UNIVERSITÀ DEGLI STUDI DI TRENTO

Dipartimento di Ingegneria e Scienza dell'Informazione

Corso di Laurea in
Informatica

ELABORATO FINALE

RICOSTRUZIONE 3D AUTOMATICA DI EDIFICI DA NUVOLE DI PUNTI

Supervisori
Prof. Mauro Brunato
Prof. Fabio Remondino

Laureanda
Elisa De Gasperin

Anno accademico 2015/2016

Ringraziamenti

Ringrazio il mio relatore prof. Mauro Brunato per la disponibilità dimostrata, e per avermi suggerito il tema, oggetto di questa tesi.

Ringrazio il team di ricerca 3DOM per avermi dato la possibilità di svolgere il tirocinio ed elaborare i diversi argomenti che sono stati trattati in questa tesi. In particolare ringrazio il prof. Fabio Remondino e Daniele Morabito, per il costante supporto offertomi.

Intendo ringraziare la mia famiglia, per avermi supportato e sopportato in questi anni e per avermi dato la possibilità di arrivare a questo traguardo.

Inoltre ringrazio i miei amici per il costante sostegno, in particolare ringrazio Federica e Sara.

Infine ringrazio me stessa per non aver mai mollato in questi anni, specialmente in quest'ultimo che è stato il più difficile dalla mia vita.

INDICE

1 Analisi di 3dfier.....6
1.1 File YAML.....7
1.2 File LAS.....8
1.3 File OBJ.....8
2 3Dfier.....9
2.1 Trasformazione file di input.....9
2.2 Primo utilizzo di 3dfier.....10
2.2.1 Creazione del file LAS.....10
2.3 Secondo utilizzo di 3dfier.....10
2.3.1 Modifica file LAS.....11
2.4 Terzo utilizzo di 3Dfier.....11
2.5 Utilizzo di 3Dfier con edificato e terreno.....12
3 Modifica di 3Dfier.....14
4 Conclusioni.....16

Sommario

Lo scopo del progetto di tesi è quello di ricostruire in 3D gli edifici della città di Trento utilizzando nuvole di punti. Per svolgere questo è stato impiegato 3Dfier, un programma opensource disponibile su GitHub.

In un primo momento è stato analizzato il codice sorgente in modo da capire il funzionamento di 3Dfier, che consiste nel prendere in input dei datasets topografici 2D e renderli tridimensionali alzando ogni poligono. L'altezza dei punti di ogni poligono viene data da una nuvola di punti.

Successivamente si è passati all'utilizzo vero e proprio del programma. Il programma è stato utilizzato più volte con file di input diversi, in modo da raffinare il risultato finale fino ad ottenerne uno realistico.
In un primo momento è stato utilizzato con un file di input di piccole dimensioni, questo per facilitarne l'utilizzo e per analizzare meglio le problematiche riscontrate.
Una problematica riscontrata è che 3Dfier solleva gli edifici attraverso la lettura della nuvola di punti, ma non tiene conto che l'altezza contenuta nella nuvola di punti è la quota dell'edificio più l'altezza dello stesso. Per risolvere questo problema è stato scritto uno script esterno in Python.
Dall'analisi di 3Dfier è emerso che questo funziona correttamente solo quando ha come input un datasets completo, cioè i file rappresentanti gli edifici, il terreno, le strade, l'acqua, le foreste e i ponti.
Per un ottenere un risultato corretto, ogni poligono dell'input di 3Dfier deve avere degli elementi adiacenti, che siano essi terreno, strade, edifici o altro non è importante, in quanto in mancanza di questi non vengono create le pareti verticali. Perciò è stato deciso di modificare il codice sorgente in modo da forzare 3Dfier ad alzare i muri verticali in ogni caso, anche in assenza di poligoni adiacenti. La versione di 3Dfier inizialmente è stata testata su un singolo poligono per vedere più rapidamente se il risultato era soddisfacente. Dopo aver appreso questo, è stata testata prima sulla stessa porzione di input iniziale e successivamente su gran parte del centro di Trento. Con queste modifiche è stato raggiunto l'obbiettivo finale inizialmente prefissato, cioè di ricostruire in 3D solamente gli edifici senza altri elementi di una città.

Il risultato ottenuto prima della modifica di 3Dfier, utilizzando solamente gli edifici, non era per niente soddisfacente in quando non strutturava nessun edificio, tranne quelli che avevano elementi adiacenti. Per migliorarlo 3Dfier è stato eseguito sia con i poligoni rappresentanti il terreno che gli edifici, anche se questo si discostava dall'idea iniziale di ricostruire solamente gli edifici, però il risultato sicuramente è migliore ma non ancora realistico. Per questo motivo è stato deciso, come detto precedentemente, di modificare il codice sorgente. Dopo questa modifica il risultato è soddisfacente e per verificare questo è stato inoltre confrontato con quello ottenuto utilizzando un plugin messo a disposizione dal software Qgis.
Con le modifiche effettuate, è possibile utilizzare 3Dfier sia per ricostruire solamente gli edifici ma anche per rappresentare la mappatura completa di una città.

Gli obbiettivi principali di questa tesi sono:

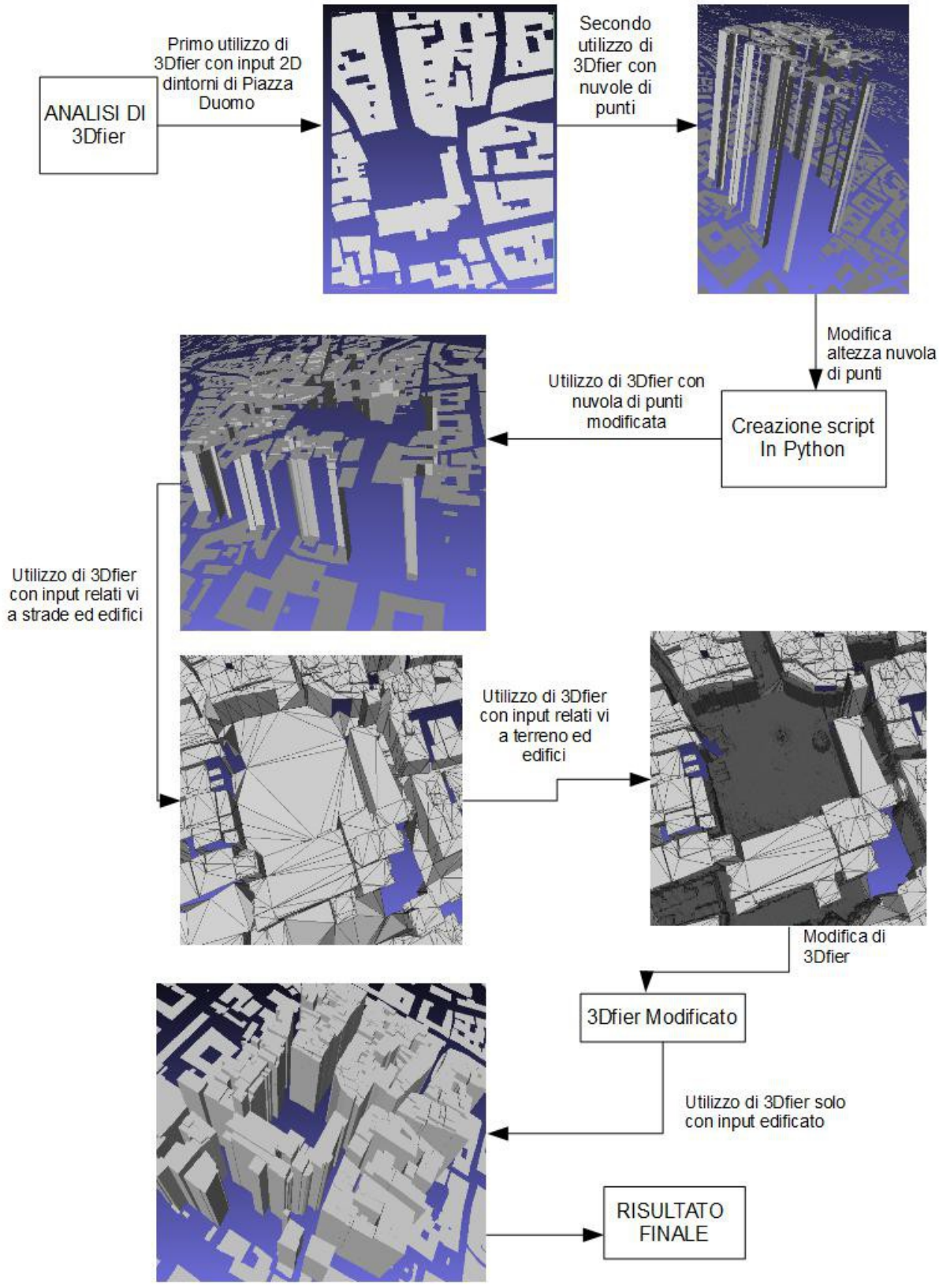
- l'analisi del codice sorgente di 3Dfier in modo da capirne il funzionamento
- l'utilizzo di 3Dfier per da ricostruire in 3D gli edifici, ottenendo un risultato il più realistico possibile

Quando si è rivelato necessario modificare 3Dfier, questo è stato fatto mantenendone tutte le funzionalità e implementandone di nuove. In quando dopo le modifiche apportare 3Dfier funziona correttamente sia con un datasets completo, come di default, inoltre funziona anche con un datasets di input rappresentante solamente gli edifici.

Una possibile modifica futura è la possibilità di implementare una funzionalità che permette di creare un file di output utilizzabile con i vari programmi GIS esistenti, in modo da dare la possibilità di utilizzare l'oggetto anche all'utente finale che non ha dimestichezza con i programmi per la visualizzazione 3D.

Per visualizzare il risultato finale inoltre può essere realizzata un'applicazione, implementando una

funzionalità in modo che se viene selezionato un edificio, vengono elencare tutte le caratteristiche di quest'ultimo.
Lo schema del lavoro svolto nel corso del tirocinio e oggetto di questa tesi è quello che emerge dallo schema sottostante:



1 Analisi di 3dfier

Per realizzare il progetto di tesi è stato deciso di utilizzare il programma opensource 3dfier [1], che è scritto nel linguaggio C++. 3dfier prende in input dati topografici 2D e li rende tridimensionali alzando ogni poligono presente. In questo software ogni poligono per essere rappresentato viene triangolato mediante la triangolazione di Delaunay [7], che è una funzione geometrica che consente di definire una griglia di triangoli in una superficie in 2D o in 3D, dato un insieme di punti P. La griglia viene costruita in modo che, per ogni circonferenza circoscritta ad un triangolo, nessun punto di P (oltre a quelli che formano il triangolo stesso) giace all'interno della circonferenza, come si vede dall'esempio in figura 1.

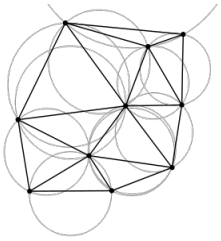


Figura 1. Un esempio di triangolazione di Delaunay

Prima di compilare ed eseguire il programma è stato necessario installare delle librerie specifiche:

- GDAL (Geospatial Data Abstraction Library) [8]: una libreria per leggere e scrivere dati geografici in formato raster o vettoriale;
- Boost library[3]: un insieme di librerie che estendono le funzionalità del C++;
- CGAL (Computational Geometry Algorithms Library) [4]: una libreria che semplifica l'utilizzo di algoritmi geometrici efficienti;
- yaml-cpp: una libreria utilizzata per definire i file YAML specifica per il C++;
- LIBLAS [10]: una libreria per il C++ utilizzata per la lettura e la scrittura di file LAS
- LASZIP [11]: trasforma rapidamente file LAS ingombranti in file LAZ compatti senza perdita di informazioni

Il programma e strutturato in file header (.h) e in file C++(.cpp) che contengono le implementazioni delle funzioni dichiarate nel file header. I file principali, ognuno con il suo file header, che compongono il software sono:

- Bridge: aggiunge tutti gli attributi di input relativi ai poligoni classificati come Bridge al file di output, presi tramite file Yaml, crea e solleva i relativi poligoni.
- Building: aggiunge tutti gli attributi di input relativi ai poligoni classificati come Building al file di output, presi tramite file Yaml, crea e solleva i relativi poligoni.
- Forests: aggiunge tutti gli attributi di input relativi ai poligoni classificati come Forests al file di output, presi tramite file Yaml, crea e solleva i relativi poligoni.
- Map3d: legge i file sqlite di input ed estrae ogni poligono classificandolo correttamente in base alla classe relativa del file YAML, legge i file LAS dei punti di input, crea e aggiorna il file di output.
- Road: aggiunge tutti gli attributi di input relativi ai poligoni classificati come Road al file di output, presi tramite file Yaml, crea e solleva i relativi poligoni.
- Terrain: aggiunge tutti gli attributi di input relativi ai poligoni classificati come Terrain al file di output, presi tramite file Yaml, crea e solleva i relativi poligoni.
- TopoFeature: attribuisce ad ogni vertice del poligono l'altezza relativa letta dal file LAS tramite una funzione apposita implementata nel file Map3d e crea muri verticali per ogni poligono che ha elementi adiacenti.
- Water: aggiunge tutti gli attributi di input relativi ai poligoni classificati come Water al file di output, presi tramite file Yaml, crea e solleva i relativi poligoni.
- GeomTools: crea i poligoni mediante triangolazione per ogni vertice, utilizzando gli algoritmi geometrici messi a disposizione dalla libreria CGAL precedentemente installata.

- Definition: definizione delle strutture utilizzate all'interno dell'intero programma. Ad esempio Triangle che è composto da tre vertici, Polygon, che definisci i poligoni, composto dall'id del poligono, dall'altezza relativa e dalla lista dei vertici da cui è composto.
- Io: contiene la funzione che mostra a video la percentuale di progressione dell'esecuzione del programma.
- Main: valida il file YAML e lancia l'esecuzione vera e propria del programma, chiamando le funzioni definite in Map3d.

Per compilare il file è stato utilizzato Cmake, che è un software opensource multipiattaforma nato per semplificare la creazione automatica dei Makefile. Il programma non ha un interfaccia grafica ma viene utilizzato da riga di comando. Le opzioni di lifting dei poligoni possono essere definite in un file YAML [20] e l'altezza è data da una nuvola di punti nel formato LAS/LAZ.

1.1 File YAML

YAML è un linguaggio per la rappresentazione delle informazioni che risulta particolarmente utile per la memorizzazione dei dati ed è facilmente leggibile per gli esseri umani.
Nel file YAML di 3dfier vengono specificate le seguenti opzioni:

- input_polygons: gruppo degli input dei poligoni 2D
  - datasets: lista dei dataset in input con specifiche caratteristiche
    - lifting: classe, che deve essere scelta tra Building, Water, Road, Separation, Bridge, Terrain, Forest, in cui i file in input devono essere mappati
    - height_field: attributo che contiene livello di altezza relativo
    - handle_multiple_heights: può avere valore True o False.
      - True → usa solo i poligoni il cui campo height_field ha valore 0
      - False → usa tutti poligoni
- lifting_options: gruppo delle opzioni di lifting per ogni classe
  - Building, Water, Road, Separation, Bridge, Terrain, Forest: definizioni delle opzioni di lifting per le varie classi
  - height_roof: percentuale di punti all'interno del raggio dei vertici degli edifici per il sollevamento dell'altezza del tetto.
  - height_floor: percentuale di punti all'interno del raggio dei vertici degli edifici per il sollevamento dell'altezza del pavimento
  - lod: definisce il livello di dettaglio LOD (Level Of Detail) che può essere 0 o 1
  - semplification: fattore di semplificazione per punti aggiunti all'interno dei poligoni che rappresentano il terreno o le foreste
- input_elevation: gruppo delle nuvole di punti
  - datasets: dataset dei file LAS
  - omit_LAS_classes: opzione per omettere le classi specificate
- options:
  - building_radius_elevation_vertex: raggio in metri utilizzato per la distanza punto-vertice tra i punti 3D e i poligoni degli edifici. Nel caso in cui non sia specificato viene utilizzato radius_elevation_vertex
  - radius_elevation_vertex: raggio in metri utilizzato per la distanza punto-vertice tra i punti 3D e i poligoni
  - threshold_jump_edges: threshold in metri per unire oggetti adiacenti, quando la differenza di altezza tra due punti è maggiore alla soglia allora viene creata una parete verticale
  - stitching: può avere valore true o false. Aggiusta l'altezza dei poligoni dopo averli uniti e dopo aver aggiunto i muri verticali
- output: gruppo dei file di output
  - format: formato dell'output che può essere OBJ, OBJ-NoID, OBJ-BUILDINGS, CSV-BUILDINGS, CityGML, CityGML-IMGeo
  - building_floor: crea o meno il pavimento degli edifici

- vertical_exageration: fattore di moltiplicazione dell'altezza

1.2 File LAS

L'altezza di ogni edificio viene letta da una nuvola di punti, nel caso specifico da un file LAS. Una nuvola di punti è un insieme di punti nello spazio individuati dal valore delle loro coordinate. Il file LAS è un file binario ed è composto principalmente da un header, che contiene tutte le informazioni necessarie ad un programma per interpretare il file, e dalla lista dei punti. Per ogni punto sono specificate 3 coordinate (x, y, z) che identificano il punto. Inoltre possono essere specificate altre caratteristiche come la classificazione. Nel caso di 3dfier i punti dei file LAS in input devono essere classificati, secondo lo standard definito da ASPRS [2], seguendo la tabella 1.
3dfier supporta le classificazioni dei punti relativi agli edifici, al terreno, alle strade, all'acqua, alla vegetazione e ai ponti. Questi punti vengono utilizzati per alzare il poligono relativo alla classe a cui corrispondono.

Classification Value	***Meaning***
0	Created, never classified
1	Unclassified[1]
2	Ground
3	Low Vegetation
4	Medium Vegetation
5	High Vegetation
6	Building
7	Low Point ("low noise")
8	High Point (typically "high noise"). Note that this value was previously used for Model Key Points. Bit 1 of the Classification Flag must now be used to indicate Model Key Points. This allows the model key point class to be preserved.
9	Water
10	Rail
11	Road Surface
12	Bridge Deck
13	Wire - Guard
14	Wire – Conductor (Phase)
15	Transmission Tower
16	Wire-structure Connector (e.g. Insulator)
17	Reserved
18-63	Reserved
64-255	User definable – The specific use of these classes should be encoded in the Classification lookup VLR.

Tabella 1. Classificazione dei punti di un file LAS

1.3 File OBJ

Un file object [13] (obj) rappresenta un formato utilizzato per definire la geometria di oggetti grafici. Nel quale possono essere elencate tutte le informazioni per la definizione di linee, poligoni etc. Nel caso di 3dfier è necessaria la rappresentazione dei soli poligoni. Come si vede dalla Figura 2, che rappresenta l'object di un quadrato, il file contiene la semplice descrizione dei vertici (v) e delle facce (f). La prima riga sta a indicare che nel punto con coordinate x = 0.0, y = 2.0 , z = 0.0 è presente un vertice. Mentre la quinta riga descrive la faccia che ha come vertici il primo, il secondo, il terzo e il quarto della lista. Il file object può essere visualizzato con qualsiasi programma per la visualizzazione 3D, tra cui MeshLab[12] e CloudCompare[5].

```
v 0.000000 2.000000 0.000000
v 0.000000 0.000000 0.000000
v 2.000000 0.000000 0.000000
v 2.000000 2.000000 0.000000
f 1 2 3 4
```

Figura 2. Rappresentazione testuale del file .obj e corrispondente grafica dello stesso file

2 3Dfier

Dopo aver analizzato, in un primo momento, il codice e il funzionamento 3dfier, si è passati all'utilizzo vero e proprio del programma utilizzando un dataset di input relativo alla città di Trento. Inizialmente per testare 3dfier, riducendo così i tempi di esecuzione, è stata adoperata solo una piccola porzione del dataset di input, cioè la parte relativa a Piazza Duomo e dintorni come si vede dalla figura 3.

Figura 3. Rappresentazione della porzione di input utilizzata per i test

2.1 Trasformazione file di input

Per utilizzare 3dfier è stato necessario trasformare il dataset, inizialmente disponibile nel formato SHP (Shapefile)[16] , in quanto i file a disposizione erano incompatibili con il formato richiesto dall'input. Lo Shapefile è un formato vettoriale molto comune per rappresentare sistemi informativi geografici. Gli Shapefile sono stati visualizzati con il software open source QGIS [15].
Il dataset di input è stato trasformato nel formato SQLITE [17] richiesto. Questo è stato fatto utilizzando il tool Ogr2Ogr [14] da riga di comando messo a disposizione dalla libreria GDAL precedentemente installata.
Utilizzando questo tool prima lo Shapefile è stato convertito nel formato GML (Geography Markup Language) [9], che è linguaggio di modellazione per sistemi geografici. Quest'ultimo, poi è stato trasformato nel file corretto in Sqlite. Per visualizzare e modificare il file Sqlite è stato utilizzato il programma DB Browser for SQLITE[18].
Il database è composto da tre tabelle:

- una contenente la definizione dei poligoni
- una contenente la descrizione delle geometrie
- una contente la definizione del sistema di riferimento.

I campi principali che devono essere presenti nella tabella di definizione dei poligoni sono fid e relatievehoogteligging (quota relativa). *Fid* è un identificatore che permette al programma di identificare univocamente il poligono. Mentre *relatievehoogteligging* è un campo che specifica l'altezza relativa ed è necessario in quanto viene richiesto nell'opzione height_field del file YAML (si veda paragrafo 1.1).
Nel file Sqlite precedentemente creato non è presente questo campo, così è stato creato utilizzando la query di inserimento

ALTER TABLE ctr_edificato ADD relatievehoogteligging INTEGER;

successivamente è stato aggiunto il valore di altezza relativo ad ogni record con una query di update

update ctr_edificato
set relatievehoogteligging = 0

2.2 Primo utilizzo di 3dfier

3dfier è stato utilizzato per una porzione di dati della città di Trento. In un primo momento è stato lanciato con un dataset di input che prevedeva il file Sqlite dell'edificato e con una nuvola di punti con altezza del punto pari a zero, solo per testare se il dataset fornito era corretto. Non ci sono stati problemi nel leggere l'input e nel disegnare i footprint ed il risultato ottenuto è quello che si vede nella figura 4.

Figura 4. Rappresentazione dei footprint

2.2.1 Creazione del file LAS

Per creare il file LAS è necessario aprire il file SHP dell'edificato tramite QGIS e esportare i dati nel formato CSV[6]. Questo poi è stato trasformato in un file di testo e successivamente è stato convertito nel formato LAS, tramite il tool Txt2Las[19] a disposizione con l'installazione della libreria LibLAS. Subito emerge che l'altezza dei punti presenti nel file non viene letta correttamente. Il problema viene risolto utilizzando un file LAS reale e non creato con questo metodo.

2.3 Secondo utilizzo di 3dfier

Dopo aver provato 3difier ed aver visto che il dataset è corretto, il programma questa volta viene eseguito con un LAS a me fornito. Una volta terminata l'esecuzione e dopo aver aperto l'output con MeshLab, in quanto è stato creato un file OBJ, si nota fin da subito che c'è un problema. Il quale consiste, come si vede dalla Figura 5, nell'altezza di ciascun edificio, perché quella contenuta nel file LAS è rappresentata come quota + altezza dell'edificio. Per risolvere il problema precedentemente riscontrato sono emerse due soluzioni possibili. La prima consiste nell'aggiungere al file SQLITE un campo contente la quota dell'edificio. Successivamente nel modificare il codice sorgente in modo che i poligoni vengano sollevati partendo dall'altezza data nel campo precedentemente creato. Mentre la seconda prevede la modifica del file LAS, sottraendo dall'altezza totale del punto la quota dell'edificio. È stato deciso di implementare quest'ultima soluzione, per cercare di evitare di modificare il codice sorgente.

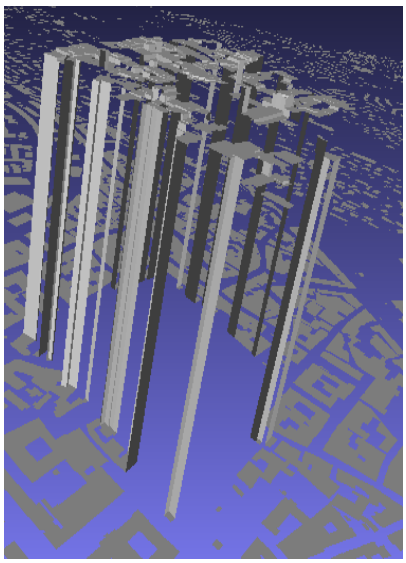


Figura 5. Esempio di visualizzazione OBJ da cui si nota il problema dell'altezza

2.3.1 Modifica file LAS

L'altezza contenuta nel file LAS, utilizzato precedentemente, è riferita al DSM (Digital Surface Model), che è la rappresentazione della superficie terrestre inclusi gli oggetti presenti su di essa. Per ottenere il file corretto, al DSM è stato sottratto il DTM (Digital Terrain Model), che rappresenta solo la superficie del territorio, in modo da ottenere l'altezza reale di ogni singolo punto.
Per ottenere il nuovo file in un primo momento è stato modificato il codice sorgente di 3dfier in modo che leggesse contemporaneamente due file LAS, uno rappresentante il DSM e uno il DTM , e li sottraesse memorizzando il valore senza la necessità di creare un nuovo file. Però nella modifica del codice è emerso un problema con la lettura contemporanea dei due file e quindi è stato deciso di creare uno script esterno, scritto in Python, da lanciare precedentemente all'utilizzo del programma vero e proprio.
Lo script legge ogni punto del DSM, cerca nel DTM quello più vicino, che si discosta al massimo di 0,5, sottrae le loro altezze e lo riscrive aggiornato in un nuovo file. Siccome si è notato che avrebbe impiegato troppo tempo, è stato ottimizzato in modo da non dover leggere tutti i punti del secondo file, immaginando che il DTM fosse rappresentato come una matrice bidimensionale, quindi viene prima ricercata la coordinata x e successivamente la y. La prima versione, nel caso peggiore, in cui l'ultimo punto del DSM corrisponde all'ultimo del DTM, ha complessità O(nm), con n che corrisponde al numero di punti presenti nel DSM mentre m a quelli del DTM. Invece nella versione ottimizzata ha complessità O(l(n+m)), cioè l'algoritmo per la ricerca in una matrice che ha complessità O(n+m), con n numero di righe e m numero di colonne, viene ripetuto tante volte quanti sono i punti presenti nel primo file.

2.4 Terzo utilizzo di 3Dfier

Dopo la creazione del nuovo file LAS, 3Dfier è stato testato con il nuovo file LAS ed il risultato ottenuto si vede nella figura 6. Dalla visualizzazione dell'output si nota che l'altezza è corretta mentre molti muri verticali dei poligoni non vengono creati. Dopo un analisi del codice si nota che i muri verticali vengono creati solo per i poligoni che hanno oggetti adiacenti. Non crea tutti i muri verticali in quanto non ha riferimenti dell'altezza se non esiste nessun poligono adiacente e quindi non riesce a confrontare la differenza delle due quote con la threshold definita nel file YAML.

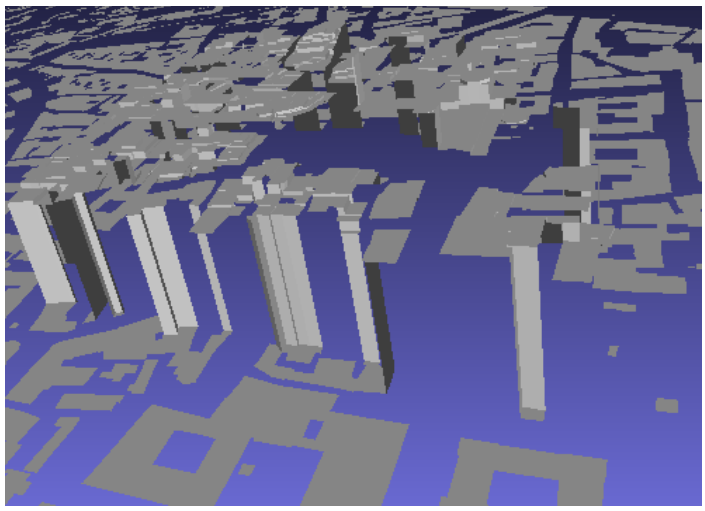


Figura 6. Risultato con LAS modificato

Dopo questi risultati ottenuti è stato modificato anche il file YAML in modo da verificare se si poteva migliorare il risultato. Alcune combinazioni effettuate sono quelle presenti nella tabella 2. Sono stati modificati principalmente i campi relativi alle opzioni: building_radius_elevation_vertex, radius_elevation_vertex e threshold. Le modifiche di questi parametri non hanno portato miglioramenti significativi. Quindi per ottenere un risultato migliore è stato utilizzato 3Dfier con due file Sqlite di input, uno rappresentante i poligoni delle strade mentre uno i poligoni dell'edificato.

Height_roof	Height_floor	Building_radius_elevation_vertex	Threshold	Radius_elevation_vertex
95	5	0,2	1	0,5
95	5	0,5	0,5	0,1
95	5	1	1	1
95	5	3	0,5	1
95	5	3	6	1
95	5	10	1	1
95	5	20	1	1
90	10	0,2	1	0,5
90	10	0,5	0,5	0,1
90	10	1	1	1
90	10	3	0,5	1
90	10	3	6	1
90	10	10	1	1
90	10	20	1	1
85	15	0,5	0,5	0,1
85	15	1	1	1
85	15	3	0,5	1
85	15	3	6	1
85	15	10	1	1
85	15	20	1	1

Tabella 2. Tabella contenente le modifiche effettuate al file YAML

2.5 Utilizzo di 3Dfier con edificato e terreno

Siccome 3Dfier crea dei muri verticali solo per i poligoni che hanno oggetti adiacenti, è stato utilizzato con due file di input 2D, uno che rappresenta l'edificato e uno la strada. Il primo risultato ottenuto è quello che si vede in figura 7 a sinistra. Per risolvere il problema della "voragine" creata al centro della figura 7 a sinistra,

è stata modificato il file YAML portando il building_radius_elevation_vertex a 12 in modo da avere una densità di punti maggiore all'interno del poligono (come si nota dalla figura 7 a destra).

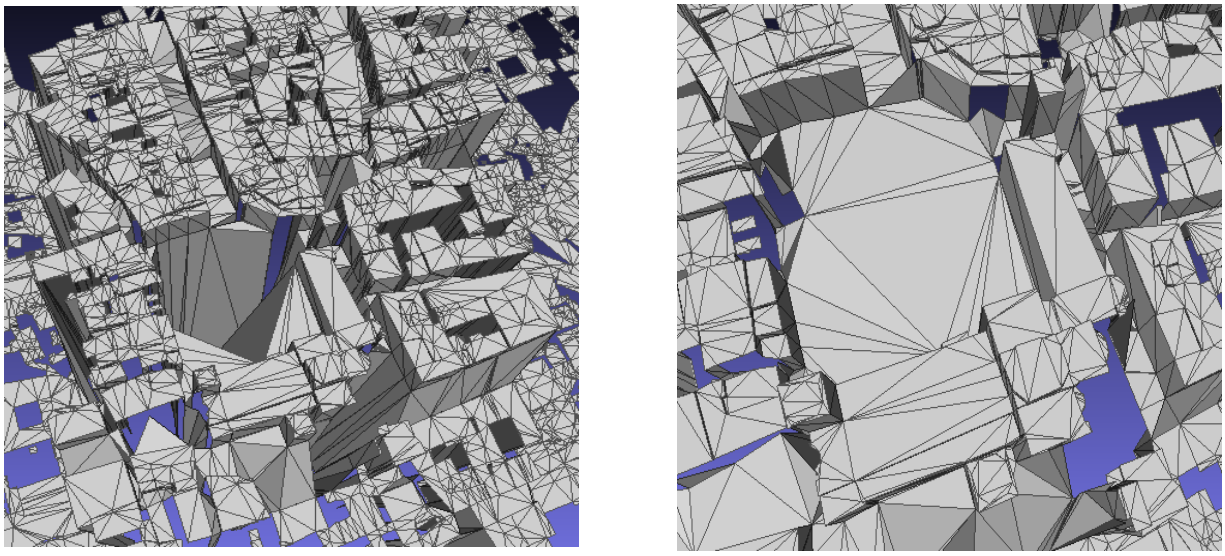


Figura 7. Rappresentazione output edificato a sinistra e a destra successiva rappresentazione modificando il file YAML

Per migliorare il risultato, i punti del file LAS sono stati classificati direttamente al momento della lettura.
Se il punto appartiene ad un poligono che rappresenta un edificio viene classificato come Building con valore 6 (si veda Tabella 1) utilizzando la funzione SetClassification() messa a disposizione dalla libreria LibLas. Mentre se appartiene al terreno viene classificato come Terrain con valore 2. Dopo queste modifiche il risultato ottenuto è quello che si vede dalla figura 8. Come si nota dalla figura 7, tutti i poligoni vengono rappresentati mediante triangolazione anche se sono identificati come strada. Questo perché i punti del LAS relativi non sono classificati. Mentre dalla figura 8 emerge che il terreno e gli edifici vengono considerati separatamente in quanto le caratteristiche vengono specificate da punti opportunamente classificati. Il risultato è sicuramente migliore ma non ancora soddisfacente perché non crea ancora tutti i muri verticali. Così è stato deciso di modificare il codice sorgente in modo da forzare la creazione dei muri in qualsiasi caso, anche in assenza di poligoni adiacenti.

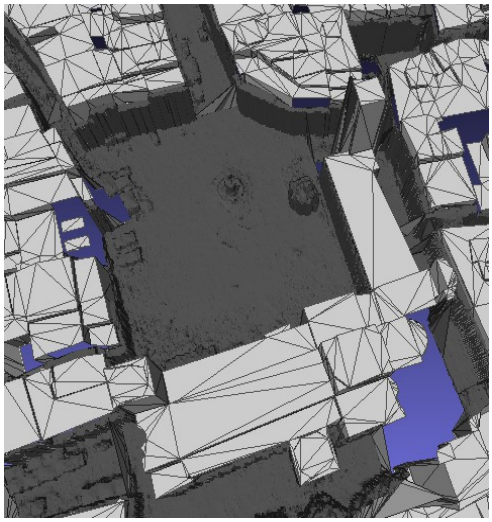


Figura 8. Rappresentazione output, edificato con terreno, con punti classificati

3 Modifica di 3Dfier

Per modificare 3Dfier in modo da forzarlo ad alzare i muri, sono state create delle nuove funzioni e sono state modificate alcune di quelle già esistenti. Inizialmente le modifiche effettuate per arrivare alla soluzione definitiva sono state testate con la creazioni di un singolo edificio. È stato studiato come vengono definiti i triangoli mediante triangolazione di ciascun edificio e come vengono descritti nel file Object (si veda paragrafo 1.3). Per modificare il programma è stato pensato di creare un solido per ogni edificio presente. Nella funzione già definita nel file obj vengono scritte le descrizioni dei vertici di ciascun poligono, con la relativa altezza e la descrizione delle facce realizzate mediante triangolazione, mente in quella nuova vengono aggiunti all'object i vertici con altezza zero in modo da creare la base del poligono come si vede dalla figura 9.

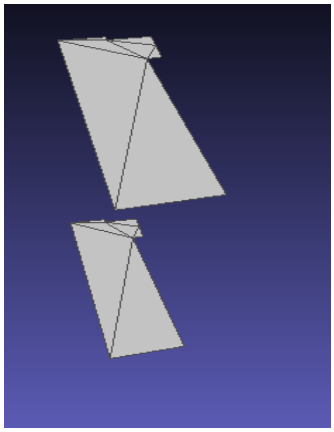


Figura 9. Poligoni triangolati che rappresentano il tetto e la base

Per alzare i muri è stata creata una funzione nuova che crea un muro verticale per ogni lato di un triangolo. Questa funzione scrive nel file Obj la descrizione di ciascuna faccia, che in questo caso viene definita dalla congiunzione dei vertici del tetto e della base. Ogni faccia viene definita da un vertice del triangolo, dal successivo vertice del segmento, e dai due vertici del segmento parallelo.
Inizialmente i vertici della base e del tetto di ciascun poligono sono stati memorizzati in una lista. Successivamente sono stati memorizzati prima tutti i vertici dei tetti di tutti i poligoni e poi quelli della base. Questo per facilitare la creazione delle facce nel file OBJ, perché prima viene indicata la posizione dei primi due vertici del segmento del tetto, poi viene indicata la posizione del segmento corrispondente della base. Il numero identificatore del vertice non è specificato esplicitamente nel file, ma viene considerato in base all'ordine di apparizione. Quindi l'identificazione dei vertici del secondo segmento deriva dalla posizione dei vertici del tetto più il numero di vertici totale. Dopo aver apportato queste modifiche 3Dfier funziona correttamente e ne è un esempio la figura 10, che mostra il risultato finale della figura 9.

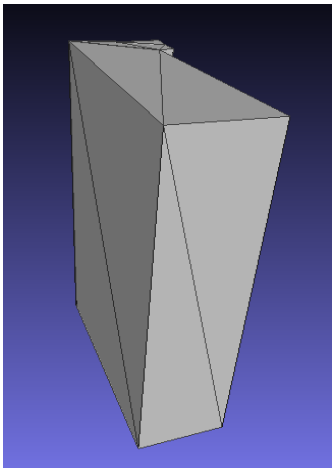


Figura 10. Risultato dell'esecuzione di 3Dfier modificato con input della figura 9

Dopo aver modificato 3Dfier ed aver visto che esegue correttamente e che i muri vengono creati, è stato eseguito con l'input relativo ai dintorni di Piazza Duomo. Per terminare l'esecuzione sono stati impiegati circa 4 minuti ed il risultato ottenuto è quello che si vede nella figura 11.

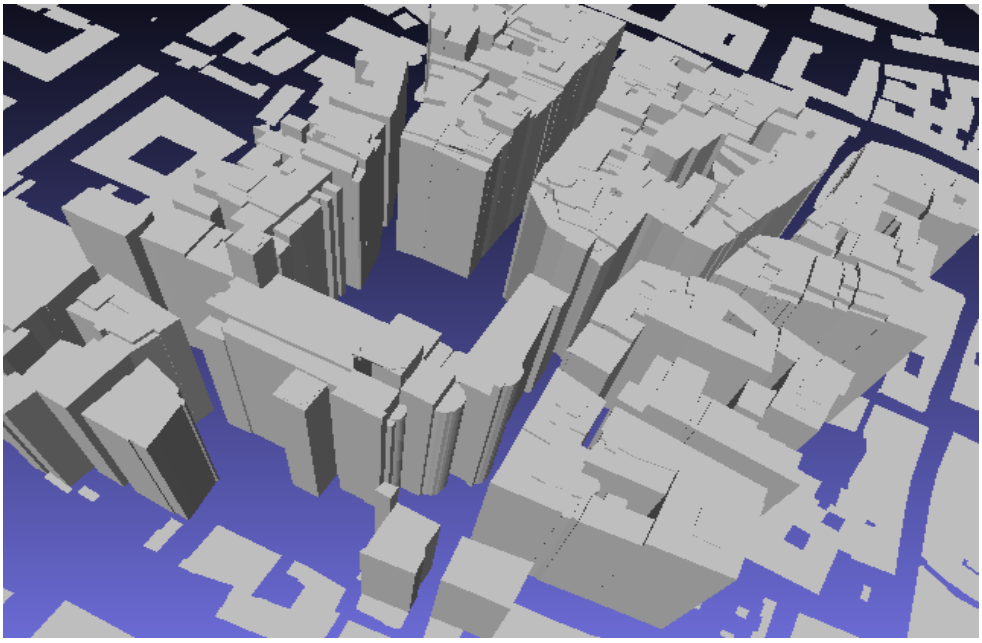


Figura 11. Risultato Obj dei dintorni di Piazza Duomo, realizzato con 3Dfier modificato

Per avere un confronto e per determinare se il risultato è corretto e verosimile è stato utilizzato il plugin threeJS messo a disposizione da Qgis per realizzare una mappa 3D. L'output è un file visualizzabile tramite Browser. Nella specifica delle opzioni è possibile sia scegliere di leggere l'altezza da un campo apposito dello shapefile stesso, oppure definirne una di default uguale per tutti gli edifici. Il risultato ottenuto è quello che si vede in figura 12. Come si nota dal confronto tra la figura 11, che rappresenta il risultato ottenuto con 3Dfier, e la figura 12, ottenuta con Qgis, entrambe riguardanti la porzione di edificato di Piazza Duomo, non ci sono differenze a parte l'altezza. La differenza di altezza deriva dal fatto che nel primo caso viene letta dal file LAS mentre nel secondo caso viene letta da un campo dello shapefile.

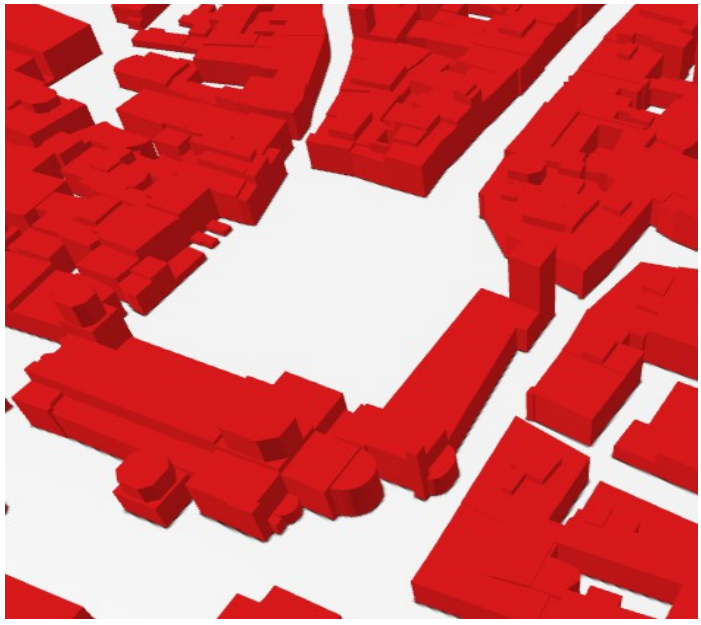


Figura 12. Risultato ottenuto utilizzando il plugin threeJS messo a disposizione da Qgis

Inoltre per confermare ulteriormente la validità delle modifiche effettuate, 3Dfier è stato provato utilizzando sia gli edifici che le strade in modo da avere una visione completa e più realistica della città di Trento. Per terminare l'esecuzione sono stati impiegati circa 8 minuti.

4 Conclusioni

Per terminare il progetto, questo programma è stato testato su una porzione di territorio della città di Trento molto più grande dell'input inizialmente utilizzato. Questo per verificare se 3Dfier con le modifiche apportate è utilizzabile anche con grandi dataset di input. Per testarlo è stato utilizzato un computer dell'ufficio di 3DOM, che è più potente e performante di quello utilizzato in precedenza con dataset di piccole dimensioni.

L'utilizzo di 3Dfier senza apportare modifiche ha dato risultati soddisfacenti solo nel caso del dataset di input di default fornito da 3Dfier. In quanto ogni poligono aveva degli adiacenti e quindi il risultato finale ottenuto è verosimile e per ogni poligono venivano creati i muri verticali necessari. Invece con l'input a me fornito l'utilizzo non è mai stato soddisfacente fino alla modifica effettuata.
Questa modifica si è rivelata molto utile in quanto è possibile utilizzare 3Dfier anche solo per una specifica categoria se non si vuole utilizzarlo per realizzare la mappatura completa. Infatti nel nostro caso è stato testato in particolare per la realizzazione degli edifici, ma funziona correttamente anche se a questi vengono aggiunti dei poligoni che rappresentano altre categorie (strade, terreno, vegetazione, etc). L'output che si vede nella prima immagine della figura 13 è stato realizzato con 3Dfier utilizzando sia terreno che edifici mentre il quello della seconda immagine della figura 13 è ottenuto con 3Dfier modificato e dando in input solo l'edificato.

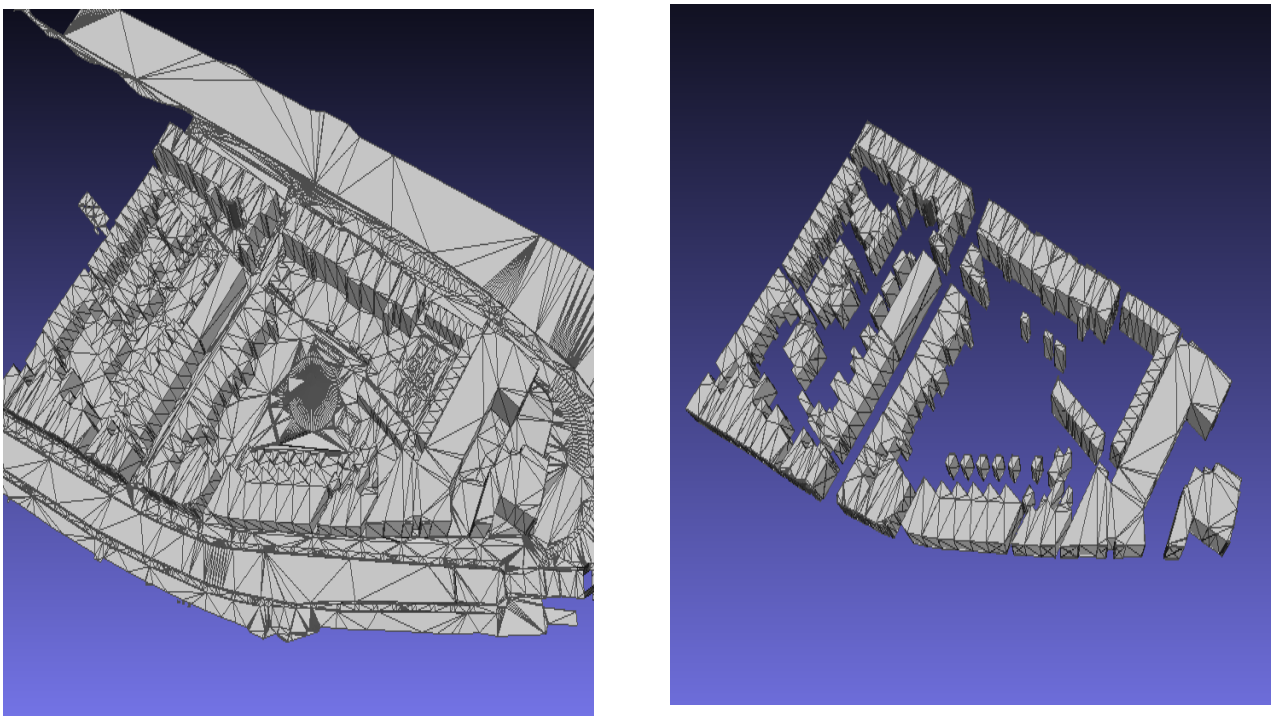


Figura 13. Output ottenuto utilizzando 3Dfier, nella prima immagine con dataset di input completo, mentre nella seconda utilizzando solo gli edifici

Per quanto riguarda le prestazioni di 3Dfier, una gran parte del tempo di esecuzione è impiegato nella funzione più consistente che è *construct_vertical_walls*, che consiste nella creazione di muri verticali. Questo perché per ogni poligono, trova un segmento, composto dal vertice a e dal vertice b, che corrisponde ad un lato dello stesso e successivamente cerca il poligono adiacente al segmento ab. Se la differenza di

altezza tra i vertici del primo segmento e quelli del secondo è maggiore della threshold, impostata tramite file YAML, allora viene costruito un muro verticale.

Questa funzione, nel caso di 3Dfier non modificato, viene chiamata comunque per ogni poligono anche in assenza di elementi adiacenti in quanto, 3Dfier presume che ogni poligono con altezza diversa da zero , abbia dei muri verticali. Questo per dire che, anche dopo le modifiche, 3Dfier impiega lo stesso tempo di prima per creare i muri verticali in quanto utilizza la stessa identica funzione con solo una operazione di scrittura, nel file OBJ, in più.

Come si nota nella tabella 3, non c'è differenza di tempo di esecuzione tra la versione di 3Dfier modificata e non, in quanto nella nuova versione è stata modificata la funzione già esistente. Mente si nota un raddoppio del tempo nel caso in cui siano stati utilizzati sia il terreno che l'edificato in quanto, molti poligoni hanno elementi adiacenti vengono creati muri verticali per questi.

Input	Tempo di esecuzione in minuti
3Dfier non modificato utilizzato con edificato	~4:00
3Dfier non modificato utilizzato con edificato e terreno	~8:00
3Dfier modificato utilizzato con edificato	~4:00

Tabella 3. Tempi di esecuzione di 3Dfier

In conclusione, per arrivare al risultato finale soddisfacente, prima è stato utilizzato 3Dfier solo con gli edifici modificando il file YAML per avere un risultato migliore. Siccome il risultato non era ancora realistico è stato utilizzato con i file di input, uno relativo al terreno e uno relativo all'edificato, anche se questo si discosta dall'obbiettivo inizialmente prefissato. Successivamente per ottenere il risultato finale con il sollevamento dei singoli edifici è stato modificato il codice sorgente in modo da forzare la creazione dei muri.

Inizialmente, per quanto possibile, si è cercato di non modificare il codice sorgente. Infatti per risolvere il problema dell'eccessiva altezza è stato creato uno script esterno. Però per ottenere un risultato soddisfacente non c'era altra possibilità che modificare il codice.

Le maggiori problematiche emerse nel corso dello sviluppo del progetto riguardano l'altezza dei poligoni, che inizialmente era considerata come quota più altezza effettiva dell'edificio. Questo è stato risolto creando uno script esterno per sottrarre all'altezza del DSM quella del DTM.
Un altra problematica evidenziata è nel funzionamento di 3Dfier, perché crea i muri verticali solo nel caso in cui il poligono abbia elementi adiacenti. Questo è stato risolto, per rimanere fedeli all'obbiettivo iniziale, modificando 3Dfier.
Una possibile modifica futura è la possibilità di implementare una funzionalità che permette di creare un file di output utilizzabile con i vari programmi GIS esistenti, in modo da dare la possibilità di utilizzare l'oggetto anche all'utente finale che non ha dimestichezza con i programmi per la visualizzazione 3D.
Inoltre con il file OBJ ottenuto come risultato è possibile visualizzarlo attraverso la realizzazione di un'applicazione, implementando una funzionalità in modo che se viene selezionato un edificio, vengono elencate tutte le caratteristiche di quell'edificio.

Bibliografia

[1] 3dfier (https://github.com/tudelft3d/3dfier
[2] ASPRS (https://www.asprs.org/)
[3] BOOST library (http://www.boost.org/)
[4] CGAL library (http://www.cgal.org/)
[5] CLOUDCOMPARE (http://www.danielgm.net/cc/)
[6] CSV (https://en.wikipedia.org/wiki/Comma-separated_values)
[7] Delaunay Triangulation (https://en.wikipedia.org/wiki/Delaunay_triangulation)
[8] GDAL library (http://wwwd.gdal.org/)
[9] GML (https://en.wikipedia.org/wiki/Geography_Markup_Language)
[10] LAS library (http://www.liblas.org/)
[11] LASZIP(http://www.laszip.org/)
[12] MESHLAB (http://www.meshlab.net/)
[13] OBJ (https://en.wikipedia.org/wiki/Wavefront_.obj_file)
[14] OGR2OGR (http://www.gdal.org/ogr2ogr.html)
[15] QGIS (http://www.qgis.org/it/site/)
[16] SHP (https://en.wikipedia.org/wiki/Shapefile)
[17] SQLITE (https://www.sqlite.org/)
[18] SQLITE BROWSER (*http://sqlitebrowser.org/*)
[19] TXT2LAS (http://www.liblas.org/utilities/txt2las.html)
[20]YAML (http://yaml.org/)