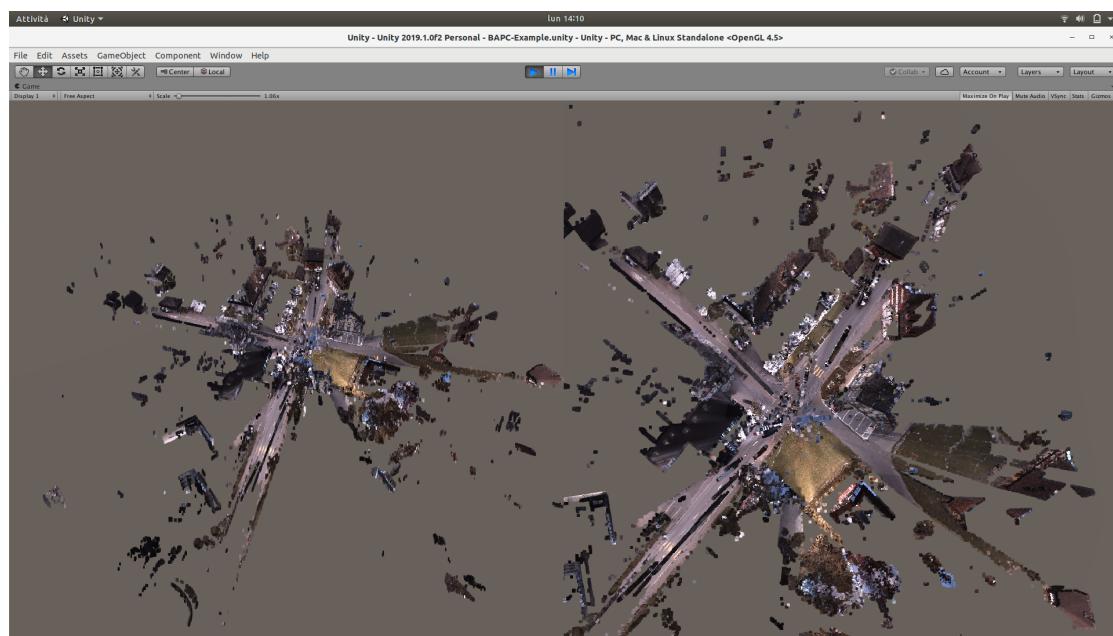


Figura 5.11: BA\_Pointcloud: tentativo di visualizzazione in contemporanea di due nuvole.

([Sezione 2.5]). La scelta finale della libreria non è stata semplice in quanto entrambe presentano pro e contro che vengono elencati di seguito.

1. Open3D:

- ✓ Interazioni ad alto livello con nuvola: selezione di un punto mediante click con il mouse, selezione di aree ..
- ✓ API python ricche di funzionalità
- ✓ Documentazione ampia e ricca di tutorial
- ✓ Codice C++ ben strutturato e provvisto di test
- ✗ Mancanza di un meccanismo di LOD
- ✗ Mancanza di un visualizzatore in grado di caricare una nuvola da una gerarchia basata su Octree (Sezione 3.3.3)

2. BA\_Pointcloud:

- ✓ Ottime prestazioni grazie al meccanismo di LOD (nativo)
- ✓ Ampie funzionalità offerte da Unity
- ✗ Difficoltà d'implementazione della selezione di un singolo punto della nuvola
- ✗ Codice C# non provvisto di test
- ✗ Stretta dipendenza dal formato utilizzato da Potree ([Sezione 5.1.1]) e quindi dal convertitore associato (PotreeConverter<sup>12</sup>).

Valutando questi aspetti la libreria scelta è stata **Open3D**.

Una funzionalità che ha fatto la differenza nella decisione è stata la possibilità di selezionare un punto attraverso il click del mouse: quest'operazione risulta infatti fondamentale quando si desidera conoscere la classe di appartenenza di un punto.

Un altro punto decisivo che ha portato a prendere questa decisione è il voler implementare un applicativo indipendente da convertitori esterni. Infatti BA\_Pointcloud, utilizzando il formato usato da Potree per la lettura delle nuvole, dipende dal convertitore PotreeConverter. Questo convertitore serve per convertire i file delle nuvole in una cartella contenente la gerarchia di file necessaria per il meccanismo di LOD. Implementare invece un convertitore nuovo consente di racchiudere nello stesso applicativo sia convertitore che visualizzatore. Inoltre un convertitore costruito ad hoc consente di creare il formato di salvataggio più adatto al contesto: convertire la nuvola includendo le classificazioni trovate e se necessario anche altre meta-informationi.

<sup>12</sup><https://github.com/potree/PotreeConverter>.

Dato che la parte mancante ad Open3D è un meccanismo di LOD, conoscere e provare BA\_Pointcloud è comunque servito in quanto è stato possibile prendere spunto dall'implementazione descritta nel paper associato [18] per implementare il suddetto meccanismo in Open3D.

Successivamente alla scelta della libreria è quindi iniziata l'implementazione del prototipo finale.

## 5.2 Implementazione prototipo finale

Punto di partenza per l'implementazione del prototipo finale è stato quello sviluppato con Open3D nella fase precedente. Partendo da questa base, è iniziata la creazione di tutta la parte software necessaria al raggiungimento degli obiettivi del progetto di diploma ([Sezione 2.5]). Ci si è concentrati sulla creazione di un convertitore che facilitasse la visualizzazione, un visualizzatore con meccanismo di LOD ([Sezione 3.3.3]) e di un metodo per mostrare la classe associata ad un determinato punto scelto tramite l'operazione di click.

### 5.2.1 Convertitore

Per poter costruire un meccanismo di LOD ([Sezione 3.3.3]) è stata inanzitutto necessaria la creazione di un convertitore che convertisse la nuvola di punti da un file unico ad una gerarchia di file che facilitasse tale meccanismo. La struttura scelta sulla quale basare la gerarchia è stata l'Octree ([Sezione 3.3.3]). Questo particolare tipo di albero in cui un nodo<sup>13</sup> possiede al massimo 8 figli, consente di dividere un ambiente tridimensionale in più livelli. Ogni livello della gerarchia rappresenta la nuvola di punti ad un determinato livello di dettaglio (LOD) ([Sezione 3.3.3]).

Così facendo si può scegliere il livello della gerarchia da caricare in base al valore dello zoom che l'utente sta utilizzando. Se per esempio l'utente sta visualizzando tutta la nuvola, si potrebbe caricare solo il livello associato alla radice dell'albero. Invece se ci si sta focalizzando su un particolare, bisognerà caricare i punti di quel particolare fino ad un livello di dettaglio adatto. In questo modo il numero di punti caricati sarà ottimizzato, e di conseguenza la computazione necessaria a visualizzarli sarà notevolmente ridotta rispetto al caricamento di tutti i punti della nuvola.

La Figura 5.12 illustra questa conversione. La gerarchia creata è quindi un altro modo di rappresentare la stessa nuvola. Ogni file della gerarchia rappresenta un nodo dell'albero e possiede una determinata struttura.

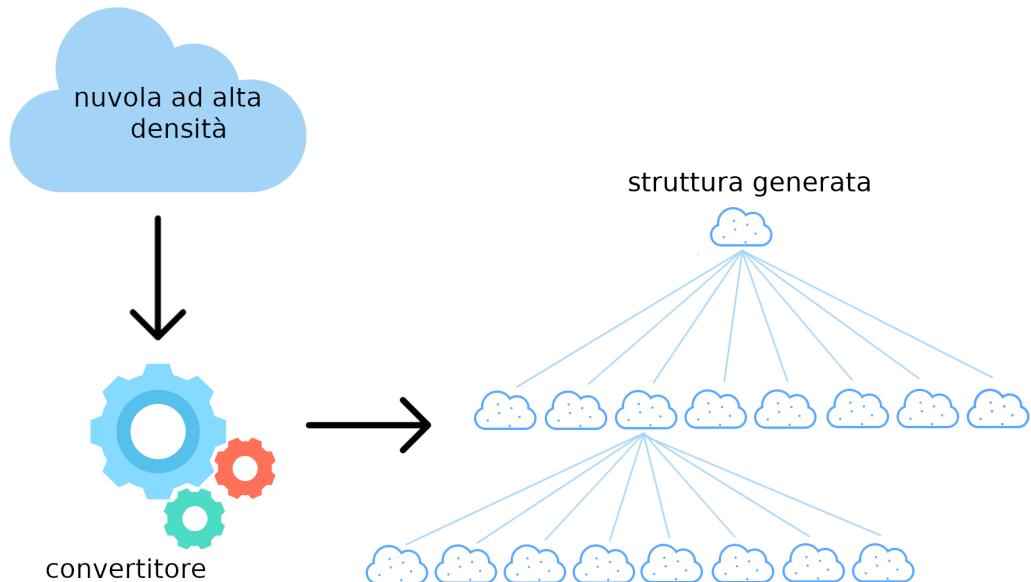


Figura 5.12: La nuvola di punti ad alta densità viene convertita attraverso un convertitore. La gerarchia generata risulta adatta all'implementazione di un meccanismo di LOD ([Sezione 3.3.3]).

```
1 import OctreeFormatTools as oft  
2  
3 gen = oft.Generator("marketplacefeldkirch.txt", "xyzirgb", classes)  
4 gen.parse()  
5
```

Figura 5.13: Codice python per la creazione di un generatore e il suo avvio. Parametri: file, struttura, dizionario classi (opzionale).

### 5.2.1.1 Struttura dei nodi

La struttura dei nodi dev'essere in grado di mantenere tutte le informazioni presenti nel file di input. Questa funzionalità risulta particolarmente impegnativa in quanto il convertitore supporta file testuali con diverse strutture:

1. XYZ: coordinate spaziali.
2. XYZRGB: coordinate spaziali, colore.
3. XYZIRGB: coordinate spaziali, intensità, colore.
4. XYZC: XYZ con classificazione.
5. XYZRGBC: XYZRGB con classificazione.
6. XYZIRGBC: XYZIRGB con classificazione.

È possibile specificare il tipo di struttura come parametro nel costruttore del generatore ([Figura 5.13]). I nodi della gerarchia generati dovranno essere in grado quindi di adattarsi ai dati provenienti dall'input. Per questo è stato scelto di utilizzare un array che viene inizializzato con dimensioni diverse in base ai dati da gestire. Per mantenere le informazioni sulla presenza o meno dei nodi figli di un nodo, si è scelto di usare un byte come attributo. Avendo un nodo al massimo 8 figli, un byte risulta particolarmente comodo per il salvataggio di questo dato. Infatti basterà inserire "1" per indicare la presenza di un nodo figlio nel bit ad esso associato.

### 5.2.1.2 Suddivisione dell'ambiente tridimensionale

Una volta capita la struttura da usare per rappresentare un nodo, si è dovuto suddividere l'ambiente tridimensionale. A questo scopo sono stati introdotti i Bounding Box (BB). Con il termine Bounding Box si intende un cuboide<sup>14</sup> che racchiude una porzione di punti. Per esempio: il BB relativo a tutta la nuvola è il cuboide che ne contiene tutti i punti. È possibile dividere un ambiente tridimensionale partendo da un cuboide semplicemente suddividendolo ricorsivamente in 8 cuboidi ([Figura 5.14]). Possiamo quindi associare ogni BB ad un nodo della gerarchia. Così facendo ogni nodo conterrà una porzione di punti relativa alla posizione nello spazio del suo BB. Sorge a questo punto la necessità di identificare ogni nodo.

### 5.2.1.3 Identificazione nodi

Per poter gestire programmaticamente la gerarchia dei nodi di cui sopra, è necessario che ognuno di essi abbia un identificativo. A questo fine si è pensato di usare le prime 8 lettere

<sup>13</sup>Con il termine nodo viene inteso il singolo elemento dell'albero. Questo elemento corrisponde ad un file nella nuvola convertita.

<sup>14</sup>Figura solida composta da 6 facce regolari, in cui tutti gli angoli sono retti. (Fonte [24])

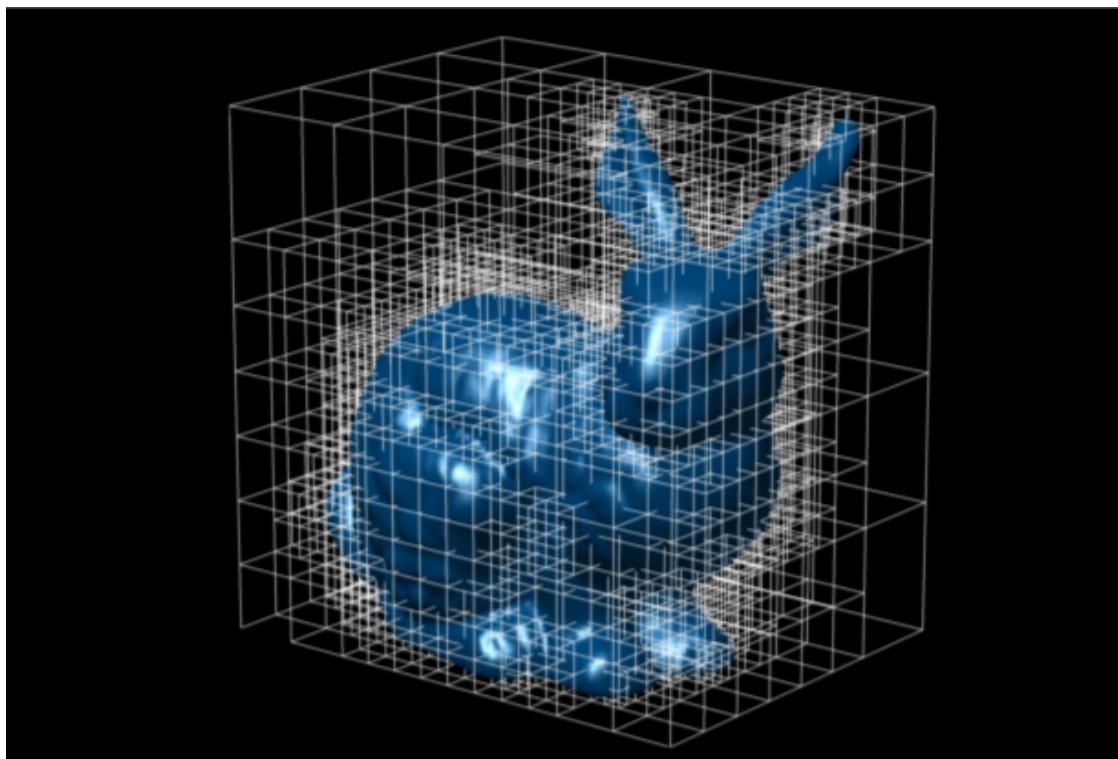


Figura 5.14: Suddivisione di un ambiente tridimensionale attraverso Octree e Bounding Box.  
(Fonte [25])

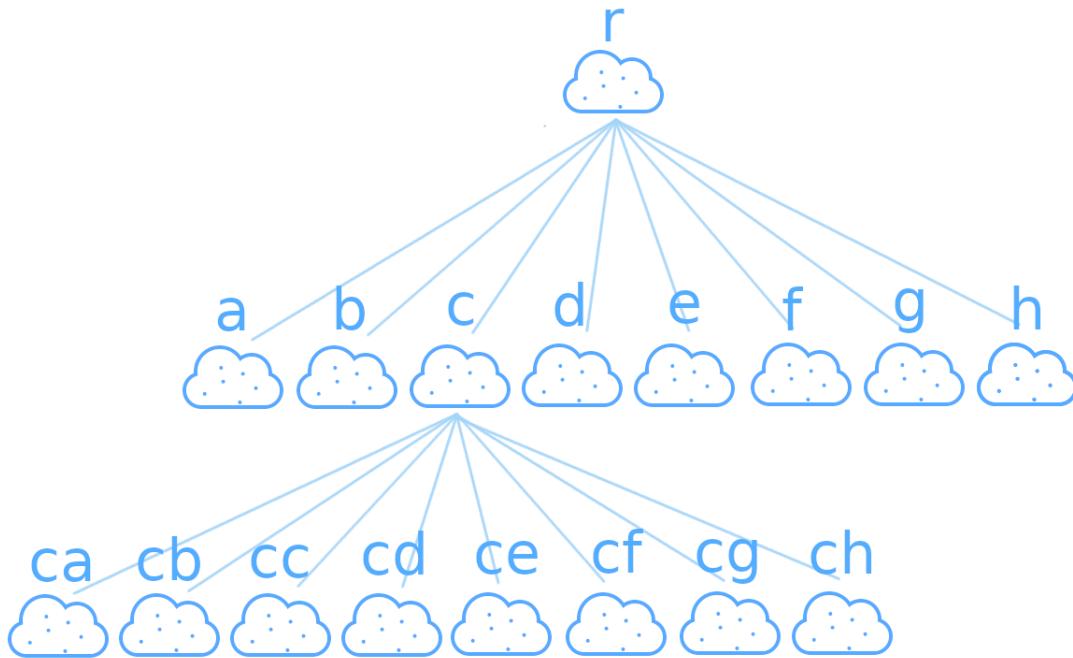


Figura 5.15: Illustrazione rappresentante il metodo di identificazione scelto per i nodi.

dell’alfabeto. Ogni nodo quindi otterrà una lettera da a ad h in base alla sua posizione nel livello della gerarchia. Eccezione fatta per il nodo radice che possiede l’identificativo ‘r’. Dal secondo livello (il livello radice rappresenta il livello 0) in poi ogni nodo viene identificato con l’identificativo del padre concatenato alla lettera del figlio. La Figura 5.15 illustra questa scelta. Grazie a questa struttura esiste un collegamento tra identificativo e Bounding Box. Infatti conoscendo il BB associato al nodo radice è possibile ottenere il BB di un nodo dal suo identificativo ([Figura 5.16]).

#### 5.2.1.4 Salvataggio delle meta-informationi

È stato scelto di aggiungere dei file di supporto alla struttura creata. Questi file consentono ad un programma che desidera leggere la nuvola di punti di conoscere le sue caratteristiche. Viene salvato per questo un file "cloud.json" contenente informazioni quali: nome della nuvola, numero di punti, BB del nodo radice e struttura ([Sezione 5.2.1.1]). Nel caso in cui la nuvola in input contenesse anche le classificazioni viene salvato un file "classes.json". Esso contiene il collegamento tra l’identificativo della classe e il suo nome (dizionario passato per parametro al generatore (Figura 5.13)). In Figura 5.17 è possibile osservare la cartella contenente la nuvola convertita completa delle meta-informationi.

Di seguito verrà descritta la parte software in grado di leggere e visualizzare la struttura generata.

```

1 def id_to_bb(self, id):
2     ret = self.bb
3
4     for c in id:
5         midx = midpoint(ret maxx, ret minx)
6         midy = midpoint(ret maxy, ret miny)
7         midz = midpoint(ret maxz, ret minz)
8
9         if c in "a":
10             ret = BoundingBox(ret minx, ret miny, ret minz, midx, midy, midz)
11         elif c in "b":
12             ret = BoundingBox(midx, ret miny, ret minz, ret maxx, midy, midz)
13         elif c in "c":
14             ret = BoundingBox(ret minx, midy, ret minz, midx, ret maxy, midz)
15         elif c in "d":
16             ret = BoundingBox(midx, midy, ret minz, ret maxx, ret maxy, midz)
17         elif c in "e":
18             ret = BoundingBox(ret minx, ret miny, midz, midx, midy, ret maxz)
19         elif c in "f":
20             ret = BoundingBox(midx, ret miny, midz, ret maxx, midy, ret maxz)
21         elif c in "g":
22             ret = BoundingBox(ret minx, midy, midz, midx, ret maxy, ret maxz)
23         elif c in "h":
24             ret = BoundingBox(midx, midy, midz, ret maxx, ret maxy, ret maxz)
25
26     return ret
27

```

Figura 5.16: Metodo appartenente alla classe BBManager per ottenere il Bounding Box (BB) relativo ad un identificativo. La variabile ret viene inizializzata al BB del nodo radice. Successivamente viene scelto il BB interno corrispondente alla lettera trovata nell'identificativo. Questo viene fatto per ogni lettera presente ed infine otteniamo così il BB riferito all'identificativo passato come argomento.



Figura 5.17: Screenshot della cartella contenente la nuvola convertita. I file json sono di supporto ad eventuali lettori di questo formato. Il file "r.bin" rappresenta il nodo radice, mentre all'interno della cartella "r" è presente il resto della gerarchia.

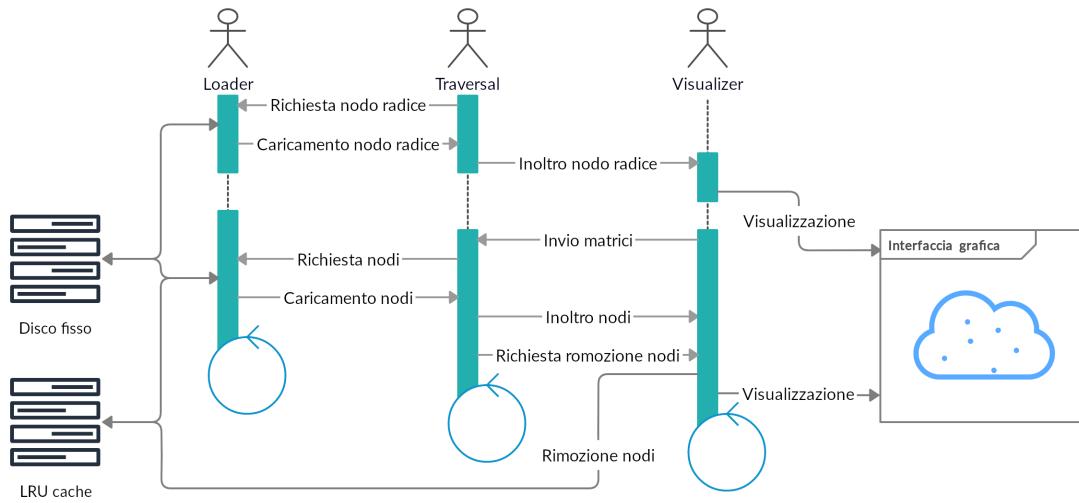


Figura 5.18: Diagramma di sequenza indicante le azioni che intercorrono tra i thread del visualizzatore. Quindi il caricamento e il passaggio di nodi, la loro visualizzazione, la loro rimozione, il loro salvataggio in cache, ecc...

### 5.2.2 Visualizzatore

Un visualizzatore con meccanismo di LOD deve compiere una grande quantità di lavoro. La computazione necessaria potrebbe rallentare di molto l'applicativo se svolta da un singolo processo. In considerazione di questo, e grazie alla lettura di [18], è stato scelto di basare il visualizzatore su 3 thread: `loader`, `traversal` e `visualizer`.

Il diagramma di sequenza in Figura 5.18 illustra come questi thread cooperano. Di seguito ogni thread viene spiegato nel dettaglio.

#### 5.2.2.1 Loader

Il thread `loader` ha il compito di caricare i nodi della gerarchia. A questo scopo viene utilizzato un oggetto di tipo `NodeLoader` messo a disposizione dal modulo `OctreeFormatTools` ([Figura 5.19]). `Loader` attende che venga richiesto un nodo e alla richiesta procede con il caricamento. Per questo sono state utilizzate due code condivise dai thread: `toload` usata per gli id dei nodi richiesti, `loaded` usata per i punti dei nodi caricati. La Figura 5.20 mostra il codice di questo procedimento. Inoltre, per ottimizzare il caricamento, è stata utilizzata una cache di tipo `lru`. Alla linea 5 del codice di Figura 5.20 è presente il controllo dell'eventuale presenza di un nodo in cache. Questo meccanismo consente di non dover interagire ogni volta con il disco fisso, ma mantenere dei nodi (secondo la logica "last recently used") in ram per un caricamento più rapido. I nodi caricati verranno in seguito gestiti dal thread `traversal`.

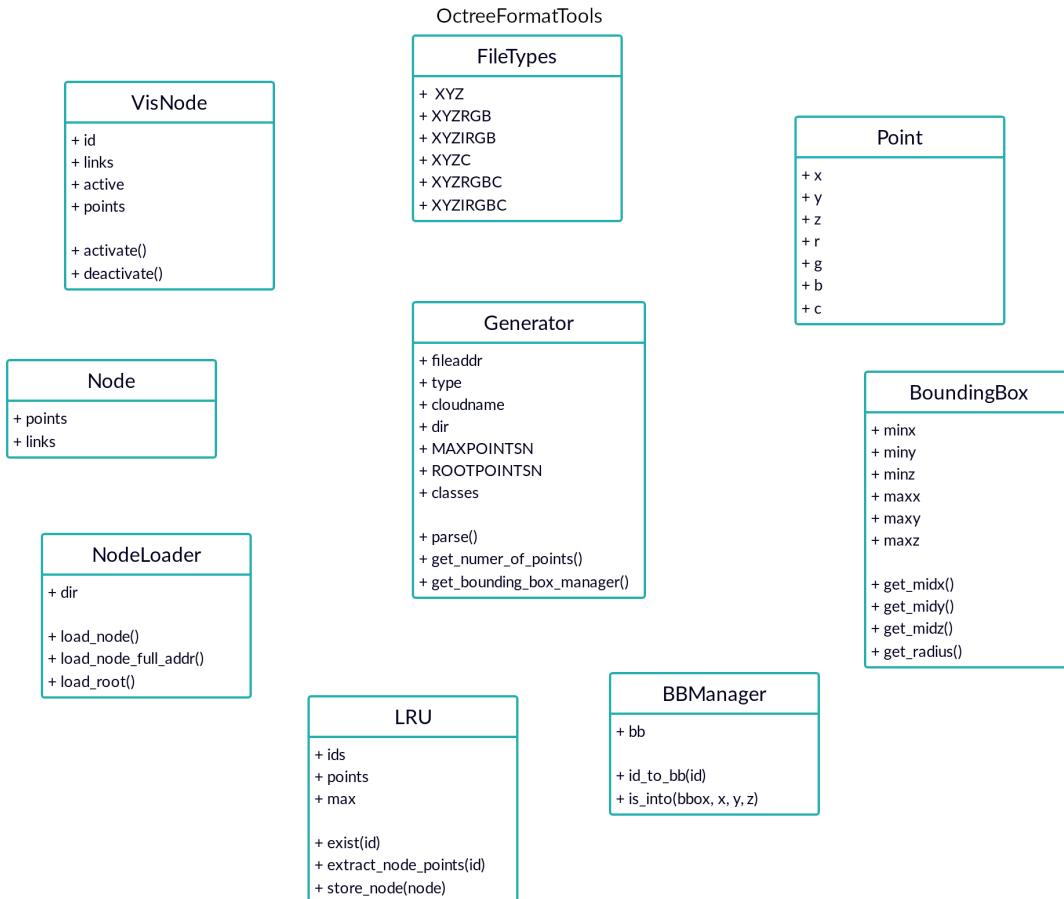


Figura 5.19: Diagramma rappresentante il modulo python OctreeFormatTools implementato. Oltre ad un generatore per la conversione di nuvole, esso mette a disposizione classi per poter interagire con la struttura creata, come ad esempio una classe `NodeLoader` per il caricamento di nodi, una cache `lru` per nodi, un Bounding Box manager ecc. Nell'immagine non sono state riportate le relazioni tra le classi al fine di favorire la leggibilità del diagramma (esse non fornivano informazioni utili al discorso trattato).

```

1  while True:
2      if not toload.empty():
3          id = toload.get()
4
5          if lru.exist(id):
6              loaded.put(lru.extract_node_points(id))
7          else:
8              loaded.put(nl.load_node_full_addr(id).points)
9

```

Figura 5.20: Codice python rappresentante il ciclo continuo che `loader` compie in attesa di nodi da caricare.

```

1 def is_point_infront(self, x, y, z):
2     return (self.a*x + self.b*y + self.c*z + self.d) > 0
3

```

Figura 5.21: Metodo della classe `Plane` che verifica se un punto si trova di fronte ad un piano espresso con l'equazione:  $ax + by + cz + d = 0$ .

### 5.2.2.2 Traversal

Il compito principale del thread `traversal` è quello di attraversare tutta la gerarchia e decidere quali nodi rimuovere e quali aggiungere al visualizzatore. A questo scopo, tramite la classe `VisNode` ([Figura 5.19]), viene inanzitutto creata una gerarchia fittizia: gerarchia di nodi che non contengono i punti. Dopo di che si itera su questa gerarchia valutando quali nodi aggiungere e quali rimuovere. Questa valutazione viene fatta basandosi sulla posizione di un nodo rispetto alla piramide di visualizzazione<sup>15</sup> e al livello di dettaglio che si desidera caricare.

Per poter capire quali nodi sono all'interno della piramide di visualizzazione è stato implementato il modulo `FrustumManager`. Questo modulo consente di estrarre le 6 equazioni dei piani che compongono la piramide e successivamente verificare se un Bounding Box (BB) si trova all'interno di essa. Infatti basandosi sull'utilizzo del metodo `is_point_infront` ([Figura 5.21]), è stato possibile creare un metodo che dato un BB ritorna un valore di visibilità ad esso associato. Questa variabile può assumere tre valori:

- 0: il BB non è visibile.
- 1: il BB è parzialmente visibile, ossia non tutti i punti che lo compongono sono all'interno del tronco di visualizzazione.
- 2: il BB è completamente visibile, quindi tutti i punti che lo compongono sono all'interno del tronco di visualizzazione.

Considerando che ad ogni nodo è associato un Bounding Box è stato possibile implementare la tecnica del frustum culling ([Sezione 3.3.2.1]) attraverso il modulo `FrustumManager`.

Conoscere la visibilità di un nodo serve per caricare o rimuovere tale nodo, ma non basta per un buon funzionamento della tecnica di LOD ([Sezione 3.3.3]). Per poter implementare un buon meccanismo bisogna infatti tenere in considerazione quanto l'utente si sta focalizzando (in termini di livello dello zoom) sui dettagli della nuvola. Per questo sono state introdotte altre due variabili:

---

<sup>15</sup>Piramide tronca costruita con i 6 piani che contengono tutta la scena 3D.

- distanza tra camera e punto medio del nodo di cui si sta considerando il caricamento/scaricamento: per poter capire quanto la camera è vicina al nodo, viene calcolata la distanza tra il punto medio del Bounding Box associato al nodo e la camera.
- numero di nodi interni completamente visibili: se i nodi figli di un nodo sono tutti completamente visibili, significa che l'utente sta visualizzando il nodo nella sua interezza. Se invece solamente pochi nodi figli sono visibili, vuol dire che l'utente si sta focalizzando su di un'altra area specifica della nuvola.

L'introduzione di queste due variabili ha portato all'implementazione finale del meccanismo di LOD del quale viene mostrato il codice in Figura 5.22. Valutando un nodo il meccanismo prevede i seguenti casi:

1. Il nodo ha una visibilità maggiore di zero e la variabile  $v^{16}$  è minore del 40% del numero di nodi presente nel nodo padre (riga 14). Il nodo viene aggiunto alla lista dei nodi candidati (`ids_to_evaluate`).
2. Il nodo ha una visibilità maggiore di zero e la distanza tra il centro del BB associato al nodo e la camera è minore o uguale al raggio<sup>17</sup> del BB (riga 14). Il nodo viene aggiunto alla lista dei nodi candidati (`ids_to_evaluate`).
3. Il nodo è stato precedentemente caricato ed aggiunto al visualizzatore. In questo istante però possiede un valore di visibilità pari a zero (riga 18). Quindi il nodo viene rimosso dal visualizzatore.
4. Il nodo è stato precedentemente caricato ed aggiunto al visualizzatore. Tutti i nodi del padre sono completamente visibili (riga 22). Il nodo e tutti i suoi nodi figli caricati vengono rimossi.

Dopo l'individuazione dei nodi candidati all'aggiunta e alla rimozione, bisogna decidere quali nodi andranno caricati e visualizzati per primi. Per questo è stato introdotto un calcolo di priorità. L'equazione 5.1 mostra la formula utilizzata, dove  $R$  rappresenta il raggio del BB,  $V$  il valore della visibilità e  $D$  la distanza dal centro del BB alla camera. In questo modo avranno la priorità i nodi con dimensioni maggiori, maggior vicinanza alla camera e maggior visibilità.

$$\rho = R \cdot V \cdot \frac{1}{D} \quad (5.1)$$

`Traversal` itererà sulla lista di nodi candidati, estraendo il nodo con priorità maggiore e lo aggiungerà alla coda dei nodi da visualizzare. Sarà poi compito del thread `visualizer` controllare questa coda, estrarne i nodi e visualizzarli.

<sup>16</sup>Rappresenta il numero di nodi completamente visibili nel padre del nodo che si sta valutando.

<sup>17</sup>Dimensione media di un BB che consente di rappresentarlo come se fosse una sfera. Nello specifico il raggio sarebbe la metà della media delle dimensioni sugli assi x, y e z.

```
1 def modify_nodes(r, rendered_nodes):
2     v = 0
3
4     # calculate how many bb are completely visible
5     for n in r.links:
6         if fm.bbox_in_frustum(bbman.id_to_bb(n.id)) == 2:
7             v += 1
8
9     for n in r.links:
10        bb = bbman.id_to_bb(n.id)
11        visible = fm.bbox_in_frustum(bb)
12        dist = fm.dist_from_near(bb.get_midx(), bb.get_midy(), bb.get_midz())
13
14        if visible > 0 and (v <= 0.4 * len(r.links) or dist <= bb.
15            get_radius()) and v != len(r.links) and not n.active:
16            ids_to_evaluate.append(n.id)
17            bb_to_evaluate.append(bb)
18            bb_visibility.append(visible)
19        elif n.active and visible == 0:
20            remove(n)
21            n.deactivate()
22            rendered_nodes -= 1
23        elif n.active and v == len(r.links):
24            rendered_nodes = remove_branch(n, rendered_nodes)
25
26    rendered_nodes = modify_nodes(n, rendered_nodes)
27
28    return rendered_nodes
29
```

Figura 5.22: Funzione appartenente al thread traversal con il compito di trovare i nodi da caricare e i nodi da rimuovere. Questa funzione opera sulla gerarchia fittizia e inserisce gli id dei nodi da caricare nella lista `ids_to_evaluate` (riga 15).

```

1 def add_geometry():
2     updated = not torender.empty()
3
4     if updated:
5         node = torender.get()
6         onrenderid.append(node.id)
7         onrender.append(node)
8         vis.add_geometry(create_pcd(node))
9
10    return updated
11

```

Figura 5.23: Funzione appartenente al thread `visualizer` che aggiunge un eventuale nodo al contesto di Open3D.

```

1     def remove_geometry():
2         if not todelete.empty():
3             id = todelete.get()
4             index = onrenderid.index(id)
5             cached.put(onrender.pop(index))
6             vis.remove_geometry(onrenderid.pop(index))
7

```

Figura 5.24: Funzione appartenente al thread `visualizer` che rimuove un nodo dal contesto di Open3D.

### 5.2.2.3 Visualizer

Il thread `visualizer` ha il compito di visualizzare la nuvola di punti composta da tutti i nodi ricavati secondo il meccanismo precedentemente descritto. L'insieme di nodi visualizzati risulta in continuo mutamento. Infatti per la generazione di ogni frame `visualizer` si occupa principalmente di:

- controllare la presenza di nodi nella coda `torender`. Questa coda contiene i nodi che dovranno essere visualizzati. Se sono presenti nodi da visualizzare, `visualizer` li aggiunge al contesto di Open3D. Il codice in Figura 5.23 mostra questo procedimento.
- controllare la presenza di id nella coda `todelete`. Questa coda contiene tutti gli identificativi dei nodi che vanno rimossi dal visualizzatore. `Visualizer` dovrà quindi rimuovere il nodo associato all'id estratto. Implementazione in Figura 5.24.

`Visualizer` è l'unico thread ad utilizzare Open3D, quindi anche l'unico a poter ottenere le informazioni relative alla posizione della camera. Dato che queste informazioni sono fondamentali per `traversal` un altro compito di `visualizer` è fornire questi dati. Nello specifico si tratta di recuperare tramite Open3D la matrice `mvp`<sup>18</sup> (`Model * View * Projection`) e

<sup>18</sup>Matrice ottenuta dalla moltiplicazione delle matrici `Model`, `View` e `projection`. Tale matrice consente l'estrazione dei 6 piani che contengono la scena 3D, fondamentali per capire se un oggetto 3D verrà visualizzato in base alla posizione della camera nell'ambiente.

passarla a traversal, il quale dalla matrice potrà estrarre le equazioni necessarie al frustum culling ([Sezione 3.3.2.1]) e quindi al meccanismo di LOD ([Sezione 3.3.3]). Per consentire questo passaggio di dati e per altre motivazioni si è dovuto apportare delle modifiche alla libreria in uso, che in seguito verranno descritte.

### 5.2.3 Modifiche apportate ad Open3D

Per poter implementare quanto descritto in precedenza ([Sezione 5.2.2]), è stato necessario modificare il codice sorgente di Open3D. Serviva un visualizzatore che mettesse a disposizione principalmente tre funzionalità:

1. Registrazione di una callback chiamata alla generazione di ogni frame. Questa callback risulta utile per il lavoro svolto da `visualizer`, che per ogni frame generato deve fare delle operazioni.
2. Selezione punti. La possibilità di selezionare un punto è necessaria per implementare un meccanismo che permetta di sapere la classe che l'algoritmo di classificazione ha associato al punto.
3. Registrazione di una callback che ritorni l'identificativo della classe relativa al punto selezionato.

Le prime due funzionalità sono già presenti in Open3D, ma in classi diverse. È stata creata una classe C++ che include entrambe le opzioni. Si è scelto di chiamare tale classe `VisualizerWithKeyAndEdit`. Inoltre, essendo il codice C++ di Open3D esposto in python attraverso pybind<sup>19</sup>, è stato necessario aggiungere del codice per rendere disponibile nelle api python questa classe. Per il recupero della matrice mvp come descritto precedentemente, è bastato aggiungere il collegamento con pybind ad un metodo getter che di default non viene esposto.

Per quanto riguarda invece la terza funzionalità, è stato necessario modificare la classe `PointCloud` aggiungendo come attributo una lista di id (ogni id è riferito ad una classe). In questo modo una nuvola di punti potrà avere un id per ogni punto. Così facendo, alla selezione del punto, oltre alle coordinate, è possibile leggere anche l'id della classe associata. Una volta ottenuto l'id viene passato ad una callback registrata attraverso le api python. La Figura 5.25 mostra il codice python per quest'operazione. Mentre nella Figura 5.26 è presente il codice C++ che recupera e passa l'id della classe. La parte software che utilizza Open3D sarà quindi in grado di ottenere l'id della classe del punto selezionato. Grazie all'id si può poi recuperare il nome della classe dal dizionario (file `classes.json`) e stamparlo nel terminale.

---

<sup>19</sup>Libreria in grado di esporre del codice C++ in python. (Fonte [26])

```
1 def on_point_picked(id):
2     picked.put(id)
3
4 vis.register_onpick_callback(on_point_picked)
5
```

Figura 5.25: Codice python per registrare una callback chiamata alla selezione di un punto. Questa callback riceve l'id della classe del punto selezionato.

```
1 int index = PickPoint(x, y);
2 if (index == -1) {
3     onpick(-1);
4 } else {
5     auto pcd = ((const geometry::PointCloud &)(*editing_geometry_ptr_));
6
7     if(!pcd.classes_.empty()){
8         int classid = pcd.classes_[index];
9         onpick(classid);
10        pointcloud_picker_ptr_->picked_indices_.push_back((size_t)index);
11        is_redraw_required_ = true;
12    }
13 }
```

Figura 5.26: Codice C++ che all'evento del punto selezionato, trova il suo indice nei vettori, recupera l'id della classe e lo passa attraverso la funzione registrata in precedenza.

Si conclude così l'implementazione, in seguito verranno descritti i risultati ottenuti.



# Capitolo 6

## Risultati

Il prototipo realizzato, basato su Open3D, è un visualizzatore con meccanismo di LOD ([Sezione 3.3.3]) in grado di mostrare le classi associate ai punti selezionati. È possibile selezionare un punto premendo il tasto CTRL e il tasto destro del mouse. Inoltre questo visualizzatore permette di muoversi nell'ambiente 3D attraverso i classici tasti W, A, S e D e di poter tornare alla visualizzazione iniziale premendo R.

Si è scelto di chiamare questo prototipo CloudInterpreter. Il successo di questo prototipo dimostra la fattibilità di uno strumento in grado di visualizzare e interagire con una nuvola di punti ad alta densità. CloudInterpreter può essere una buona base di partenza per uno strumento del genere. In seguito vengono descritti i principali risultati ottenuti.

### 6.0.1 Visualizzatore con meccanismo di LOD

Il meccanismo di LOD implementato ha consentito di migliorare le prestazioni. Per verificare questo miglioramento sono state fatte delle prove con la nuvola grande ([Sezione 5.1.3.2]). Una notevole differenza è stata riscontrata all'apertura della nuvola. L'apertura della nuvola con CloudInterpreter risulta infatti istantanea ([Figura 6.1]). Un'altra differenza sta nel valore degli fps e quindi indirettamente nel frame-time. Come possiamo notare nel grafico in Figura 6.2 il prototipo realizzato è molto più veloce nella generazione di un nuovo frame. Infatti un frame appare sullo schermo per un tempo di 0.022 secondi. Mentre con la libreria originale questo tempo era di 1.08 secondi.

Il risultato è un visualizzatore fluido e veloce nell'apertura delle nuvole indipendentemente dalla loro densità. Questo ha permesso il raggiungimento di uno degli obiettivi principali del progetto di diploma ([Sezione 2.5]). La Figura 6.3 mostra un'istantanea ottenuta durante l'implementazione in cui è possibile notare un differente livello di dettaglio tra le parti visualizzate della nuvola.

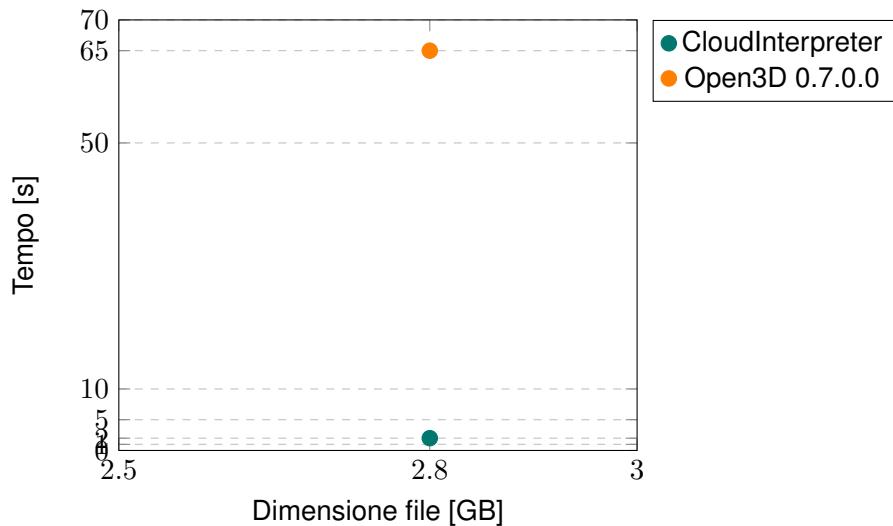


Figura 6.1: Tempo di caricamento della nuvola.

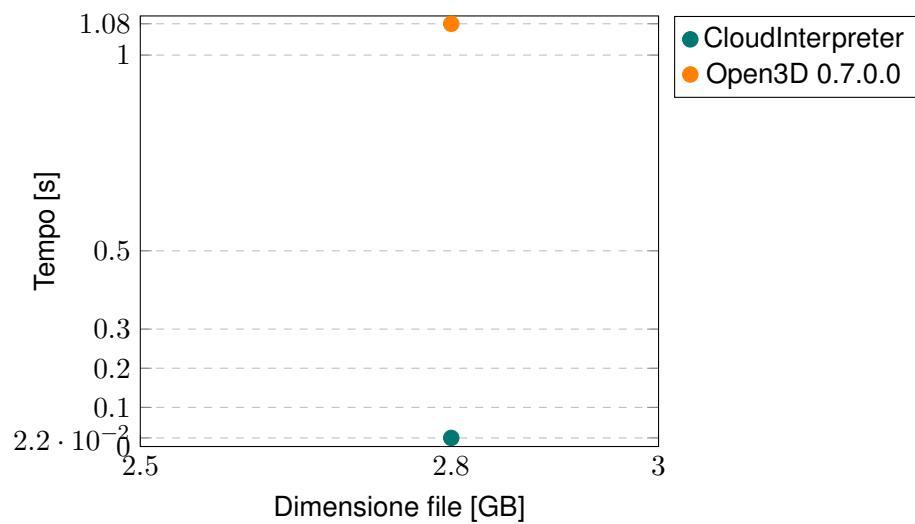


Figura 6.2: Frame-Time durante lo zoom.

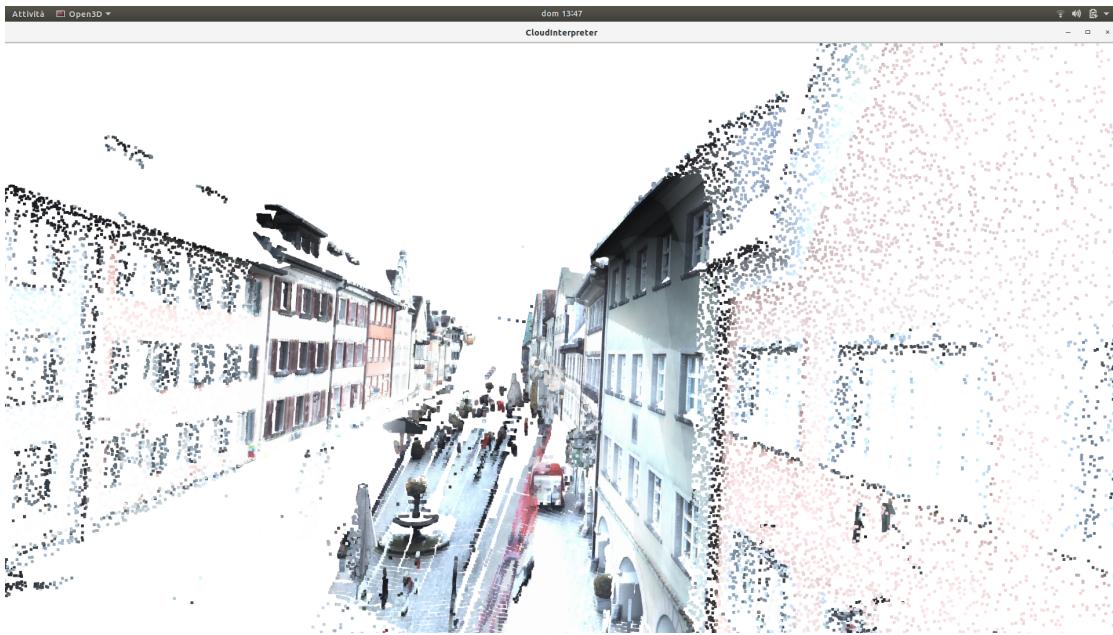


Figura 6.3: Prove durante l'implementazione del meccanismo di LOD. È possibile notare come la parte centrale della nuvola sia caricata con un livello di dettaglio maggiore.

### 6.0.2 "Rilevatore" di classi

Il "rilevatore" di classi consente di conoscere la classe che l'algoritmo di classificazione ha associato ad un punto selezionato. Questa funzionalità è molto utile per conoscere la bontà dell'algoritmo che ha effettuato la classificazione. La Figura 6.4 mostra questa funzionalità con un esempio di classificazioni generate da un algoritmo. Tale funzionalità rappresentava uno degli altri obiettivi principali di questo progetto di diploma ([Sezione 2.5]).

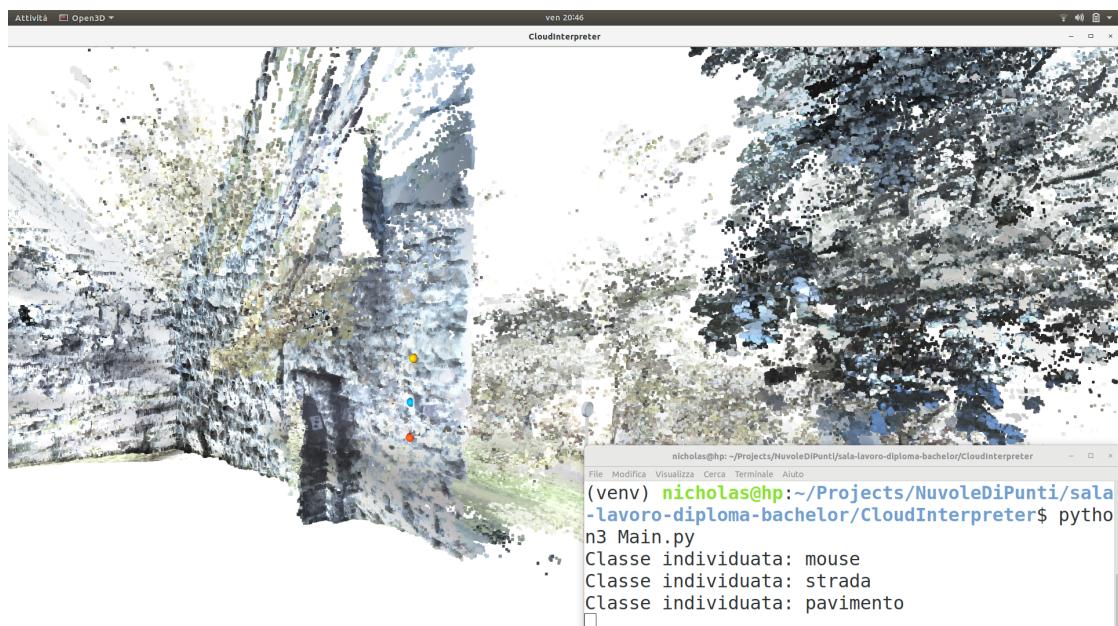


Figura 6.4: Individuazione di tre classi. La sfere indicano i punti selezionati. Le classificazioni sono casuali e puramente d'esempio.

# Capitolo 7

## Conclusione

In questo capitolo vengono presentate le conclusioni inerenti a tutto il lavoro svolto durante questo progetto di diploma. Verrà descritto anche come questo lavoro mi abbia permesso di acquisire nuove competenze e di ampliare le mie conoscenze in ambito informatico.

### 7.1 Sviluppi futuri

Nonostante l'ottimo risultato raggiunto con il prototipo, vi sono degli aspetti che in un proseguimento di progetto, si potrebbero approfondire. Di seguito vengono elencati gli sviluppi futuri che si potrebbero apportare al prototipo realizzato.

- Velocizzazione del convertitore: il convertitore per le nuvole di punti soprattutto con nuvole dense, risulta lento. Questo è dovuto alla grande computazione necessaria ed al fatto che questa conversione è eseguita da un singolo processo. Uno sviluppo futuro potrebbe quindi essere la sua velocizzazione, probabilmente eseguendo quest'operazione su più thread.
- Velocizzazione dell'aggiunta/rimozione di nodi dalla visualizzazione: la parte che richiede una computazione maggiore durante la visualizzazione è l'aggiunta o la rimozione di nodi. Infatti ogni volta si va a modificare una nuvola di punti (nel contesto di Open3D) e questo richiede tempo perché bisogna iterare su una serie di punti per aggiungerli o per rimuoverli. Questo è l'unico modo che attualmente è stato trovato per poter mantenere la possibilità di selezionare un punto. La fluidità potrebbe essere molto maggiore se quest'operazione venisse fatta in modo ottimizzato, operando quindi su una lista di nuvole di punti e non modificando sempre la stessa istanza di una nuvola.
- Tuning dei parametri: con lo scopo di ottenere buone prestazioni con il visualizzatore, è stato scelto di aggiungere o rimuovere alla lista dei nodi visualizzati al massimo un nodo per frame generato. Inoltre il numero di nodi aggiunti è limitato in modo da non

sovrcaricare la memoria. Questi valori potrebbero essere modificate per cercare di migliorare le prestazioni del visualizzatore.

## 7.2 Competenze acquisite

Lo studio e la prototipazione delle librerie ([Sezione 5.1]) ha portato all'acquisizione di una buona competenza nel compilare librerie complesse. Infatti per provare le librerie è stato necessario compilare il loro codice e spesso risolvere problemi legati alle dipendenze da terze parti.

Un'altra buona competenza acquisita è la programmazione in python. Questo linguaggio di programmazione era già stato visto durante il corso degli studi, ma non in modo dettagliato. Essendo la maggior parte del codice sviluppato codice python, molto è stato imparato sulle tecniche di programmazione, la sintassi e la semantica di questo linguaggio.

Inoltre grazie al progetto di diploma è stato possibile imparare a valutare oggettivamente una libreria (in base ai requisiti di progetto).

## 7.3 Ringraziamenti

Ringrazio il relatore di questo lavoro di diploma Fabio Landoni per aver seguito diligentemente l'evolversi del progetto. Grazie ai suoi consigli è stato possibile seguire un corretto approccio al problema ([Capitolo 4]), portare a termine l'implementazione ([Capitolo 5]) e scrivere una buona documentazione tecnica. Ringrazio inoltre il correlatore Roberto Guidi per la sua disponibilità durante il corso del progetto di diploma.

# Bibliografia

- [1] Inf574 - digital representation and analysis of shapes.
- [2] M. carlberg, p. gao, g. chen, and a. zakhor - classifying urban landscape in aerial lidar using 3d shape analysis.
- [3] Wikipedia. Classificazione — wikipedia, l'enciclopedia libera, 2018. [Online; in data 29-agosto-2019].
- [4] Wikipedia. Telerilevamento — wikipedia, l'enciclopedia libera, 2019. [Online; in data 29-agosto-2019].
- [5] Spar3d - leica's rtc 360 lidar scanner registers your point clouds automatically, in real time, in the field.
- [6] Paul bourke - collection of references to a range of data formats in use in the computer industry.
- [7] Laser scanning forum - about 3d point cloud format.
- [8] Stackoverflow - how to display point cloud in vtk with different colors ?.
- [9] Lighthouse3d - view frustum culling.
- [10] Silent's blog - clever bug exploitation - backface culling.
- [11] Claus Scheiblauer. *Interactions with Gigantic Point Clouds*. PhD thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, 2014.
- [12] Gkocree - a data structure for organizing objects based on their locations in a three-dimensional space.
- [13] Wikipedia. Metodologia agile — wikipedia, l'enciclopedia libera, 2019. [Online; in data 21-agosto-2019].
- [14] Wikipedia. Scrum (informatica) — wikipedia, l'enciclopedia libera, 2019. [Online; in data 21-agosto-2019].

- [15] Wikipedia contributors. Continuous delivery — Wikipedia, the free encyclopedia, 2019. [Online; accessed 21-August-2019].
- [16] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.
- [17] Markus Sch' Potree: Rendering large point clouds in web browsers. Master's thesis.
- [18] Simon Maximilian Fraiss. Rendering large point clouds in unity, September 2017.
- [19] Jeff Baumes. Information visualization in vtk.
- [20] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.
- [21] The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 4.14 edition, 2019.
- [22] pptk - point processing toolkit.
- [23] Timo Hackel, N. Savinov, L. Ladicky, Jan D. Wegner, K. Schindler, and M. Pollefeys. SEMANTIC3D.NET: A new large-scale point cloud classification benchmark. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume IV-1-W1, pages 91–98, 2017.
- [24] Wikipedia. Cuboide (geometria) — wikipedia, l'enciclopedia libera, 2015. [Online; in data 24-agosto-2019].
- [25] Octree - partitioning 3d points into spatial subvolumes.
- [26] pybind11 - seamless operability between c++11 and python.

## **Appendice A**

### **Scheda di progetto**

Viene allegata in seguito la scheda descrittiva originale del progetto.

## Nuvole di punti ad alta densità: visualizzazione ed interazione

### Persone coinvolte

PropONENTE	Landoni Fabio
RELATORE	Landoni Fabio
CORRELATORE	Guidi Roberto
STUDENTE	Sala Nicholas

### Dati generali

Codice	C10105
Anno accademico	2018/2019
Semestre	Semestre estivo
Corso di laurea	Ingegneria informatica (Informatica TP)
Opzione	Nessuna opzione
Tipologia del progetto	diploma
Stato	in corso
Confidenziale	SI
Pubblicabile	SI

### Descrizione

Nell'ambito di un progetto più ampio si sta sviluppando un sistema per la classificazione di nuvole di punti ad elevata densità.

Allo scopo di capire la bontà degli algoritmi di classificazione, è necessario poter rappresentare questi oggetti.

Con questo lavoro di diploma si vuole realizzare un prototipo in grado di consentire la visualizzazione e l'interazione con le nuvole di punti: avvicinarsi, allontanarsi, ruotarle, muoversi al loro interno, selezionarne una parte,...

A seguito di una serie di processi, estranei a questo progetto, è possibile avere più versioni di una stessa nuvola. Sarebbe quindi interessante che lo studente lavorasse allo sviluppo di un meccanismo in grado di visualizzare contemporaneamente più versioni della stessa nuvola (verosimilmente due) evidenziandone le differenze.

Allo studente è anche richiesto di individuare e sviluppare ciò che possa favorire una performance di utilizzo ragionevole. Si potrebbe per esempio ragionare sull'identificazione di una struttura dati adeguata, tecniche per caricare solamente una parte rappresentativa dei punti, ....

A dipendenza della bontà e della maturità della soluzione proposta, si potrà valutare la possibilità di integrare quanto proposto direttamente nel sistema di classificazione.

### Compiti

- Elicitare requisiti e vincoli.
- Identificare ed investigare possibili tecnologie, tecniche e librerie da utilizzare, focalizzandosi su quelle più promettenti.
- Ove possibile, capire il funzionamento delle librerie.
- Sperimentare alcuni approcci e, dopo aver individuato un sistema di confronto oggettivo, paragonarli fra loro.
- Sviluppare uno o più prototipi.
- Valutare come dovrebbero essere strutturati i dati al fine di facilitarne la visualizzazione.
- Individuare e valutare possibili migliorie.
- Documentare e presentare il lavoro svolto.

### Obiettivi

- Confrontarsi con il processo di sviluppo di un sistema che richieda di valutare molteplici approcci.
- Implementare e confrontare uno o più prototipi in grado di soddisfare le esigenze rispettando i vincoli di progetto.
- Redigere una documentazione con lo scopo di illustrare tutte le fasi del lavoro.
- Affrontare il progetto in maniera efficace, gestire il backlog di prodotto e le interazioni di sviluppo.

### Tecnologie

Da identificare. Verosimilmente: Unity, javascript/typescript, e python.

### Contatto esterno

Nessun contatto esterno presente

## Appendice B

### Email

Come precedentemente accennato sono state inviate delle email agli sviluppatori della libreria 3DTK ([Sezione 5.1.1]) al fine di risolvere un errore in fase di compilazione. Le email ricevute hanno permesso di risolvere il problema: si trattava di una libreria mancante. Di seguito viene riportato lo scambio di email.

da nicholas.sala@student.supsi.ch a andreas@nuechti.de  
Oggetto: 3DTK compile error under ubuntu

Dear Sir

I'm a computer science student in University of Applied Sciences and Arts of Southern Switzerland (SUPSI) and i'm preparing a Bachelor Thesis about gigantic point clouds render.

I have found your library 3DTK, and i have downloaded a snapshot here:  
<https://sourceforge.net/p/slam6d/code/HEAD/tree/>. I'm using Ubuntu 18.04.2 LTS so have installed all dependencies, an then during the make call i have this error:

CMake Error: The following variables are used in this project, but they are set to NOTFOUND.

Please set them or make sure they are set and tested correctly in the CMake files:

FTGL\\_INCLUDE\\_DIR

used as include directory in directory /home/nicholas/Scrivania/slam6d-code-r2098-trunk/src/pmd

used as include directory in directory /home/nicholas/Scrivania/slam6d-code-r

```
2098-trunk/src/pmd  
used as include directory in directory /home/nicholas/Scrivania/slam6d-code-r  
2098-trunk/src/pmd  
used as include directory in directory /home/nicholas/Scrivania/slam6d-code-r  
2098-trunk/src/pmd
```

-- Configuring incomplete, errors occurred!

Do you know if there is a fix for this error ?  
I have also found this repository <https://github.com/3DTK/3DTK>, can i ask  
you what is the relation between the two repo ?

Thank you  
Best regards  
Sala Nicholas

da andreas@nuechti.de a nicholas.sala@student.supsi.ch  
Re: 3DTK compile error under ubuntu

Dear Sala,

thanks for your email. Did you follow the install instructions?

You need FTGL for compiling this.

Very best,  
Andreas

da johannes.schauer@uni-wuerzburg.de a nicholas.sala@student.supsi.ch  
Re: 3DTK compile error under ubuntu

Hi Sala,

Quoting Andreas Nuechter (2019-06-27 23:21:59)  
> > I'm a computer science student in University of Applied Sciences and Arts  
> > of Southern Switzerland (SUPSI) and i'm preparing a Bachelor Thesis about  
> > gigantic point clouds render.

you have come to the right place! :)

```
> > I have found your library 3DTK, and i have downloaded a snapshot here:  
> > https://sourceforge.net/p/slamp6d/code/HEAD/tree/. I'm using Ubuntu  
> > 18.04.2 LTS so have installed all dependencies,
```

How did you install the dependencies?

```
> > an then during the make call i have this error:  
> >  
> > CMake Error: The following variables are used in this project, but they  
> > are set to NOTFOUND.  
> > Please set them or make sure they are set and tested correctly in the  
> > CMake files:  
> > FTGL_INCLUDE_DIR  
> >     used as include directory in directory  
> > /home/nicholas/Scrivania/slamp6d-code-r2098-trunk/src/pmd  
> >     used as include directory in directory  
> > /home/nicholas/Scrivania/slamp6d-code-r2098-trunk/src/pmd  
> >     used as include directory in directory  
> > /home/nicholas/Scrivania/slamp6d-code-r2098-trunk/src/pmd  
> >     used as include directory in directory  
> > /home/nicholas/Scrivania/slamp6d-code-r2098-trunk/src/pmd  
> >  
> > -- Configuring incomplete, errors occurred!  
> >  
> > Do you know if there is a fix for this error ?
```

As Andreas said, you have to install ftgl. We regularly test building 3DTK on Ubuntu 18.04 and it all builds fine (including with ftgl):

<https://travis-ci.org/3DTK/3DTK/>

```
> > I have also found this repository https://github.com/3DTK/3DTK, can i  
> > ask you what is the relation between the two repo ?
```

You can read it at the top:

"!!! TEST ONLY !!! DO NOT USE !!!"

We are evaluating certain things by mirroring our official sourceforge repository on github. But the official sources are on sourceforge in svn. If you prefer using git, then just use git-svn. But don't use the github repo.

Thanks!

cheers, josch

## Appendice C

# Esempi formati nuvole di punti

Di seguito vengono mostrati degli esempi di ogni formato delle nuvole di punti rappresentabile in modo testuale ([Sezione 1.1.1]). Questi esempi sono stati ottenuti tramite le fonti [6] e [7].

### C.1 XYZ

```
-0.007305 0.479848 -1.825868  
-0.071243 0.478286 -1.821239  
-0.070341 0.478542 -1.822264  
  
-0.007305 0.479848 -1.825868 51 153 255  
-0.071243 0.478286 -1.821239 51 153 255  
-0.070341 0.478542 -1.822264 51 153 255
```

### C.2 PTS

```
342684  
4.246445 -6.426620 -50.214615 -399 66 50 83  
-9.318283 -6.014053 -51.129257 -437 132 141 152  
-4.043655 -14.000992 -65.560776 -332 69 61 85
```

### C.3 PTX

number of columns

```

number of rows
st1 st2 st3 ; scanner registered position
sx1 sx2 sx3 ; scanner registered axis 'X'
sy1 sy2 sy3 ; scanner registered axis 'Y'
sz1 sz2 sz3 ; scanner registered axis 'Z'
r11 r12 r13 0 ; transformation matrix
r21 r22 r23 0 ; this is a simple rotation and translation 4x4 matrix
r31 r32 r33 0 ; just apply to each point to get the transformed coordinate
tr1 tr2 tr3 1 ; use double-precision variables

```

## C.4 PLY

```

format ascii 1.0 ( ascii/binary, format version number )
comment made by Greg Turk ( comments keyword specified, like all lines )
comment this file is a cube
element vertex 8 ( define "vertex" element, 8 of them in file )
property float x ( vertex contains float "x" coordinate )
property float y ( y coordinate is also a vertex property )
property float z ( z coordinate, too )
element face 6 ( there are 6 "face" elements in the file )
property list uchar int vertex_index ( "vertex_indices" is a list of ints )
end_header ( delimits the end of the header )
0 0 0 start of vertex list
0 0 1
...
1 1 0
4 0 1 2 3 start of face list
4 7 6 5 4

```

## Appendice D

# Risultati prove librerie (seconda iterazione)

In questo capitolo delle Appendici vengono presentati in modo dettagliato i risultati ottenuti per ogni libreria valutata nella seconda iterazione ([Sezione 5.1.3]). Questi valori sono suddivisi per ogni file utilizzato ([Sezione 5.1.3.2]) e per ogni parametro di valutazione scelto ([Sezione 5.1.3.1]). Ricordiamo che con il termine conversioni, si intende una modifica del file di input con lo scopo di renderlo compatibile alla struttura supportata dalla libreria.

### D.1 Legenda parametri

1. tempo di caricamento del file comprese eventuali conversioni
2. tempo di caricamento del file senza conversione
3. fps durante lo zoom
4. fps durante il movimento (con mouse) della nuvola
5. frame-time durante lo zoom
6. frame-time durante il movimento (con mouse) della nuvola
7. quantità di ram utilizzata dal processo
8. percentuale di utilizzo della cpu durante lo zoom
9. tempo di selezione dei punti scelti (tempo tra click e selezione effettiva)
10. meccanismo di caricamento della nuvola
11. interazione programmatica

### D.2 Misurazioni ottenute

Inizialmente le misurazioni sono state fatte con il file medio ([Sezione 5.1.3.2]). Da queste misurazioni si è capito che non era presente una grossa differenza, nelle prestazioni delle librerie, tra computer con scheda grafica dedicata e computer con scheda grafica integrata.

Perciò le misurazioni con gli altri file sono state fatte solamente con il primo pc (quello munito di scheda grafica dedicata). Di seguito vengono presentati tali risultati.

Risultati ottenuti con HP Envy 17:

Libreria: PCL (1.9.1)			
	nuvola piccola	nuvola media	nuvola grande
1	110 s	245 s	434 s
2	63.65 s	179.5 s	236 s
3	12.9	6.95	11.78
4	13.34	9.65	2.34
5	0.077 s	0.143 s	0.084 s
6	0.074 s	0.103 s	0.42 s
7	1.54 GiB	4.6 GiB	5.5 GiB
8	42%	40%	40%
9	non implementato		
10	caricamento completo		
11	si		

Libreria: Open3D (0.7.0.0)			
	nuvola piccola	nuvola media	nuvola grande
1	77 s	224 s	290 s
2	27.45	62.5 s	65
3	3.89	1.5	0.92
4	13.95	3.27	1.51
5	0.25 s	0.6 s	1.08 s
6	0.071 s	0.305 s	0.66 s
7	1.05 GiB	1.9 GiB	3.6 GiB
8	14%	27%	27%
9	1 s		
10	caricamento completo		
11	si		

Libreria: VTK (8.2.0)			
	nuvola piccola	nuvola media	nuvola grande
1	40 s	123 s	198 s
2	40 s	123 s	198 s
3	7.45	4.4	4.35
4	18.95	9	5.86
5	0.13 s	0.227 s	0.229 s
6	0.052 s	0.111 s	0.17 s
7	0.74 GiB	2.2 GiB	2.83 GiB
8	27%	13%	29%
9	non implementato		
10	soglia		
11	si		

Libreria: Potree 1.6			
	nuvola piccola	nuvola media	nuvola grande
1	133 s	211 s	293 s
2	4 s	4 s	5 s
3	48	34	46
4	52	48	48
5	0.020 s	0.029 s	0.021
6	0.019 s	0.020 s	0.020 s
7	0.642 GiB	0.621 GiB	0.651 GiB
8	30.7%	30%	41.5%
9	non implementato		
10	LOD nested-Octree based		
11	si		

Libreria: BA_Pointcloud 1.2			
	nuvola piccola	nuvola media	nuvola grande
1	133 s	212 s	293 s s
2	4 s	5 s	5 s
3	97.6	99.6	91.6
4	117.2	97	84.14
5	0.010 s	0.010 s	0.010 s
6	0.008 s	0.010 s	0.011 s
7	0.1 GiB	0.1 GiB	0.1 GiB
8	25%	28%	29%
9	non implementato		
10	LOD nested-Octree based		
11	si		

Risultati ottenuti con Lenovo thinkpad t470s:

Libreria: PCL (1.9.1)	
	nuvola media
1	274 s
2	198 s
3	12.89
4	21.25
5	0.077 s
6	0.047 s
7	2.4 GiB
8	39%
9	non implementato
10	caricamento completo
11	sì

Libreria: Open3D (0.7.0.0)	
	nuvola media
1	208 s
2	46.6 s
3	3.01
4	8.5
5	0.332 s
6	0.117 s
7	2.7 GiB
8	6%
9	1 s
10	caricamento completo
11	c si

Libreria: VTK (8.2.0)	
	nuvola media
1	135.19 s
2	135.19 s
3	1.46
4	1.3
5	0.684 s
6	0.769 s
7	0.6 GiB
8	4%
9	non implementato
10	soglia
11	si

Libreria: Potree 1.6	
	nuvola media
1	214 s
2	7 s
3	26.8
4	21.5
5	0.037 s
6	0.046 s
7	0.41 GiB
8	19%
9	non implementato
10	LOD nested-Octree based
11	si