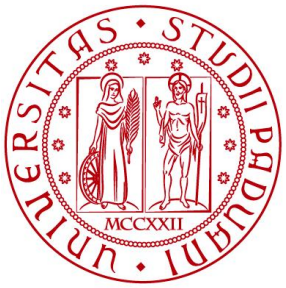


UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE - DEI

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

Identificazione automatica dei cambiamenti nel territorio e supporto alla restituzione degli oggetti con l'utilizzo di dati LiDar e deep learning

Relatore: **Prof. Rumor Massimo**

Laureando: **Esposito Giuseppe**

ANNO ACCADEMICO 2021-2022

Data di laurea 22-02-2022

Abstract

L'obiettivo principale di questo lavoro di tesi è di definire un metodo che consenta, a partire da dati ottenuti da remote sensing, di rilevare aree del territorio dove sono avvenuti cambiamenti, riducendo così l'impiego umano richiesto nella fase di aggiornamento dei dati territoriali.Un metodo è stato individuato e verificato nelle sue fasi principali utilizzando nuvole di punti ottenute da MMS(Mobile mapping system).Le verifiche seppur parziali supportano la validità dell'approccio. Si sono inoltre identificati metodi per il miglioramento dei risultati introducendo l'utilizzo della conoscenza a-priori dei luoghi interessati e si sono analizzate le possibili soluzioni disponibili per la fase di restituzione, utilizzabili per supportare la fase vera e propria di aggiornamento dei dati delle aree di cambiamento identificate.

Indice

1 Introduzione

Un problema fondamentale dell'informazione geografica è l'aggiornamento dei dati che richiede interventi manuali e quindi implica tempi e costi notevoli. Negli ultimi anni si sono rese disponibili a costi contenuti nuvole di punti ottenute con tecnologia Lidar sia da sensori aviotrasportati che da sensori montati su veicoli (sistemi i MMS) territoriali.Tali nuvole forniscono un modello digitale del territorio e degli oggetti ivi contenuti(Digital Surface Model).

L'obiettivo principale del lavoro di tesi è quello di identificare e delimitare le aree ove sono avvenuti cambiamenti utilizzando nuvole di punti acquisite una all'inizio ed una alla fine del periodo considerato. Questo consente di eliminare una onerosa fase manuale che consiste proprio nell'individuare visivamente i cambiamenti ottenuti e che richiede l'esame di tutto il territorio in aggiornamanto. Inoltre, si cercherà di perseguire un secondo obiettivo cioè verificare i la possibilità, per le aree di cambiamento rilevate, di offrire un supporto alla fase vera di aggiornamento dei dati attraverso l'impiego dei metodi di restituzione automatica che oggi si sono resi disponibili.Tali nuovi metodi si basano sul Deep Lerning e forniscono risultaii interessanti.

Il lavoro di tesi prevede la definizione di un processo che consenta di delimitare le aere di aggiornamento e di verificare sperimentalmente, con dati reali, il funzionamento delle fasi principali. Prima di iniziare l'attività di definizione del processo si è effettuata una analisi dei metodi di change detection presenti in letteratura e delle sperimentazioni effettuate rilevando come:

- non siano presenti applicazioni di change detection che utilizzino dati lidar terrestri
- nessuna applicazione abbia considerato la possibilità di utilizzare la conoscenza a priori costituita dai dati preesistenti.

La tesi è articolata in 4 capitoli. Nel primo capitolo viene fornita un'introduzione al change detection e ai diversi task di Deep Learning, come classificazione o restituzione automatica, che verranno utilizzati nel nostro lavoro. Nel secondo capitolo viene dettagliatamente spiegato il metodo proposto.Nel terzo capitolo si descrivono la sperimentazione condotta sul metodo, le verifiche sperimentali effettuate ed i risultati ottenuti da quest'ultime utili a valutare il metodo realizzato. Nell'ultimo capitolo, relativo le conclusione, si riassumono, invece, gli obietti raggiunti ed i limiti presenti, si delinano infine gli sviluppi possibili.

2 Introduzione al Change Detection, alla classificazione di nuvole di punti e immagini e alla restituzione automatica

Nei paragrafi successivi verrà presentato il *Change Detection* attraverso un excursus su cosa è, sugli utilizzi, e sui limiti che si possono presentare durante l'applicazione del processo. Successivamente, verranno descritti due argomenti intrinseci al metodo proposto e agli obiettivi di tesi fissati, la *restituzione automatica 3D* e la *classificazione automatica di immagini/modelli*.La decisione di trattere e descrivere già in questo capitolo i due argomenti sopra citati è legata da un lato a migliorare e facilitare l'esposizione per il lettore, dall'altro all'essenzialità che i due processi avranno nel metodo proposto.

2.1 Change Detection

Definiamo il *Change Detection* come il processo di rilevamento automatico di modifiche dei dati spaziali utilizzando dati ottenuti attraverso remore sensing o più semplicemente Il *Change Detection* non è altro che una branca del remote sensing che si occupa di rilevare i cambiamenti, avvenuti col passare del tempo, in determinate zone geografiche. Tale meccanismo, oggi giorno, è applicato in diversi contesti pratici ed utilizzato al fine di:

- tracciare le condizioni ambientali
- tracciare l'espansione urbana
- rilevare dei cambiamenti nella vegetazione
- valutare la desertificazione
- rilevare i cambiamenti urbani o naturali nell'ambiente

Il processo di *Change Detection* può essere classificato in base al (1)tipo di metodo/algoritmo utilizzato per ottenere i risultati ed in base ai (2)tipi di dati in input utilizzati. Tralasciando il tipo di metodo/algoritmo utilizzato che può essere intrinseco alle necessità o al tipo di applicazione, è invece opportuno precisare che solitamente i metodi di *Change Detection* sviluppati e presenti in letteratura lavorano su dati in input ottenuti da satellite, mentre nel metodo che proporremo nel capitolo successivo verranno utilizzati solamente dati ottenuti da sistemi aerei o da sistemi terrestri (Mobile Mapping System(MMS)), come nuvole di punti e foto aeree o foto non aeree a secondo del sistema utilizzato. Un generico processo di *Change Detection* è illustrato nella Figura 1 sottostante.Esso prende in input due modelli 3D, della medesima zona di interesse in due archi temporali differenti, uno rappresentante il prima ed uno rappresentante il dopo, mediante un algoritmo di change detection basato o sul calcolo delle distanze tra i punti dei due modelli in input o sul calcolo dello scostamento in un'intorno della coordinata z dei punti dei

due modelli in input , rileva e computa i cambiamenti presenti tra il prima ed il dopo e restituisce in output un modello 3D, solitamente quello in input rappresentante il dopo, su cui sono visibili i cambiamenti avvenuti nel tempo. Tali cambiamenti sono resi visibili grazie all'associazione ad ogni punto appartenente al modello outputtato di un colore che rappresenti quanto quel punto è cambiato nel tempo.Inoltre, come intuibile, il colore che viene assegnato al punto o in generale la scala di colori che viene a crearsi dopo che l'algoritmo di change detection è stato eseguito è correlata ai valori o di distanza o di scostamento calcolati dall'algoritmo di change detection stesso e rappresentanti il cambiamento.

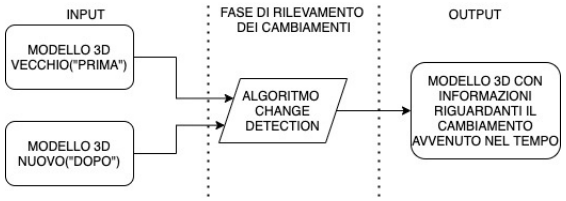


Figure 1: Processo di Change Detection

2.1.1 Limitazioni del Change Detection

Parlando di *Change Detection* è opportuno, oltre le caratteristiche viste in precedenza, accennare anche ai limiti che il processo presenta.Di seguito, quindi, vengono descritte le principali problematiche possedute dai metodi di rilevazione del cambiamento soffermandoci su tre aspetti principali:

1. accuratezza dei dati in input su cui effettuare il task di rilevamento dei cambiamenti
2. rilevazione errata di cambiamenti
3. tempo di esecuzione richiesto dal processo

Per quanto concerne il primo punto, considerando il processo di rilevamento dei cambiamenti di modelli 3D in input, dobbiamo accennare al fatto che al fine di rappresentare in modo più preciso possibile la realtà, i modelli 3D adoperati, rappresentanti un "prima" ed un "dopo" di una data area e su cui verrà effettuato il processo di rilevamento dei cambiamenti, dovranno essere rilevati con un alto grado di accuratezza.Se questo aspetto non venisse tenuto in considerazione potremmo ritrovarci ad effettuare un change detection su dati non confrontabili tra di loro, o al più, ottenere dei risultati in output falsati rispetto il reale cambiamento avvenuto nel tempo.Inoltre, altro aspetto importante correlato sempre al dato in input e quindi ai modelli 3D utilizzati è la densità di punti per volume posseduta da questi. Infatti, il processo per ottenere risultati ottimali e veritieri rispetto ai reali cambiamenti, necessita che i due modelli 3D adoperati in input posseggano lo stesso, o all'incirca simile, numero di punti per volume.Se questo valore non fosse uguale, o all'incirca

simile, ci ritroveremo ad avere, in determinate zone dei modelli considerati, una non corrispondenza tra i punti che li formano, la quale porta il processo a non poter ottenere alcuni risultati e quindi ad una perdita di informazione correlata al change detection stesso.Quest'ultimo aspetto comporterà il conseguimento di risultati, relativi al change detection, che potrebbero essere veritieri, ma che sicuramente saranno incompleti a causa della perdita di informazione precedentemente citata e quindi in parte anche errati e non veritieri rispetto il reale cambiamento avvenuto col trascorrere del tempo. Invece, per quanto concerne invece il secondo punto è possibile che il processo ritorni in output dei cambiamenti errati dovuti alla presenza di falsi positivi.Questo concetto, o piccolo argomento, verrà trattato in modo approfondito in uno dei paragrafi del capitolo 2, qui ci soffermeremo a dire che, può accadere ed accade molto spesso che nelle rilevazioni di modelli 3D di un "dopo" rispetto ad un "prima", la vegetazione presente nel modello rappresentante il "prima" sia cresciuta o sia stata tagliata, oppure siano presenti macchine non presenti nel modello 3D rappresentante la situazione "prima".Ecco, in situazioni del genere ci troveremo davanti a dei falsi positivi e se applicassimo un processo di change detection verrebbero rilevati sicuramente cambiamenti veritieri e reali, ma anche errati dovuti proprio alla presenza dei falsi positivi.Quindi è buona norma, al fine di ottenere solamente i cambiamenti reali ed opportuni, o prima, o durante o dopo l'applicazione di un processo di change detection eliminare tutti i falsi positivi. Infine, per quanto riguarda l'ultimo punto, ossia il tempo di esecuzione è stato possibile osservare che la maggior parte dei metodi oggi giorno utili a performare il task di *change detection*, soprattutto di grandi aree geografiche, richiedono abbastanza tempo di esecuzione.Questo rappresenta, sì, una limitazione importante del task, ma che comunque, oggi giorno, non spaventa in quanto il rilevamento dei cambiamenti è un processo che non verrà comunemente applicato, soprattuto per la medesima area, in lassi di tempo piccoli, ma dopo mesi o anni.Inoltre, in futuro questa limitazione verrà sicuramente ridotta grazie alla creazione di nuovi hardware o algoritmi più perfomanti.

2.2 Restituzione e Classificazione automatica 3D

Prima di addentrarci nel campo della *Restituzione automatica 3D* e della *classificazione automatica* di immagini o modelli 3D, è opportuno, essendo argomenti correlati tra di loro, accennare al Deep Learning e alle Convolutional Neural Network.

2.2.1 Deep Learning

Il Deep Learning è una sottocategoria del Machine learning e basa il proprio funzionamento sulla costruzione di una rete neurale artificiale formata da unità elementari, chiamati neuroni, organizzati in diversi strati, e uniti tra di loro medianti archi. Avremmo lo strato di input, quello di output e diversi strati nascosti ognuno dei quali sarà formato da un numero n di neuroni. Ogni neurone della rete, viene rappresentato mediante una funzione che riceve, per ogni arco in ingresso, un valore che viene pesato con un valore prestabilito sull'arco e che calcola la somma di tutti questi valori pesati in ingresso e a cui viene sommato un ulteriore valore intrinseco del nodo, chiamato bias.In uscita invece il neurone outputterà una funzione non

lineare del valore appena ottenuto.Tale funzione non lineare prende il nome di *funzione di attivazione*, viene utilizzata al fine di aiutare la rete neurale ad apprendere relazioni e schemi complessi nei dati e ne esistono di diversi tipi tutti visibili nella Figura 2 sottostante.

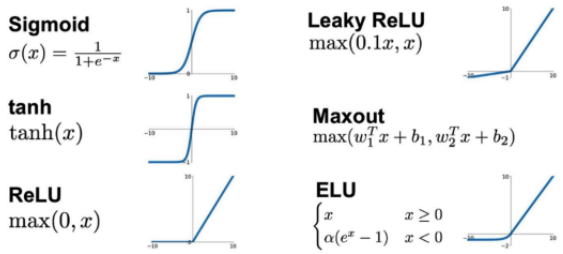


Figure 2: Principali funzioni di attivazione

Invece, nelle figure sottostanti, Figura 3 e Figura 4, sarà possibile osservare la topologia di una Deep Neural Network e le computazioni precedentemente descritte e performate dal generico neurone appartenente al generico strato per ottenere l'output.

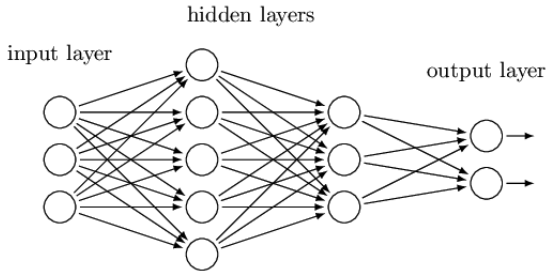


Figure 3: Topologia di una Deep Neural Network

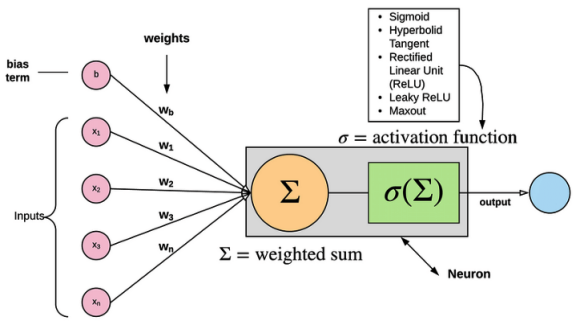


Figure 4: Computazioni effettuate dal generico neurone

Inoltre, tra le architetture di spicco del Deep Learning si annoverano le *Convolutional Neural Network*, le *deep belief network* e le *reti neurali ricorsive*, utilizzate nei campi della restituzione automatica 3D, della classificazione automatica, dell'object detection, del riconoscimento automatico del discorso, del riconoscimento audio e della bioinformatica con grandi risultati. Al fine di rimanere sempre correlati al nostro metodo e alle operazioni che verranno svolte durante l'intero processo andremo ad accennare e spiegare brevemente, nel paragrafo successivo, l'architettura delle CNN, utili soprattutto ai fini delle operazione di classificazione, ma anche di restituzione automatica.

2.2.2 Convolutional Neural Network

Una rete neurale convoluzionale(CNN) è un tipo di rete neurale di tipo feed-forward, ossia una rete dove si prevedono solamente collegamenti monodirezionali tra neuroni appartenenti a diversi strati, dove sono presenti livelli di convoluzione.Essa è costituita da uno strato di input, uno o piu strati nascosti (hidden layers) che permettono di effettuare operazioni tramite le proprie funzioni di attivazioni ed uno strato di output che effettua la classificazione o prende decisioni in merito alla predizione.Inoltre, sono presenti dei *livelli di convoluzione* i quali permettono, mediante l'utilizzo di filtri, di estrarre dal modello o dall'immagini in input diverse features, come i contorni delle figure, le linee verticali o le linee orizzontali, utili per ottenere un migliore ed accurato output.Quindi, è facilmente intuibile che in una CNN la classificazione o la predizione sono operazioni basate su particolari caratteristiche dell'input e che grazie all'elaborazioni di informazioni più specifiche i risultati ottenuti da una CNN saranno sicuramente migliori rispetto quelli di una semplice rete neurale.

Di seguito, Figura 5, mostra la topologia generale di una *Convolutional Neural Network*.

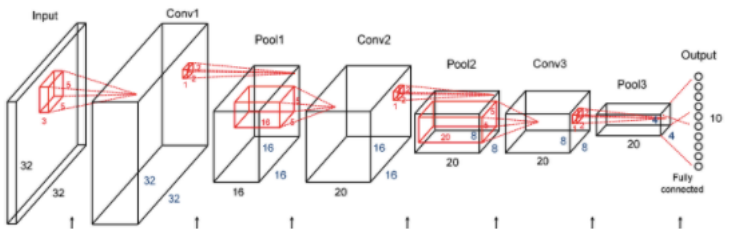


Figure 5: Convolutional Neural Network

In questa rete, ogni strato convoluzionale viene fatto seguire da uno strato di Pooling, strato che si occupa di ridurre la dimensione dell'input aumentando il livello di "astrazione" delle features che potranno essere filtrate. Nel *livello Input* della rete avremo una sequenza di neuroni che riceveranno informazioni dall'input, in questo livello verrà passato il vettore di dati che rappresentano i pixel dell'immagine o del modello in ingresso.A questo livello applicheremo un *livello Convoluzionale* che ha l'obiettivo, come già accennato in precedenza, di individuare schemi fissi, come curve, angoli, circonferenze o quadrati.Tale livello, illustrato in Figura 6, funziona

come una piccola maschera(avente dei valori) che viene fatta scorrere su tutto il dato input e per ogni posizione, mediante la computazione di un prodotto scalare tra la maschera e la porzione di input coperta, si ottiene l'opportuno output.

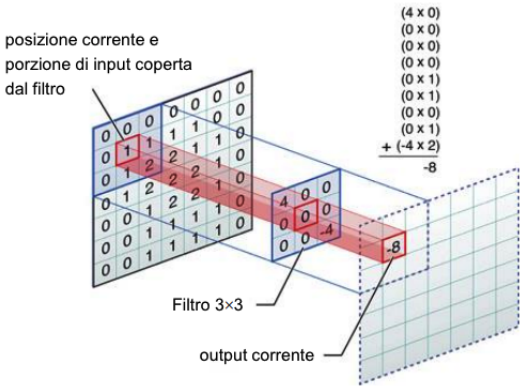


Figure 6: Livello Convoluzionale

Dopo aver applicato il livello convoluzionale, anche se non visibile dalla Figura 5, sarà presente il *livello Relu*. Codesto livello si pone l'obiettivo di eliminare tutti quei valori non utili ottenuti nei precedenti livelli.Solitamente è posto in successione ad ogni livello di convoluzione. Successivamente, all'output ottenuto dai livelli Relu applichiamo un ulteriore livello, già precedentemente citato, chiamato *livello Pooling*.Esso esegue delle aggregazioni di informazioni, generando delle feature map di dimensioni inferiori rispetto quelle in input, riducendo cosi la dimensione dell'input su cui è stato applicato. Infine, l'output ottenuto dall'ultimo livello di pooling sarà l'input per una *Fully Connected Network*.Essa formerà il *livello FC* che eseguirà la classificazione o la predizione del dato input.

2.2.3 Classificazione automatica

Il processo di *Classificazione automatica* è un processo automatizzato che dato in input un'immagine 2D o un modello 3D riesce, tramite una rete neurale, solitamente una CNN, ad ottenere la stessa immagine o lo stesso modello classificato.Solitamente, in output si ottiene la medesima immagine o il medesimo modello 3D che avevamo in input con l'unica differenza che i pixel dell'immagine o i punti che rappresentano il modello di input,che formano delle regioni di spazio e quindi l'oggetto stesso, vengono etichettati attraverso un valore(solitamente un numero intero reale) che ne evidenzi l'effettiva appartenenza ad una data classe.Questa etichettatura che rappresenta essenzialmente la classificazione può essere evidenziata sull'input attraverso una scala di colori, dove ogni colore rappresenta una classe. In generale, il processo di classificazione si presenta come mostrato nella figura successiva:

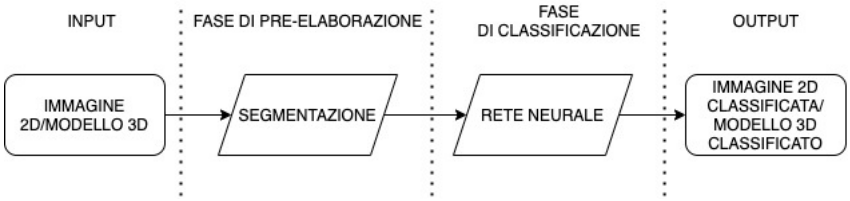


Figure 7: Processo di classificazione automatica

Come visibile dalla Figura 7, prima che l'immagine o il modello in input possa essere classificato occorre pre-elaborare i dati in nostro possesso, solitamente, applicando un operazione di segmentazione.La segmentazione permette la suddivisione dell'immagine/modello in segmenti basati su caratteristiche simili, infatti, diverse regioni dell'input possiedono valori simili di pixel/punti rispetto ad altre regioni. Questi pixel/punti verranno raggruppati in maschere di immagini/modelli che corrispondono alla forma ed ai bordi degli oggetti rilevati all'interno dell' input.Attraverso tale operazione etichetteremo in modo più accurato ogni pixel/punto e quindi anche ogni regione in cui i pixel/punti appartengono, semplificando il processo di classificazione previsto nella fase successiva. Quindi, successivamente ad aver applicato la segmentazione ed aver etichettato in modo opportuno i pixel o i punti appartenenti al modello 3D in input e le regioni in cui essi appartengono passiamo questi dati in input alla nostra rete neurale, la quale, in primis, effettua una fase di training utilizzando i dati in suo possesso e successivamente effettua la classificazione restituendo in output l'immagine o il modello classificato.

Come si è gia accennato in precedenza, una delle reti neurali più utilizzate negli ultimi anni per scopi legati alla classificazione è stata proprio la Convolutional Neural Network (CNN) che grazie all'operazione di convoluzione performata dalla rete permette di estrarre ed apprendere le caratteristiche più importanti e rilevanti per la classificazione di un oggetto, riducendo notevolmente la necessità di etichettare e segmentare in modo accurato i dati durante la fase di pre-elaborazione. Infatti, se utilizzassimo una rete CNN per effettuare il task di classificazione non avremo necessariamente bisogno di una segmentazione iniziale, in quanto la rete finale fully connected, che si occupa della computazione della classificazione, apprende già dalle caratteristiche di alto livello in input ottenute dagli strati convoluzionali profondi e finali e non necessita, per effettuare la classificazione, anche di caratteristiche a basso livello che si potrebbero ottenere con una semplice segmetazione in fase di pre-elaborazione.Inoltre, nel caso di classificazione di modelli 3D, la rete neurale, oltre alle informazioni ottenute dagli strati convoluzionali profondi e finali, per effettuare il task di classificazione, possiede ed utilizza anche altre informazioni intrinsiche alla forma/distribuzione spaziale di oggetti di interesse da classificare.Queste informazioni, che rappresentano le distribuzioni spaziali comuni di oggetti che potrebbero essere di interesse, come macchine, alberi, case, possono essere imparate dalla rete in fase di training attraverso determinati dataset.

Infine, è opportuno esplicitare che nel metodo presentato nel capitolo successivo, il task di classificazione adottato lavorerà solamente su modelli 3D.La scelta di introdurre la classificazione automatica di modelli 3D generalizzandola alla classificazione automatica di immagine è legata al facilitare la comprensione dell'argomento al let-

tore ed in quanto, come intuibile, la sostanziale differenza tra i due processi risiede solamente sul dato su cui effettuare la predizione, i pixel che formeranno gli oggetti nel caso dell'immagine ed i punti, anche essi formanti oggetti, nel caso dei modelli 3D.Inoltre, nel caso di task di classificazione con input modelli 3D è opportuno ricordare che la predizione relativa alla classificazione è effettuata dalla rete neurale utilizzando oltre le features ad alto livello ritornate dagli stati convoluzionali finali, anche altre informazioni correlate alla distibuzione spaziale comune di oggetti di interesse ed imparate dalla rete mediante training su opportuni dataset.

2.2.4 Limiti della classificazione automatica

Tralasciando quelle che possono essere le problematiche presenti nei task di classificazione automatica di immagini in quanto ampliamenti trattate in letteratura è invece opportuno visionare quelle che possono essere le problematiche correlate alla classificazione automatica di modelli 3D. Senza entrare troppo nei particolari, parlando di modelli 3D, è opportuno accennare al fatto che ogni modello, o ogni oggetto presente nel modello possiede una propria distibuzione spaziale che lo identifica in modo univoco.Attraverso questo concetto è facile intuire che gli alberi avranno una distribuzione che li accomuna, così come per le macchine o qualsiasi altro oggetto che potrebbe essere presente nell'ambiente.Quindi, la rete neurale che si occupa di classificare il modello in input, effettuerà le proprie predizioni come descritto nel paragrafo precedente tenendo anche conto della distribuzione spaziale comune di più oggetti, cioè, essa allenandosi tramite opportuni dati di training, conoscerà la distribuzione spaziale comune di un albero, così come quella di un qualsiasi oggetto di interesse e tiene conto anche di queste informazioni aggiuntive per ottenere la corretta classificazione in output. Premesso questo, possiamo affermare che le maggiori problematiche legate al task di classificazione automatica di modelli 3D sono correlate da un lato a queste informazioni in più che vengono utilizzate dalla rete neurale e dall'altro lato da come viene rilevato/ottenuto il modello 3D in input.Ovviamente, questi due concetti sono direttamente correlati tra di loro, infatti, se l'input è ottenuto da un rilievo ottimo e quindi tutti gli oggetti sono ben chiari, possiedono un grado di precisione alto allora la rete durante il suo processo di predizione sicuramente classificherà in modo opportuno tutti gli oggetti presenti, questo grazie al fatto che le distribuzioni spaziali comuni conosciute ed imparate dalla rete saranno sicuramente simili a quelli presenti sul modello 3D in input, mentre se in input trovassimo un modello 3D con un basso grado di precisione, alcuni oggetti potrebbero presentare delle distribuzioni spaziali distorte rispetto quelle reali e la rete potrebbe intaccare, anche se imparasse dalla features di alto livello ottenute dagli strati convoluzionali finali, in errori di classificazione, andando ad assegnare classi di appartenenza errate agli oggetti a causa dell'errata associazione che svolge durante il processo tra le distribuzioni spaziali distorte in input e le distribuzioni spaziali imparate e possedute.

Al fine di far capire al meglio al lettore il limite presentato precedentemente verrà esposto di seguito un piccolo esempio di caso reale che potrebbe verificarsi durante un task di classificazione di modelli 3D.Consideriamo un dato modello 3D a bassa precisione in cui internamente sono presenti macchine, strade ed alberi.Essendo un modello a bassa precisione, ammettiamo che alcune distribuzioni spaziali che

formano le macchine si presentino come distorte(i punti che formano l'oggetto non rappresentano con chiarezza la forma di una macchina), a questo punto è facile per la rete neurale, che in precedenza è stata allenata con un'opportuno dataset e quindi conosce le distribuzioni spaziali comuni di un albero, di una macchina e della strada e che successivamente imparerà anche dalle features ad alto livello ottenute dagli strati convoluzionali profondi e finali della CNN, intaccare in un'errore di classificazione, classificando parte delle macchine come alberi o strada, ma soprattuto come alberi, in quanto non riesce ad associare a quelle date distribuzioni spaziali distorte in input,che dovrebbe rappresentare delle macchine, le opportune distribuzioni spaziali comuni che possiede e che rappresentano la classe macchina.

2.2.5 Restituzione automatica 3D

La *restituzione automatica 3D* basa il suo funzionamento su una rete neurale, la maggior parte delle volte convoluzionare, che prende in input un'immagine o un modello 3D e tramite l'allenamento su un dataset di training prestabilito riesce a predire o a ricostruire in modo accurato l'opportuno modello 3D.Metodi più recenti, invece, utilizzano una rete codificato-decodificatore abbinata a reti neurali profonde per compiere il task di restituzione. In generale, il processo di restituzione automatica si presenta come illustrato in Figura 8.

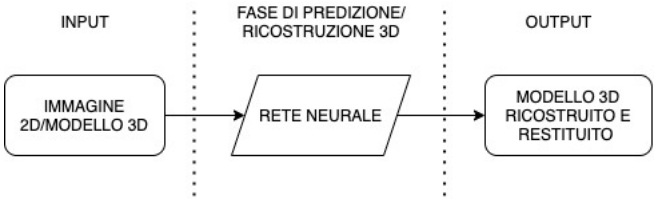


Figure 8: Processo di restituzione automatica 3D

Input

Come già accennato, in input al proccesso troviamo un'immagine 2D o un modello 3D:

- *Immagine 2D* -> solitamente, qualora utilizzassimo immagini 2D in input, faremo uso di un'immagine presa da un singolo punto di vista.Il generico oggetto presente nell'immagine o l'intera immagine dovrà essere ricostruita in 3D e restituita autamaticamente.
- *Modello 3D* -> invece,qualora utilizzassimo modelli 3D in input, il processo lavorerà su dati iniziali come cloud point, voxel, meshe poligonali(formati descritti nella sezione output) o altre strutture che presentano delle imprecisioni, delle parti mancanti o non sono accurate.Il modello verrà rimodellato/ricostruito con precisione ed infine restituito automaticamente.

Rete Neurale

E' il cuore del processo di restituzione automatica, solitamente viene impiegata una Convolutional Neural Network(CNN), ma anche altre reti neurali possono essere utilizzate.La rete neurale adoperata ha come obiettivo quello di predire/ricostruire in modo opportuno il dato input e restituirlo automaticamente in output.

Se considerassimo l'utilizzo di una CNN per completare il task di restituzione automatica potremmo affermare che la rete fully connected finale imparando dalle features ad alto livello possedute in input ed ottenute dagli strati convoluzionali finali e profondi svolgerà in modo più accurato e preciso possibile l'operazione di ricostruzione 3D e quindi di restituzione automatica.

Output

L'output del processo, come gia detto, è un modello 3D.I modelli 3D possono essere rappresentati attraverso diversi "formati",

- *Point Cloud*: chiamata anche nuvola di punti, come visibile dalla Figura 9 sottostante, rappresenta una raccolta di punti tridimensionali distribuita su uno spazio 3D.Ogni punto appartenente alla Point Cloud è caratterizzato da una posizione determininista, indicata attraverso coordinate (x, y, z), e da vari attributi. L'argomento Point Cloud verrà ampliamente trattato nel capitolo successivo, per adesso ci limiteremo ad affermare che tra i formati che verranno analizzati, grazie alla flessibilità del dato ed alle informazioni che potremmo correlare ad esso, è il più utilizzato.

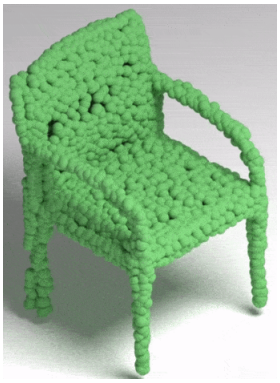


Figure 9: Rappresentazione Point Cloud

- *Voxel*:un voxel non è altro che un pixel volumetrico, cioè un pixel appartenente ad uno spazio tridimensionale.Quindi, l'opportuna unione di più voxel in una forma ben definita, come visibile in Figura 10, rappresenta un modello 3D.I voxel, oggi, sono poco utilizzati per quanto riguarda la ricostruzione a causa delle scarse rappresentazioni, dei pochi dettagli ottenibili e dell'eccessiva richiesta di risorse computazionali.

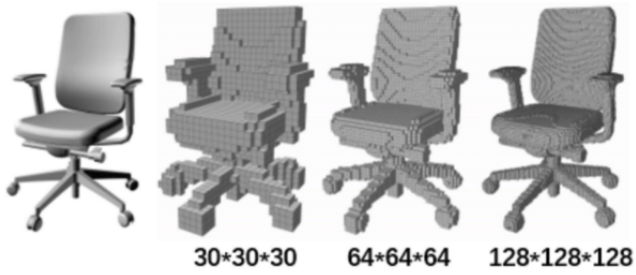


Figure 10: Rappresentazione Voxel

- *Mesh Poligonale*: rappresenta un'insieme di bordi, vertici e facce triangolari, quadrilatere o poligonali convesse che insieme delimitano la forma ed il volume di un oggetto poligonale.Un esempio di mesh poligonale è riportata nella Figura 11 sottostante, in questo formato, le facce poligonali convesse sono unite tra di loro per approssimare una superficie geometrica, ed inoltre, al fine di ottenere rappresentazioni il più realistiche possibili le mesh ottenute possono essere costituite da poligoni con fori o poligoni concavi.Questo formato, è utilizzato perlopiù per la ricostruzione e restituzione automatica di singoli/piccoli oggetti, come un volto o un'automobile, grazie alla precisione ed ai tanti dettagli che si possono ottenere, ma se volessimo ricostruire/restituire un modello 3D rappresentante grosse porzioni di superficie terresti è consigliabile utilizzare il formato PointCloud, in quanto le mesh poligonali richiedono eccessive risorse computazionali e non permettono l'associazione di informazione al dato stesso.

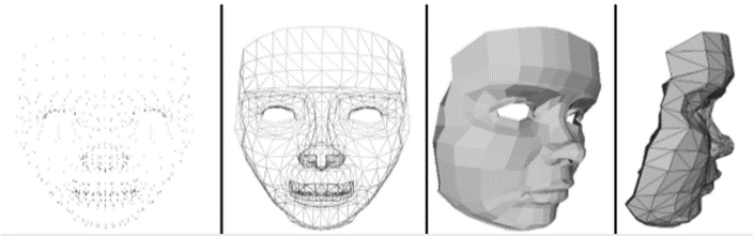


Figure 11: Rappresentazione Mesh Poligonale

- *Immagine di Profondità*:rappresenta una semplice immagine di un oggetto in 3D.Esse, saranno i formati meno utilizzati, perchè si costituiscono pur sempre una rappresentazione in tre dimensioni di un oggetto, ma non potranno essere considerate come un vero e proprio modello tridimensionale.Inoltre, tali formati, necessitano dell'operazione di discretizzazione al fine di poter conservare informazioni geometrica di media/alta qualità.Un'esempio di immagine di profondità è visibile nella Figura 12 sottostante.

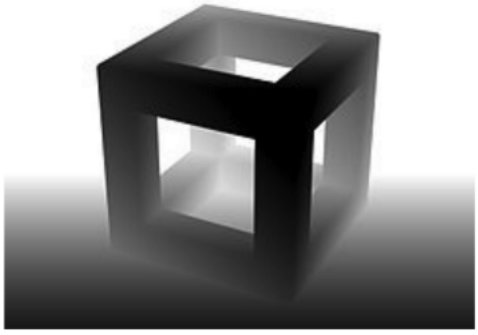


Figure 12: Rappresentazione di un'immagine di profondità

2.2.6 Limiti della restituzione automatica 3D

Le problematiche principali da affrontare, qualola si parlasse di *restituzione automatica 3D*, sono legate sia alla generica rete neurale utilizzata durante il processo che e alla tridimensionalità dei punti con cui lavoreremo, infatti, possiamo affermare che le limitazioni sostanziali sono:

1. il mancato ordinamento tra i punti
2. il labeling dei punti per la costituzione di maschere 3D

Per quanto riguarda 1. diremo che, lavorando su punti tridimensionali, potremmo ottenere un ordinamento secondo una delle coordinate, x, o y, o z, ma non sarà possibile ottenere un ordinamento ottimale rispetto la tridimensionalità dei punti stessi.Questo incide notevolmente sul processing dei punti attraverso gli usuali kernel convoluzionali dipendenti dall'ordinamento, comportando perdite di informazioni riguardanti la forma del cluster dei punti in input.Mentre, per quanto riguarda 2. affermiamo che, il labeling dei punti 3D risulta essere un problema per cui non è presente una soluzione automatizzata.Esso risulta indispensabile per la stima della qualità del modello durante le fasi di training e test della rete neurale.

Durante l'ultimo anno, il divario con le problematiche precedentemente citate è stato drasticamente ridotto grazie ad innovativi metodi e librerie per trattare il dato 3D.Infatti, anche al fine di mostrare al lettore le migliorie apportate allo stato dell'arte della restituzione automatica 3D , verranno descritti, nel capitolo successivo, quando si tratterà la fase di ricostruzione automatica, una libreria ed un metodo innovativo applicabili al nostro metodo, che possiedono ottimi risultati per il task di restituzione e che hanno ovviato, in parte, i problemi legati alla restituzione tridimensionale.

Con questo concludiamo l'excursus sul *Change detection*, sulla *classificazione automatica* e la *restituzione automatica 3D*.Abbiamo visto come tali argomenti sono legati tra di loro da un filo conduttore invisibile e si spera che il lettore abbia percepito l'intrinsicità posseduta da codesti nel metodo che verrà proposto e argomentato dettagliatamente nel capitolo successivo.

3 Metodo Proposto

Nel capitolo seguente viene illustrato e descrito il metodo di change detection proposto, soffermandoci sugli aspetti funzionali e implementativi dello stesso.

Il flowchart, in Figura 13, mostra la struttura generale del metodo e tutte le operazioni che vengono applicate al fine di ottenere aree rilevanti, della generica zone di studio, per il change detection.Come visibile il metodo proposto è basato su modelli 3D in formato nuvola di punti acquisiti tramite tecnologia LiDar.

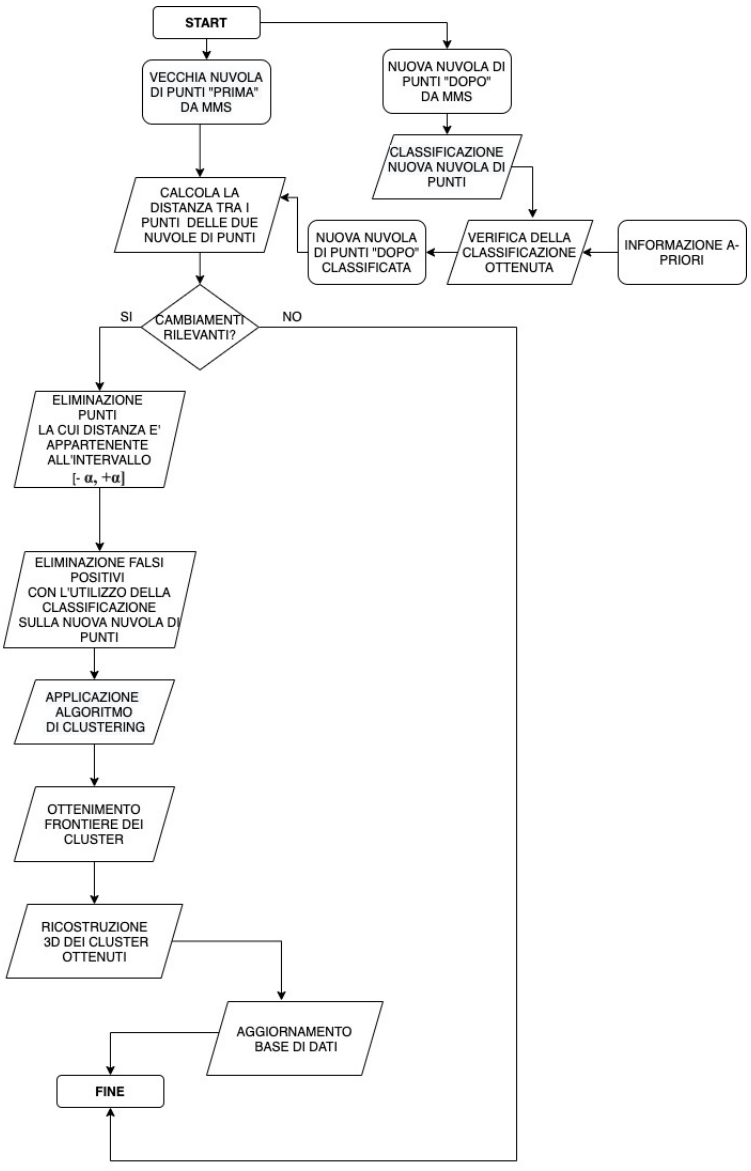


Figure 13: Struttura del metodo proposto

E' opportuno, prima di passare alla vera descrizione di ogni fase del processo, accennare al fatto che, al fine di ottenere delle determinate informazioni, per alcuni step del metodo, sono stati utilizzati dei software open source, in particolare due,

CloudCompare e AgiSoft Metashape, i quali verranno descritti successivamente e che permettono di automatizzare alcune computazioni importanti e complesse per il nostro processo di *change detection*.

3.1 Descrizione del metodo

Il metodo viene descritto partendo dall'input, passando per tutte le operazioni intermedie applicate, fino a giungere all'output dove viene ottenuto l'obiettivo fissato dal metodo, ossia il rilevamento di opportune aree di cambiamento avvenuti nel tempo per la data zona di interesse.L'output ottenuto, in fine, viene utilizzato per aggiornare i dati o la base di dati, in nostro possesso, della determinata zona di studio tramite aggiunta di nuovi modelli 3D e aggiornamento dei parametri associati ai dati per le aree di cambiamento ritrovate.

3.1.1 Input

In input il metodo prenderà tre diversi dati terresti da utilizzare:

1. modello 3D in formato nuvola di punti, rappresentante una situazione temporale, relativa alla zona di interesse, vecchia, meno recente.Faremo riferimento ad esso col termino modello "prima".Inoltre, abbiamo anche foto relative al modello considerato.

2. modello 3D in formato nuvola di punti, rappresentante la situazione temporale, relativa alla stessa zona di interessa, nuova, più recente. Faremo riferimento a questa nuvola di punti col termine "dopo".Anche per questo modello si possiedono le relative foto.

3. Database preesistente da aggiornare relativo alla zona di studio o più specificatamente contemporaneo alla nuvola di punti vecchia da cui poter estrarre, anche grazie all'osservazione e allo studio della situazione più recente della zona di interesse, l'informazioni a-priori correlabile alla zona stessa e quindi alle nuvole di punti precedenti.

Infatti, come accennato in precedenza, per effettuare il task di change detection occorre possedere due modelli rappresentanti una situazione temporale, per la data zona di studio, "prima" ed una "dopo" su cui poter computare operazioni utili proprio al rilevamento del cambiamento avvenuto nel tempo.I modelli 3D presi in considerazionione sono nuvole di punti terrestri ottenute attraverso Mobile Mapping System.Esse possiedono diverse informazioni correlate al generico dato che forma il modello stesso, tra cui coordinate, colori ed intensità, ma possono anche possederne altre.Infatti, come spiegato nel capitolo precedente, durante la trattazione dei modelli 3D, la nuvola di punti, oggigiorno, è molto utilizzata nell'ambito dei modelli 3D proprio grazie al grande numero di informazioni correlabili al dato stesso. Altre informazioni utili al processo e utilizzate in input sono le *informazioni A-priori* correlate alla zona di studio. Le informazioni A-priori rappresentano informazioni imprescindibili e non modificabili correlate all'ambiente di studio, esse sono ottenibile dall'utente che utilizza il metodo tramite conoscenza della zona di studio.La

conoscenza della zona può provenire dai dati contenuti nel database preesistente correlato alla zona di studio o dalle foto legate ai modelli che abbiamo in input o anche dai modelli 3D stessi.Vengono utilizzate nel metodo come parte essenziale al fine di ampliare notevolmente l'accuratezza dei risultati ottenuti, modificare operazioni automatizzate aventi errori o problemi ed aggiornare, alla fine del processo, solo le aree aventi interesse legato al change detection maggiore e che dall'informazione a-priori conosciuta risulta essere una modifica avvenuta realmente col passare del tempo.

A-priori Information

Col termine *informazione a-priori* faremo riferimento ad informazioni, correlate ai nostri modelli 3D in input, imprescindibili ed immodificabili.Tali informazioni vengono acquisite dal metodo tramite dati presenti nel data base preesistente relativo alla zona di studio e correlato alla nuvola di punti "vecchia" o tramite deduzioni o conoscenza delle caratteristiche intrinseche dei modelli in input e successivamente utilizzate al fine di minimizzare gli errori presenti durante i vari step del metodo e quindi poter ottenere in output risultati accurati e di sole zone "interessanti" per il change detection. Ma perchè proprio l'informazione A-priori? Oggigiorno, in letteratura non esistono metodi di change detection che utilizzano o sono basati sull'informazione reale correlata ai modelli studiati o alle zone terresti di studio e proprio per questo si è scelto di pensare e studiare un change detection "innovativo" basato anche su queste informazioni e che insieme ad altre metodologie, come visibile dalla Figura 13, permette l'ottenimento e la restituzione automatica delle aree di cambiamento più rilevanti e quindi il successivo aggiornamento dei dati correlati ad esse. Di seguito, tramite la Figura 14, spieghiamo due piccoli esempi di cosa è un'informazione a-priori correlata ad un modello 3D, in questo caso specifico una nuvola di punti.

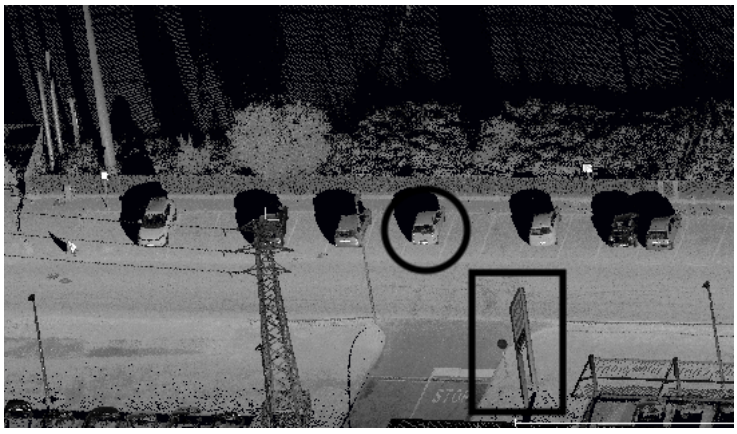


Figure 14: Esempio utile alla spiegazione dell'a-priori information

Nella nuvola di punti, visibile nella figura sopra, sono evidenziati tramite un cerchio ed un rettangolo due oggetti presenti internamente al modello stesso considerato, quali una macchina ed un cartello stradale.L'informazioni a-priori conosciuta nel caso di questi due oggetti, in relazione anche agli altri oggetti presenti che

possiederanno una propria informazione a-priori, è proprio che nella posione tot. , e quindi nell'area formata da tot. punti, quei punti rappresentano e sono realmente una macchina ed un cartello stradale.Queste informazioni sono legate all'ambiente di studio in modo imprescindibile ed immodificabile, quindi qualsiasi computazione su questi punti che porta all'ottenimento di risultati errati o non accurati potrà essere modificata o non considerata proprio sulla base dell'a-priori information.Infatti, se considerassimo i punti che formano la macchina e durante una delle fasi del nostro metodo questi punti venissero considerati come punti "non macchina", ma tipo "vegetazione" è possibile grazie proprio alla conoscenza a-priori legata all'ambiente di studio, la quale ci suggerisce che quei punti non rappresentano una vera "vegetazione" in quanto sono punti su una rete viaria con delle determinate caratteristiche dove non è possibile la presenza di vegetazione e al più possono rappresentare una "macchina", notare questo "errore" e correggerlo automaticamente. Quindi, come deducibile e come visibile dalla Figura 13, l'informazione a-priori è parte essenziale, nel metodo proposto, della fase di classificazione.L'informazione imprescindibile, immodificabile e correlata all'ambiente di studio, come spiegato in modo dettagliato in uno dei paragrafi successivi, nel nostro metodo viene impiegata in modo specifico per ridurre drasticamente gli errori che possono essere presenti nell'output di un processo di classificazione automatica di un modello 3D.Infatti, il metodo conoscendo le informazioni reali correlate all'ambiente di studio grazie all'informazioni a-priori, può verificare la classificazione ritornata dal processo di classificazione automatica e correggere automaticamente la classificazione di eventuali punti o oggetti classificati erroneamente associandoli alla giusta classe di appartenenza, permettendo così al metodo stesso di avere a disposizione una classificazione accurata e con pochi errori, poter rilevare ed eliminare tutti i falsi positivi presenti e quindi ottenere in output risultati finali correlabili al cambiamento avvenuto per la zona di studio/interesse più accurati e veritieri.

Si spera che con questo excursus generale sull'a-priori information il lettore percepisca l'importanza dell'utilizzo di queste informazioni in quanto perni essenziale e promotori dell'accuratezza nel metodo.

Ottenimento dei dati in input

I dati utilizzati in input, soprattutto per quanto riguarda i due modelli 3D utilizzati, rappresentati mediante formato nuvola di punti, possono essere ottenuti tramite rilevazione da aereo, elicottero o da *Mobile mapping System*(MMS).Nel nostro caso specifico essendo le due nuvole di punti utilizzate in input ottenute tramite rilevazione da MMS ci soffermeremo sulla trattazione di questo sistema e di come esso ottiene i dati necessari per il metodo dall'ambiente circostante.E' opportuno dire, prima di passare alla trattazione dei sistemi MMS, che anche nel caso di rilevamenti da aereo o elicottero il processo di acquisizione dei dati rimane pressochè uguale a quello di un MMS con l'unica differenza che nel caso di aerei o elicotteri le acquisizione verranno effettuate dall'alto , mentre nel caso dei MMS i dati verranno acquisiti da terra.

Un Mobile Mapping System rappresenta uno dei più avanzati sistemi di rilevazione terrestre che combina in un'unico sistema camere fotogrammetiche, laser scanner, telecamere, DGPS, sistemi inerziali ed altri strumenti con lo scopo di ac-

quisire immagini e nuvole di punti georeferenziate da cui poter estrarre dati localizzati, qualitativi e metrici relativi alle diverse classi di elementi territoriali di specifico interesse. Più semplicemente un MMS può essere definito come un sistema che permette di rilevare, in modalità dinamica e mediante l'utilizzo di telecamere, laser scanner, sensori e sistemi di orientamento, dati 2D e 3D terrestri georeferenziati, ad alta risoluzione ed accuratezza. Un generico Mobile Mapping System è illustrato nella Figura 15 ed composto da 3 principali componenti hardware:

- sensori ottici, come laser scanner e fotocamere, utili a rilevare l'ambiente circostante in formato nuvola di punti ed ottenere foto dello stesso.
- sensori di posizionamento/navigazione.Solitamente vengono utilizzati il sistema satellitare globale di navigazione(GNSS) e un'unità di misurazione inerziale(IMU).La combinazione dei dati ottenuti da GNSS e IMU permette, mediante L'applicazione di un algoritmo chiamato *SLAM*, l'ottenimento continuo e preciso della traiettoria del sistema stesso e quindi di conoscere il posizionamento reale dell'ambiente circostante rilevato dai sensori ottici.
- unità di controllo e sincronizzazione.Tale unità è utilizzata per adottare una rigorasa sincronizzazione temporale dei sensori ottici e dei flussi di dati ottenuti dai sensori di posizionamento al fine di ottenere una mappatura precisa dell'ambiente circostante.

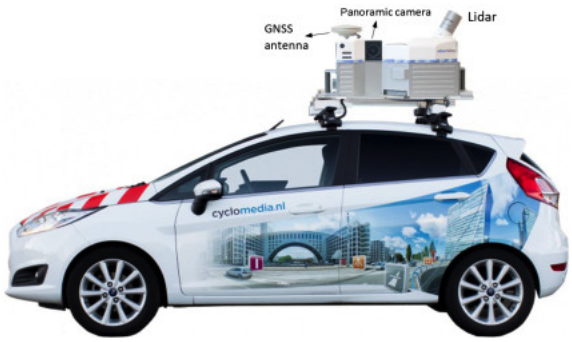


Figure 15: Mobile Mapping System

Intuativamente, il sistema viaggia sul territorio terrestre e mediante questi hardware, anche visibili dall'immagine, rileva e contemporaneamente geolocalizza l'ambiente circostante a 360 gradi, ma come avviene la geolocalizzazione dei dati rilevati?

E' opportuno, prima di comprendere meglio come avviene il processo, precisare che qualora il sistema utilizzasse dei laser scanner per ottenere le rilevazioni dell'ambiente circostante in formato nuvola di punti prenderà il nome di *Mobile Laser Scanning*(MLS). Un MLS, infatti, si serve della tecnologia LiDar per acquisire con elevata velocità ed in tempo reale dense nuvole di punti georeferenziate dell'ambiente circostante. La tecnologia LiDar sta per *Laser Detection and Ranging*.Essa permette di misurare la distanza da un oggetto illuminandolo con luce laser e di restituire

informazioni 3D ad alta risoluzione sull'ambiente rilevato.Solitamente un sistema LiDar è composto da laser scanner, fotorilevatori e circuiti integrati di lettura, chiamati *ROIC*, utilizzati per misurare la distanza tramite illuminazione del bersaglio ed analizzare la luce riflessa. Solo a questo punto è facile intuire per il lettore che un MMS geolocalizza automaticamente i propri dati rilevati tramite l'utilizzo sia dell'algoritmo di *SLAM*, che grazie ai dati ottenuti dai sensori GNSS/IMU calcola in modo continuo la traiettoria e quindi la posizione del sistema, che della tecnologia LiDar, la quale tramite *ROIC*, in primis computa il *time of fly* di un raggio rispetto un oggetto presente nell'ambiente o una porzione dell'ambiente circostante, ossia il tempo impiegato da un raggio laser luminoso per andare dal trasmettitore laser presente sul MMS verso un oggetto o porzione di ambiente circostante e per tornare indietro verso il rilevatore laser presente sempre sul sistema, e successivamente moltiplica tale valore per $C/2$, dove c rappresenta la velocità di propagazione della luce, ottenendo la distanza dell'oggetto o della porzione di ambiente circostante considerata rispetto il laser o il sistema.Applicando questa metodologia a livello 3D si ottiene una nuvola di punti geolocalizzata, come quella visibile in Figura 16, che rappresenta la distanza degli oggetti o delle porzioni di ambiente circostante rilevati/e dal sensore LiDar o dal sistema, dove la geolocalizzazione di ogni punto è stata ottenuta proprio sulla base dell'accurata e continua conoscenza delle coordinate della troaiettoria seguita dal sistema MMS e dalla conoscenza della distanza degli oggetti circostanti o porzioni di ambiente circostante considerate rispetto al sensore laser o al sistema.

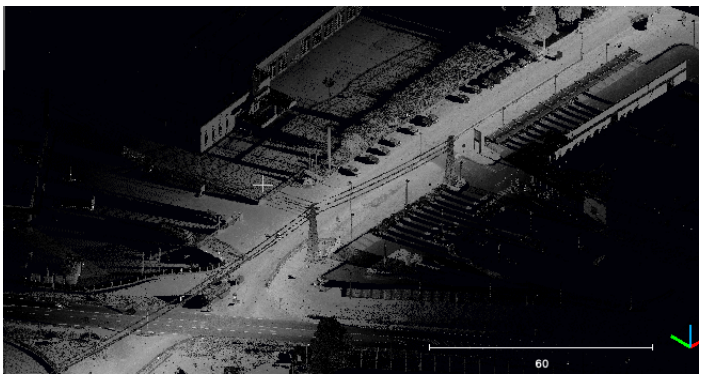


Figure 16: nuvola di punti ottenuta da MMS utilizzante tecnologia LiDar

Infine, è utile precisare ai fini delle caratteristiche di funzionamento e implementazione del metodo proposto che le nuvole di punti utilizzate in input ed ottenute da rilevazioni MMS(o MLS più specificatamente) sono nuvole terrestri che possono rappresentare, come esempio, strade, aree urbane, aree rurali, montagne, etc, e dove ogni punto appartenente a queste nuvole di punti utilizzate in input possiede associati 4 attributi essenziali:

- coordinata x
- coordinata y

- coordinata z
- intensità

I primi 3 attributi permettono l'identificazione della tridimensionalità del punto stesso appartenente alla nuvola di punti considerata, mentre l'ultimo valore, l'intensità, rappresenta il valore di intensità ritornato al ricevitore a seguito dell'impatto tra il raggio laser emesso dal sistema LiDar e il generico punto considerato.

3.1.2 Classificazione

Il passo successivo alla descrizione dell'input utilizzato dal metodo è la computazione, sulla nuvola di punti in input rappresentante la situazione temporale più recente, definita come "dopo", della classificazione degli oggetti presenti nel modello stesso.Tale classificazione è operata e sarà utile in uno degli step successivi al fine di eliminare quelli che possono essere definiti come falsi positivi, ossia punti appartenenti alla nuvola considerata che portano il processo di change detection ad ottenere risultati errati. Inoltre, la computazione della classificazione degli oggetti presenti nella nuvola in input è stata automatizzata tramite l'utilizzo di un software, open-source, chiamato *Agisoft MetaShape* che permette, tramite vari tool messi a disposizione, di lavorare su modelli 3D, in particolare su nuvole di punti, ed operare su di esse diverse computazioni. Metashape infatti, effettua la classificazione dei punti, e quindi degli oggetti, presenti nella nuvola in input suddividendoli in classi di appartenenza tramite un simil processo a quello presentato e descritto in un dei paragrafi del capitolo precedente trattante la classificazione e tramite lo studio della distribuzione spaziale del modello stesso.Infatti, in base alle distribuzioni dei punti che formano l'oggetto nella nuvola in input, il software riesce a "rilevare" l'appartenenza del generico punto ad una data forma che rappresenta l'oggetto stesso ed associarlo ad una delle classi conosciute. Le classi a cui può essere associato un punto presente nella nuvola in input sono:

- vegetazione
- costruzione
- macchina
- strada
- mare/fiume

L'associazione viene effettuata dall'algoritmo utilizzato da MetaShape elaborando ogni punto della nuvola considerata ed attribuendo ad ognuno di essi un valore reale intero che rappresenta la classe di appartenenza e che va ad aggiungersi agli attributi citati nel paragrafo precedente e caratterizzanti il generico punto.Su metashape al punto rilevato e predetto come "vegetazione" viene attribuito il valore 5, come "strada" il valore 11, come "macchine" il valore 2, come "mare/fiume" il valore 1 ed infine come "costruzioni" il valore 6. L'output ottenuto da metashape dopo l'applicazione del tool di classificazione è la stessa nuvola di punti passata

in input al processo, ma classificata, cioè su di essa sono visibili, come mostrato dalla Figura 17, per ogni punto e attraverso un colore , le classi di appartenenza precedentemente citate.

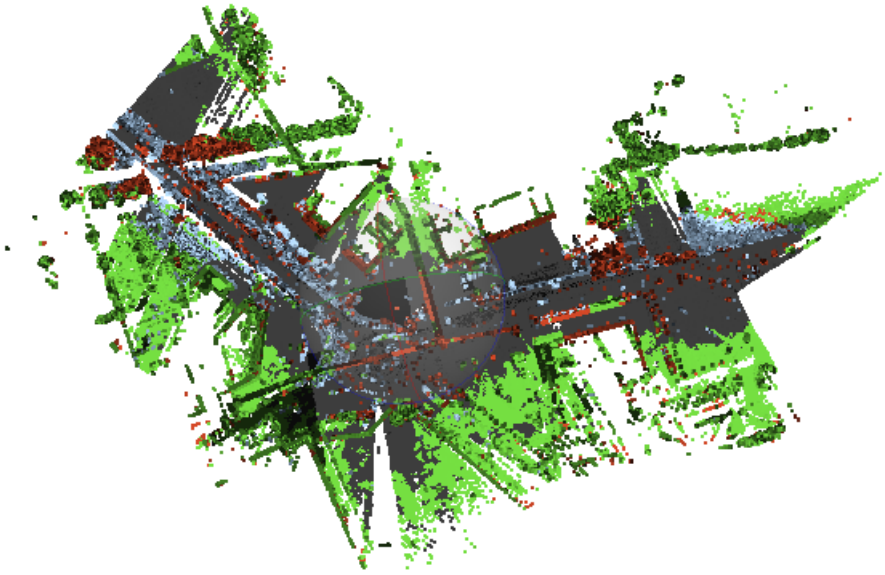


Figure 17: Vista dall'alto di una nuvola di punti classificata attraverso Agisoft MetaShape

Infine, dall'immagine soprastante è visibile che ai punti classificati come "vegetazione" è stato assegnato un colore verde, i punti classificati come "strada" un colore grigio, i punti "macchina" sono rappresentati attraverso un colore azzurro, mentre ai punti classificati come "costruzione" è assegnato un colore rosso.Anche se non presenti nell'immagine considerata punti classificati come "mare/fiumi" ad essi verrà assegnato un colore blu scuro.

Utilizzo dell'a-priori information per verificare la classificazione

In questo contesto di classificazione della nuvola di punti rappresentante la situazione temporale più recente è possibile utilizzare, come già accennato in precedenza, l'informazione imprescindibile e correlata all'ambiente circostante di studio ed ottenibile dai dati presenti nel database con lo scopo di raffinare automaticamente i risultati ottenuti dal processo di classificazione automatizzata applicato da Agisoft Metashape. Infatti, come mostrato dalla figura 18, è probabile che l'algoritmo utilizzato dal software generi in output errori di classificazioni dovuti alla scarsa accuratezza, in alcune aree, della nuvola considerata e proprio con l'obiettivo di ridurre codesti errori, quindi ottenere una più possibile consona classificazione rispetto alle reali informazioni della zona di interesse, grazie a questa fase di verifica della classificazione , è possibile verificare e modificare automaticamente, sulla base dell'informazione a-priori conosciuta, la classificazione ottenuta dal software Agisoft Metashape, rilevando tutti quei punti classificati erroneamente e ri-classificandoli automaticamente alla giusta classe di appartenenza.

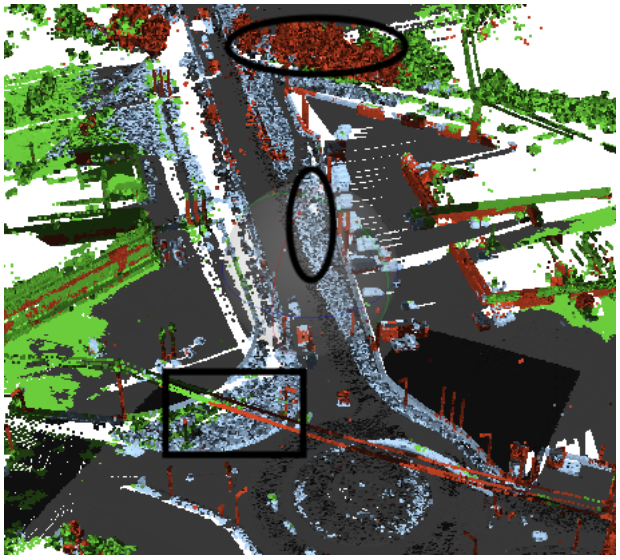


Figure 18: Errori presenti in una nuvola di punti classificata tramite Agisoft MetaShape

La figura 18, mette alla luce alcuni errori presenti in un generico output di classificazione automatica ottenuto dal software Metashape.Come è visibile sono state delimitate aree classificate erroneamente, il rettangolo è stato classificato attraverso la classe "macchina", uguale storia per l'area delimitata da un cerchio verticale, mentre l'area delimitata dal cerchio orizzontale è stata classificata come "costruzione", ma dalla conoscenza dell'informazioni a-priori ottenibile dal database e correlata all'ambiente di studio e quindi alla strada stessa e all'intorno spaziale di queste si evince che per i primi due casi non è possibile, per queste zone, ritrovare delle macchina in quanto fuori dal tratto viabile ed in quanto dai dati contenuti nel database, e quindi dall'informazione a-priori conosciuta, queste sono tutte zone formanti la delemitazione della rete viaria, contenenti solo alberi e quindi devono essere classificate attraverso l'opportuna classe "vegetazione".Mentre per quanto riguarda il cerchio orizzontale classificato attraverso la classe "costruzione" è facile accorgerci sempre sulla base dell'informazione a-priori che questa zona è una zona verde dove non sono presenti case o qualsiasi tipo di costruzione.Quindi il metodo proposto dall'informazioni a-priori ottenuta dai database e dallo studio della zona, attraverso questa fase, rileva che la classificazione precedentemente ottenuta per determinate aree o punti non risulta essere corretta e ri-classifica o modifica/corregge automaticamente la classificazione di aree/punti classificati erroneamente rispetto la realtà in base proprio ai dati preesistenti, migliorando sia l'accuratezza della classificazione che i risultati, relativi al *change detection*, ottenuti in output alla fine dell'intero processo. Alcuni esempi di errata classificazione e rilevazione dell'errore possono essere:

- punti classificati come "vegetazione" presenti in una zona dove, dal database, è visibile che è una zona "mare";
- punti classificati come "vegetazione" appartenenti ad oggetti reali definibili come "costruzione abitativa";

- punti classificati come "macchina" o "vegetazione" che dall'informazione reale e preesistente conosciuta appartengono ad una zona classificabile come "mare/-fiume";
- etc.

In tutti questi casi ed in altri, come già accennato, il metodo proposto, tramite una verifica condotta sulla base dei dati esistenti presenti nel database e dei dati ottenuti dallo studio della zona, e quindi in parole povere sulla base dell'informazione a-priori conosciuta e correlabile al punto o alla zona, riesce a rilevare l'errata classificazione, ottenuta dal tool di *Agisoft Metashape*, di punti o insiemi di punti e a modificarla/-correggerla automaticamente, ottenendo così sicuramente una classificazione della nuvola considerata più accurata rispetto la realtà e con meno errori.

3.1.3 Computazione dei cambiamenti

Il passo successivo alla classificazione della nuvola di punti in input più recente, quindi rappresentante la situazione temporale "dopo", prevede la computazione vera e propria dei cambiamenti avvenuti nel tempo per la data zona di studio.Il metodo proposto rileva i cambiamenti avvenuti basandosi sul calcolo dello scostamento, in un intorno, della coordinata z di ogni punto appartenente alla nuvola di punti più recente e già classificata rispetto i punti più vicini a quest'ultimi appartenenti alla nuvola di punti vecchia definita come "dopo".Tale calcolo, essendo un calcolo abbastanza complesso e dispendioso da un punto di vista temporale è stato automatizzato attraverso il software, open-source, CloudCompare, il quale permette la gestione di nuvole di punti e mediante l'applicazione di uno dei suoi tool, *CloudtoCloud distance*, lievemente modificato, la computazione dello scostamento, in un intorno, della coordinata z avvenuto tra i punti appartenenti a due date nuvole di punti in input. Il generico processo applicato in questa fase è visibile nella figura sottostante 19:

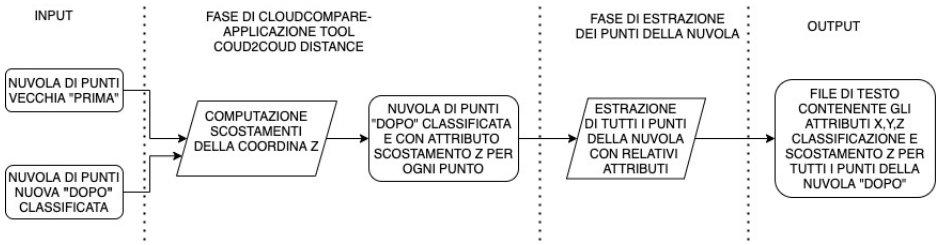


Figure 19: Processo di computazione dei cambiamenti

Come accennato in precedenza e come visibile dall'immagine, nella *fase di CloudCompare*, viene utilizzato il tool, leggermente modificato, *Cloud2Cloud Distance* di CloudCompare per calcolare lo scostamento, in un intorno, della coordinata z tra i punti delle due nuvole di punti in input.Il meccanismo applicato dal tool è il seguente:

1. la nuvola di punti in input rappresentante la situazione temporale meno recente ,definita sull'immagine come "prima", viene considerata dal tool come la

nuvola di riferimento, mentre la nuvola di punti rappresentante la situazione temporale più recente e già classificata nello step precedente, definita come "dopo", viene considerata dal tool come la nuvola di comparazione.

2. Per ogni punto appartenente alla nuvola di comparazione, attraverso l'algoritmo *Nearest Neighbor Distance*, ricerca il punto più vicino ed appartenente alla nuvola di punti di riferimento.Se la distanza euclediana tra i due punti più vicini ritrovati risulta essere < della somma delle accuratezze planimetriche possedute dalle due nuvole di punti in input allora passa al punto 4, altrimenti, il punto appartenente alla nuvola di punti di comparazione non viene considerato in quanto non esiste nessun punto abbastanza vicino nella nuvola di riferimento per poter computare un cambiamento veritiero avvenuto e gli viene attribuito un valore 0 di scostamento della coordinata z.

3. calcola lo scostamento della coordinata z tra i due punti considerati con la seguente formula:

$$d_z(A, B) = z_A - z_B$$

L'algoritmo applicato attraverso il tool Cloud2Cloud Distance di CloudCompare associa ad ogni punto appartenente alla nuvola di punti di comparazione(che rappresenta ed è proprio la nuvola definita come "dopo" classificata) un nuovo attributo, un valore decimale reale, rappresentante lo scostamento in un intorno della coordinata z rispetto il punto più vicino nella nuvola di riferimento e quindi anche il reale cambiamento avvenuto nel tempo e presente tra i punti appartenenti alla nuvola di punti "dopo" e quelli appartenenti alla nuvola di punti "prima".E' intuibile dall'affermazione precedente che il metodo proposto intende il cambiamento per un generico punto appartenente a una delle due date nuvole di punti in input come uno scostamento spaziale, in un intorno, della coordinata z rispetto il più vicino punto appartenente all'altra nuvola di punti considerara.Inoltre, la leggera modifica applicata al tool *CloudToCloud distance* di *CloudCompare* è quella prevista al passo 3, dove successivamente ad aver ottenuto il punto più vicino presente nella nuvola di riferimento tramite il *NND algorithm*, si effettua un controllo sulla distanza tra i due punti selezionati ed al punto appartenente alla nuvola di comparazione viene associato un valore di scostamento della coordinata z, rappresentante il cambiamento, pari a 0 solamente se la distanza euclediana tra i due punti considerati sia maggiore della somma delle accuratezze planimetriche possedute dalla due nuvole di punti in input.Questa verifica è stata aggiunta e performata al fine di evitare di considerare negli step successivi, e quindi nei risultati finali ottenibili, tutti quei punti appartenenti alla nuvola di punti più recente che non possiedono un punto realmente vicino nella nuvola di punti vecchia, definita come "prima", per cui non è possibile calcolare un valore veritiero di scostamento avvenuto nel tempo.Questo evita la computazione di cambiamenti poco veritieri ottenibili dal ritrovamento, nel passo 3, di punti non realmente vicini nei due modelli da studiare e che poterebbero a falsare i risultati di change detection ottenibili alla fine.

Comunque, come da Figura 19 successivamente al calcolo degli scostamenti, sulla nuvola di punti "dopo", in cui ad ogni punto appartenente ad essa è stato associato un ulteriore attributo, il valore di scostamento computato, è prevista una fase di

estrazione, eseguita sempre mediante CloudCompare, con lo scopo di ottenere in output un file di testo, utile negli step successivi del metodo, rappresentante la nuvola di punti "dopo" stessa ed al cui interno troviamo tutti i punti che la formano con i relativi attributi associati, coordinate x , y e z, valore di classificazione e il valore di scostamento calcolato rappresentante il cambiamento. Quindi nel file di testo ottenuto in output, rappresentante la nuvola di punti definita come "dopo", classificata e per cui sono stati computati i valori di cambiamento(gli scostamenti), ogni riga, rappresentante un punto appartenente alla nuvola di punti, possiede 5 colonne, gli attributi per quel dato punto, una coordinata x, una coordina y, una coordinata z, un valore di classificazione ed il valore di distanza computato in questo step.

E' utile per il lettore, prima di chiudere questo paragrafo relativo alla computazione del cambiamento per la zona di interesse e quindi alla computazione della distanza per ogni punti della nuvola di punti definita come "dopo", accennare alle problematiche che tale meccanismo proposto può presentare ai fini dei risultati ottenibili e correlabili al *change detection*.

Problematiche della computazione

Come si è accennato precedentemente il metodo proposto intende il cambiamento per un generico punto appartenente alla nuvola di punti più recente, definita col termine "dopo" e già classificata, come uno scostamento spaziale, avvenuto nel tempo, rispetto il più vicino punto appartenente all'altra nuvola di punti presente in input e rappresentante la situazione temporale meno recente.Da questo si evince facilmente che il metodo di change dection proposto dipende dalla distanza computata in questo step e da come essa viene calcolata, ma anche dall'accuratezza dei modelli 3D posseduti in input, in quanto se la nuvola di punti considerata come di "riferimento" dal tool è abbastanza densa allora calcolare la distanza euclediana tra un punto appartenente alla nuvola di "confronto" e il punto più vicino presente nella nuvola di "riferimento" risulta essere una stima abbastanza precisa ed attendibile da poter essere utilizzata per il processo di change detection e quindi per rilevare il cambiamento stesso, mentre se la nuvola di "riferimento" non fosse abbastanza densa allora questo valore di scostamento calcolato dal punto più vicino non rappresenta una stima attendibile e precisa del reale cambiamente avvenuto nel tempo per il dato punto, non potrà essere utilizzata e quindi dovremmo trovare un metodo alternativo per calcolare i valori di queste distanze che rappresentano il cambiamento.Per ovviare a quest'ultimo problema e al fine di ottenere una migliore approssimazione della vera distanza e quindi del reale cambiamento, CloudCompare mette a disposizione dei modelli locali che approssimano la superficie della nuvola di punti di riferimento poco densa.Tali modelli sono 3:

1. Least Square best fitting plane
2. 2D1/2 Delaunay triangulation
3. Quadratic heigh function

Il primo modello approssima localmente la nuvola ad un piano, il secondo utilizza la triangolazione di Delaunay per operare l'approssimazione locale, mente il terzo

approssima localmente la nuvola di punti attravero l'utilizzo di una funzione di altezza del tipo $z = ax + by + cx^2 + dy^2 + exy$.

L'idea di base applicata dal tool di CloudCompare per migliorare la precisione della distanza calcolata ed ovviare alle problematiche della densità è quella di computare, per un generico punto appartenente alla nuvola di punti di comparazione, cioè quella più recente e già classificata, il punto più vicino appartenente alla nuvola di riferimento(quella meno densa che presenta la problematica).Successivamente, l'algoritmo applicato dal tool di CloudCompare, modella localmente, attraverso l'applicazione sul punto più vicino computato precedentemente e su alcuni vicini di questo di uno dei tre modelli matematici elencati, la superficie della nuvola di punti di riferimento sottostante al punto più vicino e solo a questo punto computa lo scostamento avvenuto sulla coordinata z tra il punto considerato inizialmente nella nuvola di comparazione e il punto più vicino appartenente al nuovo modello creato ed associa tale valore calcolato al primo punto attraverso l'attributo distanza.Tale computazione della distanza di ogni punto appartenente alla nuvola di comparazione rispetto i punti appartenenti al nuovo modello ottenuto modellando la superficie della nuvola di riferimento attraverso un modello locale matematico risulta essere statisticamente più precisa e meno dipendente dal campionamento/rilevamento o dalla densità delle nuvole.

Infine, è utile precisare al lettore che il nostro metodo, in caso di nuvola di riferimento poco densa, può utilizzare questi modelli matematici, messi a disposizione dal software CloudCompare, per migliorare la precisione delle distanze computate in questo step rappresentanti a tutti gli effetti i cambiamenti reali avvenuti nel tempo per generici punti, ma è sempre consigliato e risulta essere più efficiente applicare il metodo proposto a nuvole di punti in input aventi simil densità spaziale al fine di non incorrere in errori, che potrebbero presentarsi nei risultati finali di change detection ottenuti, dovuti o al calcolo approssimato delle distanze o addirittura al calcolo di valori di distanza, per generici punti appartenenti alla nuvola di comparazione, non in linea coi reali cambiamenti avvenuti nel tempo.

3.1.4 Filtraggio dei punti significativi per il change detection

Dopo aver ottenuto, dallo step precedente, il file di testo rappresentate la nuvola di punti definita come "dopo", classificata ed in cui per ogni punto appartenente ad essa sono state calcolate le distanze che rappresentano il cambiamento, è opportuno filtrare da questo solamente i punti rilevanti per il processo di change detection. in questo step ci serviremo di un algoritmo, appositamente creato in C++, per poter eliminare, dal file di testo in input, tutti quei punti che non possiedono una distanza associata rappresentante un cambiamento rilevante e tutti quei punti classificati come dei falsi positivi. Al fine di poter eliminare i primi si è definito un intervallo di threshold, cioè una soglia su quanto sia rilevante un dato punto in relazione alla distanza calcolata, che permetta di definire un valore di distanza rappresentante un cambiamento rilevante.Tutti i punti presenti nel file in input a cui è stata associata, attraverso lo step precedente, una distanza appartenente all'intervallo di threshold verranno eliminati.Solitamente, tale intervallo considerato è $[-0.20, +0.25]$ metri in quanto, durante la sperimentazione del metodo, ci si è accorti che tutti quei punti a cui CloudCompare, attraverso il suo tool, ha associato un valore di distanza cal-

colata compreso tra $-0.2e + 0.25$ metri non risultano essere dei punti per cui sono avvenuti dei cambiamenti rilevanti.Inoltre, anche grazie alla conoscenza dell'a-priori information correlata all'ambiente di studio è stato possibile osservare che per tali punti non esisteva un reale cambiamento visibile ad occhio nudo tra un prima ed un dopo e quindi al fine di ridurre gli errori, diminure la dimensione dell'input ed ottenere le sole aree per cui sono presenti dei cambiamenti rilevanti si è scelto di filtrarli fuori, eliminarli e non tenerli in considerazioni nei risultati finali.Mentre, per quanto riguarda i falsi positivi c'è da dire che talvolta è possibile ritrovare dei cambiamenti rilevanti che realemente non sono dei veri e propri cambiamenti, questo è il caso, nel nostro metodo proposto, dei punti classificati come "vegetazione" o "macchina".La vegetazione come sappiamo, col passare del tempo, cresce, quindi i punti formanti la vegetazione stessa, presenti in due situazioni temporali distinte, potrebbero spostarsi verso l'alto e ad essi verrà sicuramente associato un valore di distanza rappresentante un cambiamento rilevante, ma essendo un cambiamento correlato a punti formanti una vegetazione non dovrà essere consideranto in quanto falserebbe alcune dei risultati di change detection ottenibili, non essendo un cambiamento morfologico o dovuto all'uomo, ma solo correlato al tempo.Altro caso importante di falso positivo da tenere in considerazione, nel nostro metodo, è la presenza di veicoli o punti classificati come "macchina" nella nuvola di punti considerata.Anche qui, la presenza di una macchina in una zona di studio rispetto un'altra situazione temporale rappresentante sempre la stessa zona in cui però la macchina non è presente non rappresenta un cambiamento rilevante da tenere in considerazione nel processo di change detection in quanto non correlato ad una vera modifica del territorio o della zona stessa, ma legata solamente ad un oggetto che si trovava li per caso durante le rilevazioni della nuvola di punti.Da questo excursus si evince facilmente che in questo step, tutti i punti, presenti nel file in input, a cui è associata una distanza $<$ di threshold e/o sono classificati come "vegetazione" (valore 5) o "macchina" (valore 2) vengono eliminati e quindi non tenuti in considerazione negli step successivi del metodo. Il processo applicato in questa fase è mostrato nella figura 20:

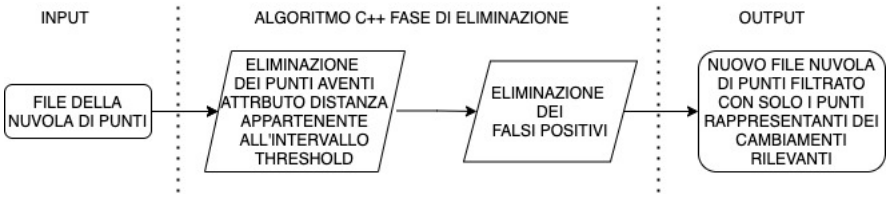


Figure 20: Fase di filtraggio dei punti rilevanti al cambiamento

Come deducibile, e come visibile dall'immagine, in output questa fase restituisce un nuovo file di testo, rappresentante una nuova nuvola di punti, contenente solamente i punti associati ad un cambiamento rilevante e che non risultano essere classificati come falsi positivi.Nel file di testo outputtato, ogni riga, come precedentemente, rappresenta un punto a cui, questa volta, sono associati solamente i 3 attributi x, y e z, rappresentanti le coordinate del punto nello spazio, attraverso la presenza di 3 colonne per ogni riga.Gli attributi relativi alla classificazione e al valore di distanza associati ad ogni punto presente nel file di testo in input a questa fase

verranno eliminati e non considerati nel file di testo ottenuto in output in quanto non più utili per il metodo stesso e al fine di poter risparmiare, negli step successivi, in termini di memoria utilizzata.

Falsi Positivi

Si è accennato già in questo paragrafo, in modo molto limitato, a cosa sono i falsi positivi al fine di spiegare in generale ciò che accade in questa fase di filtraggio ed eliminazione di punti non rilevanti per il processo di rilevamento del cambiamento, ma è utile dedicare maggiore spazio all'argomento in quanto problematica di interesse nel campo del change detection. I falsi positivi possono essere descritti come dei punti appartenenti alle nuvole di punti di interesse, rappresentanti la zona di studio, per i quali, molto probabilmento, verrà rilevato un cambiamento rilevante tra una situazione temporale antecedente ed una successiva, ma che realmente non rappresentano un cambiamento veritiero, dovuto a cambiamenti morfologici dell'area di studio o a cambiamenti dovuti all'intervento umano attraverso la costruzione di oggetti non rimovibili nella zona di interesse.Codesti punti non dovranno e non devono essere presi in considerazione durante il processo di change detection al fine di non ottenere risultati, correlabili al cambiamento stesso, errati, non veritieri e non rilevanti. Come è intuibile, è possibile mettere alla luce la presenza dei falsi positivi, e successivamente eliminarli in questo step, attraverso la fase di classificazione svolta in precedenza, che ci permetterà , come già descritto, di classificare i punti in base a delle classi di appartenenza ben definite.Alcune di queste classi, utilizzate per la classificazione dei punti, rappresentano proprio delle classi di appartenenza per oggetti/punti definibili come falsi positivi o che potrebbero portare all'ottenimento di risultati definibili come "falsi positivi".Infatti, potranno essere considerati come falsi positivi tutti quei punti classificati come:

- vegetazione
- macchina
- banchi di lavoro
- ponteggi edili
- esseri umani
- fiere o banchi di fiere temporali

Da questo elenco è deducibile che i falsi positivi sono punti direttamente correlabili ad oggetti che potrebbero mutare le loro dimensioni al passare del tempo, è il caso della vegetazione, o ad oggetti non persistenti nella zona di studio e che sono presenti nelle nuvole di punti di interesse o in una delle due solamente durante le rilevazioni, questo è il caso di macchine che potrebbero trovarsi parcheggiate nell'are di interesse, o essere umani che stanno facendo una passeggiata/corsa in quel momento in quella zona, o ponteggi edili per dei lavori di ristrutturazione degli edifici della zona o fiere di qualsiasi tipo che dopo un lasso di tempo definito potrebbero spostarsi in una differente locazione. In un processo di change detection, queste classi di punti

o più in generali i punti classificati come vegetazione, macchina, banchi di lavoro, etc hanno in comune solamente l'ottenimento di risultati correlati al cambiamenbiamento della zona definibili come "falsi positivi", ossia non rilevanti, di poco conto e non intrinsici al cambiamento terrestre.Ovviamente, nella realtà, sono presenti, oltre quelle elencate, molte altre classi e sotto-classi associabili ai falsi positivi o che rappresentano,a tutti gli effetti, dei falsi positivi.

Di seguito, tramite l'utilizzo delle figure 21 e 22, vengono riportati degli esempi di falsi positivi.

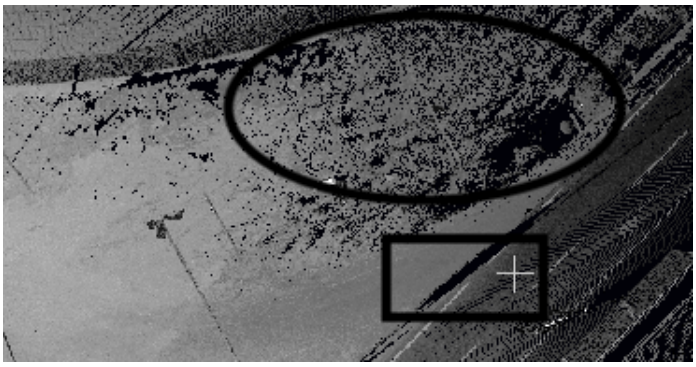


Figure 21: Nuvola di punti meno recente

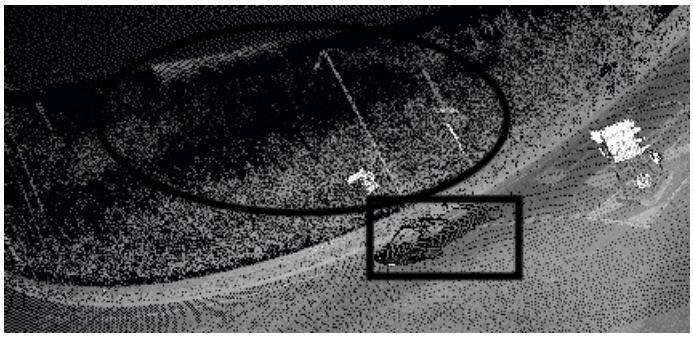


Figure 22: Nuvola di punti più recente

Lo studio del change detection per le due nuvole di punti in Figura 21 e 22, rappresentanti la medesima zona in due differenti momenti temporali, uno meno recente ed uno più recente, metterebbe alla luce, da una parte, la presenza di cambiamenti rilevanti come la presenza di nuovi cartelli stradali o la presenza di un muretto, prima non presente, che delimita l'area stradale, dall'altra anche la presenza di falsi positivi.I falsi positivi, in questo caso specifico, sono ottenuti dall'opportuno rilevamento del cambiamento avvenuto per i punti formanti la vegetazione, che come evidenziato dai due cerchi presenti nelle due nuvole di punti, è cresciuta/mutata, e per i punti formanti la macchina presente nella nuvola di punti più recente, ma non presente nella nuvola meno recente ed evidenziata con un rettangolo.infatti, il processo ottiene per questi punti, in entrambi i casi, un valore di distanza, rappresentante il cambiamento stesso, rilevante e quindi se questi punti non venissero

opportunamente eliminati o filtrati i risultati in output ottenuti e correlabili al cambiamento della zona risulterebbero, in parte, errati. Si spera che con l'aiuto di questo esempio il lettore abbia capito cosa sia un falso positivo e come lo si può identificare e soprattuto abbia inteso la rilevanza dell'eliminazione dei falsi positivi in questa fase al fine di poter ottenere risultati più accurati e veritieri possibili, ma è anche utile precisare che il metodo trattato considera solamente falsi positivi i punti classificati come "vegetazione" o "macchina" ed ammette la presenza di essi a patto che questi punti o le aree di cambiamento associate a dei falsi positivi e ritornate dal metodo siano in numero/dimensione limitato/a rispetto i punti o le aree ritornate dal metodo realmente rilevanti per il cambiamento, al fine di non incidere notevolmente sul tempo di esecuzione richiesto dagli step successivi, ma più in generale dal metodo stesso, e sui risultati finali correlati al change detection della zona. Infine, possiamo definire che l'efficacia della fase dell'eliminazione dei falsi positivi dipende notevolmente dalla fase di classificazione e da come essa viene effettuata, infatti, più sarà accurata e consona alla realtà maggiori punti definibili come falsi positivi potranno essere eliminati incidendo notevolmente sulla bontà dei risultati finali ottenibili.

Algoritmo applicato

Come già accennato, per svolgere le operazioni di filtraggio dei punti e quindi eliminazione di essi dalla nuvola di punti considerata in formato file di testo è stato sviluppato un apposito e semplice algoritmo in C++. L'algoritmo utilizzato svolge i seguenti passi:

1. Legge il file di testo in input riga per riga

2. Controlla la quinta colonna della riga considerata che definisce la distanza computata dal software CloudCompare e rappresenta il cambiamento.Se tale valore di distanza non è appartenente all'intervallo di threshold allora il punto viene mantenuto e si passa al passo 3, altrimenti se il valore di distanza è $<$ del limite inferiore di threshold o $>$ del limite superiore di threshold il punto viene eliminato(non considerandolo) e si riesegue il punto 2 passando alla riga successiva.

3. In questo passo, l'algoritmo controlla la quarta colonna della riga considerata, la quale rappresenta la classificazione e quindi la classe di appartenza per quel punto.Se tale valore fosse diverso da 5 ("vegetazione") o 2 ("macchina") allora mantengo il punto e salvo le coordinate x, y e z di quest'utimo , quindi le prime tre colonne della riga considerata, in un vettore di punti.Se invece il valore di classificazione per la data riga è uguale a 5 o 2 allora elimino tale punto(non considerandolo) e si riesegue il punto 2 passando alla riga successiva del file.

4. Successivamente ad aver eseguito i punti 2 e 3 per tutte le riga del file in input, si otterrà un nuovo vettore, contenente le coordinate x, y e z dei soli punti definiti come rilevanti ai fine del cambiamento e non classificati come falsi positivi.Questo vettore viene stampato in un file di testo, chiamato 'filtrati.txt', che rappresenta la nuova nuvola di punti ottenuta in output in questa fase e

contenente solamente i punti più di interesse per il processo change detection per la zona di studio.

Utilità dello step performato

Oltre all'eliminazione dei punti poco rilevanti ai fini del rilevamento del cambiamento e dei punti classificati come falsi positivi altro aspetto importante di questo step, che deve essere messo alla luce, è la diminuzione della dimensione dell'input. Essendo che il metodo lavora con nuvole di punti di grande dimensioni in input composte all'incirca da milioni di punti è molto utile, soprattutto per gli step successivi a questo, che come visibili dalla Figura 13 prevedono una fase di clusterizzazione ed una fase di defenizione dei poligoli frontiera per i cluster ritrovati e rilevanti, eliminare tutte quelle informazioni, e quindi i punti stessi, non utili, di poco interesse per il processo change detection al fine di riddurre i tempi di esecuzione richiesti negli step successivi ed in generale dal metodo stesso.Questa eliminazione provocherà, da un lato, una riduzione della dimensione dell'input, infatti, il nuovo file di testo ottenuto in output in questa fase avrà dimensione sicuramente ridotte rispetto le dimensioni del file di testo in input, dall'altro lato, invece, provocherà una riduzione del tempo di esecuzione richiesto per le due operazioni applicate successivamente, le quali, risultano computazionalmente complesse e richiedenti un elevato tempo di esecuzione.Questi due aspetti incideranno notevolmente sulla riduzione del tempo di esecuzione totale richiesto dal metodo stesso per poter ottenere dei risultati finali correlati al cambiamente avvenuto nella zona di studio.

3.1.5 Clusterizzazione dei punti rilevanti/significativi per il cambiamento avvenuto

Dal file di testo ottenuto nello step precedente e contenente solamente i punti più rilevanti per il cambiamento avvenuto nella zona di studio è opportuno estrarre codesti punti e raggrupparli in gruppi tramite un algoritmo di clusterizzazione iterativo basato sulle distanze ed opportunamente sviluppato in C++ al fine di poter ottenere degli insiemi di punti che rappresentano le aree di cambiamento.Il processo di clusterizzazione applicato è visibile attraverso il flow-chart presente in Figura 23 sottostante:

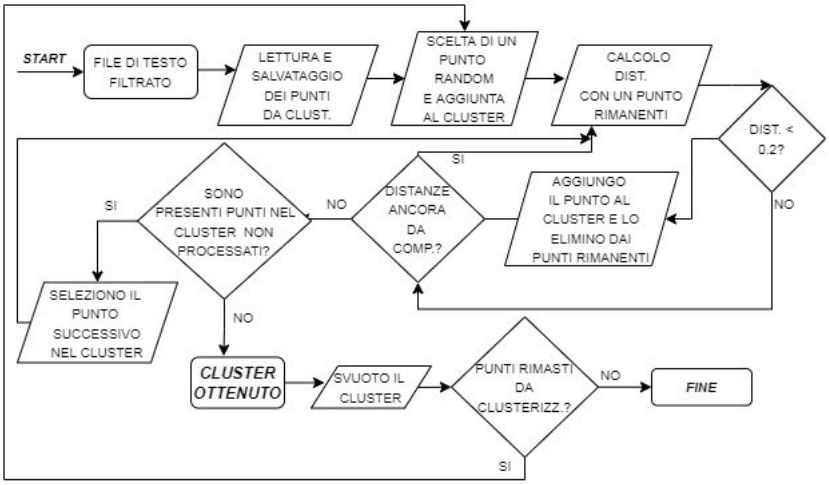


Figure 23: Flow-chart del processo di clusterizzazione

L'algoritmo sviluppato per svolgere la clusterizzazione dei punti effettua i seguenti passi:

1. legge riga per riga il contenuto del file di testo in input, ottenuto dalla fase precedente, e salva all'interno di un vettore di punti, chiamato *points*, le tre colonne presenti per ogni riga, rappresentanti le coordinate x, y e z per il generico punto.

2. seleziona in modo random un punto tra tutti i punti presenti nel vettore *points*, eliminandolo dal vettore di punti e aggiungendolo ad un nuovo vettore chiamato *cluster*.

3. Calcola, per ogni punto contenuto sul vettore *cluster* la distanza euclediana con ogni punto rimanenti sul vettore *points*, se tale distanza computata tra il punto del cluster e il punto rimanente sul vettore points fosse $<$ di un certo valore di threshold(solitamente impostato a 0.2 metri) allora aggiungo il punto appartenente al vettore *points* sul vettore *cluster* e lo elimino da *points* stesso.Riitinero il punto 2 fin quanto non ho processato tutti i punti presenti nel vettore *cluster* e solo a questo punto l'algoritmo passa al punto 3.

4. outputta, in formato file di testo, il cluster ottenuto.

5. Se non ha clusterizzato tutti i punti, cioè se ogni punto, presente nel file di input, non fosse ancora appartenente ad un cluster e quindi sono presenti ancora dei punti sul vettore *points* l'algoritmo passa al punto 2 e continua, altrimenti termina la sua esecuzione avendo clusterizzato tutti i punti.

Dalla descrizione dell'algoritmo è intuibile che l'operazione di clusterizzazione applicata è puramente iterativa e basato sulle distanze.Ovviamente l'iteratività del metodo applicato sarà controproducente per il tempo di esecuzione richiesto per effettuare l'operazione di clusterizzazione stessa, infatti uno dei "contro" dell'algoritmo proposto per effettuare questo step è proprio il tempo di esecuzione richiesto per

performare il raggruppamento.Al fine di ovviare al problema, soprattutto quando la clusterizzazione deve/dovrà essere applicata ad un numero di punti dell'ordine di milioni, è utile definire delle *condizioni di terminazione* che permettano di "stoppare" il calcolo effettuato al punto 3 ed ottenere i cluster, e quindi dei risultati,in tempi più brevi. Sono state definite o possono essere utilizzate due *condizioni di terminazioni* per il nostro algoritmo di clusterizzazione:

- la dimensione massima del cluster
- la dimensione costante nel tempo del cluster

Per la prima, possiamo dire che viene fissato un valore di dimensione massima per i cluster ottenibili in relazione alle dimensioni dell'oggetto più grande per cui è possibile rilevare un cambiamento.Quando all'interno del vettore *cluster* verranno aggiunti un numero di punti uguale a quello definito tramite il valore dimensione massima allora l'algoritmo si ferma ed outputta il cluster creato.Questa condizione comunque presenta dei limiti, in quanto, è molto probabile che si possano ottenere dei cluster molto vicini tra di loro e quindi potrebbe occorrere una fase aggiuntiva, successivamente ad aver ottenuto tutti i cluster, che permetta di unire tutti i cluster molto vicini o che si toccano. Mentre per quanto riguarda la secondo condizione è una condizione correlabile alle dimensioni del cluster in relazione al tempo di esecuzione. infatti, è possibile che con l'andare avanti delle iterazioni, quindi col passare delle "epoche", le dimensioni del cluster rimangano costanti, invariate, cioè non vengono aggiunti punti al cluster per un numero notevole di iterazioni.Fissando un valore massimo di epoche per cui il cluster può avere dimensione costante è possibile definire un'ulteriore *condizione di terminazione* che permetta qualora le dimensioni del cluster rimangono invariate per un tempo definibile attraverso il valore massimo di epoche fissato di "stoppare" l'iterazione ed outputtare il cluster in tempi più brevi. Comunque, si consiglia di applicare il metodo in forma "generale" senza le *condizioni di terminazione* definite al fine di evitare successivi ed ulteriori step di unione o verifica e non appesantire il metodo stesso.

Cluster rilevanti

In questo paragrafo è utile accennare alla rilevanza dei cluster ottenuti in questo step.Non tutti i cluster o i gruppi di punti ottenuti in questa fase e quindi le aree di cambiamento ritrovate sono significative, infatti è probabile che l'algoritmo sviluppato partendo da un punto iniziale random, outputti o ritrovi cluster contenenti un unico punto o un numero di punti inferiore ad un dato valore di threshold che definisce proprio la rilevanza del cluster in relazione alla densità dei punti per metro quadro posseduto dell'oggetto più piccolo presente nella zona di studio per cui è possibile rilevare un cambiamento.In questi casi saremo difronte a dei cluster definibili come "non rilevanti" ed i punti appartenenti a quest'ultimi verranno considerati come *outliers*.Gli *outlies*, la maggior parte delle volte, sono punti isolati rispetto l'insieme iniziale dei punti da clusterizzare, di per sè significativi ai fini del rilevamento del cambiamento, ma che, successivamente ad averli raggruppati in cluster, non sono in numero sufficiente per rappresentare un cambiamento significante in

relazione alle dimensioni dell'oggetto più piccolo per cui poter rilevare un cambiamente per la data zona di studio.Questi punti e quindi anche i cluster di appartenenza verranno scartati e non considerati nei risultati finali relativi al rilevamento e alla delimitazione delle aree di cambiamento ritrove dal metodo. Inoltre, è stato utile accennare alla rilevanza di un cluster per lo step successivo, in quanto solo per i cluster definibili come "rilevanti" verrà calcolato l'opportuno poligono frontiera, delimitante l'area di cambiamento rilevata dal generico cluster considerato, per cui aggiornare i dati, relativi ai punti appartenenti a quell'area delimitata dal poligono stesso, presenti nella base di dati in input poichè identificano aree di cambiamento significative presenti tra una situazione temporale antecedente ed una più recente in relazione alla zona di studio/interesse.

3.1.6 Ottenimento del poligono frontiera per i cluster rilevanti

Quando il generico cluster viene ottenuto nella fase precedente, in successione ed al fine di aumentare l'automazione del processo stesso, viene calcolato, solo se il cluster è rilevante, il poligono frontiera che delimita il cluster ottenuto e quindi l'area di cambiamento.Attraverso questo step si performa l'obiettivo principale della tesi, ossia quello di ritrovare e delimitare le aree di cambiamento significative per la zona di studio. Per ritrovare il poligono frontiera dato un'insieme di punti, in questo caso il generico cluster ottenuto dallo step precedentemente, è stato sviluppato ed applicato un algoritmo di *Convex-Hull*, già ben noto in letteratura, chiamato *Jarvi's Algorithm* o *Wrapping*.

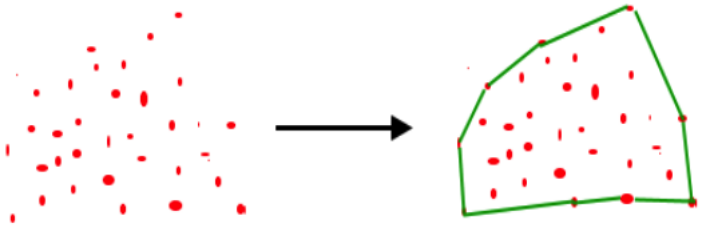


Figure 24: Ottenimento poligono frontiera

Come visibile dalla figura 24 soprastante l'obiettivo performato dall'algoritmo applicato in questa fase è quello, dato un'insieme di punti, di ottenere il poligono di areea massima contenente tutti i punti. L'idea alla base per ottenere questo poligono è molto semplice, infatti, partendo dal punto più a sinistra o comunque partendo dal punto avente coordinata x minore si seleziona, ad ogni iterazione, il punto più esterno ed antiorario rispetto l'insieme rimanente e si continua in questo modo fino al raggiungimento, dinuovo, del punto di partenza ottenendo una delimitazione dell'area formata dai punti e quindi anche una delimitazione all'area di cambiamento ritrovata attraverso i cluster precedentemente computati.Ma come avviene la selezione del punto più esterno e antiorario? Al fine di ritrovare il punto più esterno e più antiorario l'algoritmo utilizza una funzione, chiamata *oriented(p,q,i)* che permette di selezione il successivo punto appartenente alla frontiera.Comunque, di seguito è riportata una spiegazione delle fasi dell'algoritmo *Wrapping* applicato:

1. prima di inizializzare ed applicare la procedura prevista per ritrovare la frontiera, l'algoritmo controlla se il generico cluster, ottenuto in una delle iterazioni dell'algoritmo di clusterizzazione precedente, risulta essere rilevante verificando le dimensione di quest'ultimo. Se il cluster in input all'algoritmo di *Wrapping* risulta essere rilevante allora passo a 2, altrimenti non calcolo il poligono frontiera per quel dato cluster e aspetto che venga costruito, sempre dall'algoritmo di clusterizzazione precedentemente descritto, un ulteriore cluster su cui poter dinuovo riapplicare tale procedura.

2. Inizializza il punto p come il punto avente coordinata x minore;

3. Fin quando non ritrova dinuovo il punto di partenza avente coordinata x minore:

   - Aggiunge il punto q al poligono frontiera se la terna (p, q, r) è in senso antiorario per qualsiasi punto r.Per ritrovare questo punto, l'algoritmo inizializza q al prossimo punto e scorre tutti i rimanenti punti i.Per qualsiasi punto i, se la funzione *oriented(p,i,q)* ritorna un valore che definisce il senso antiorario, allora il punto i è più antiorario di q ed aggiorniamo q ad i.Alla fine , dopo tutte le iterazioni, il valore finale associato a q sarà il punto più antiorario.
   - aggiungo in coda al punto p inziale il punto q, in modo tale da impostare q come il punto successivo a p nel convex-hull di output
   - assegno al punto p il punto q

Di seguito, la figura 25, mostra in dettaglio come il *Wrapping algorithm* ottiene il poligono frontiera, iterazione dopo iterazione, dato un generico cluster in input.

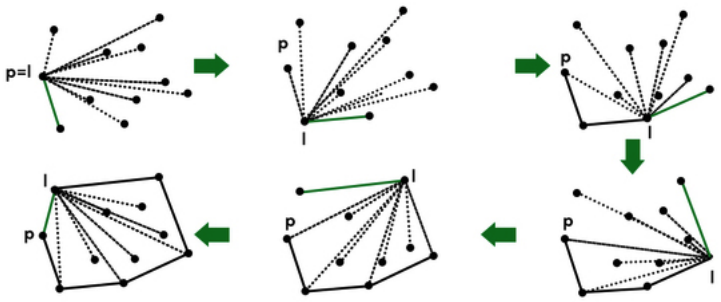


Figure 25: Esecuzione Wrapping Algorithm

Quindi, attraverso questa fase, come ben deducibile, si otterranno in output, solo per i cluster rilevanti, i poligoni che delimitano i generici insieme di punti computati dallo step precedente ottenendo, così, delle delimitazioni alle aree di cambiamento ritrovate per il processo di change detection proposto.Questo conclude il perseguimento dell'obiettivo principale della tesi in quanto le aree di cambiamento per la data zona di interesse sono state ritrovate e delimitate ed apre le porte all'obiettivo

secondario della tesi, ossia quello di supportare l'aggiornamento dei dati presenti nella base di dati preesistente per le opportune aree di cambiamento rilevate e delimitate in quest'ultima fase attraverso di metodi di restituzione automatica 3D.

3.2 Aggiornamento della base di dati e descrizione di metodi di restituzione automatica 3D

Come abbiamo già accennato nel paragrafo precedente, successivamente ad aver ottenuto le aree di cambiamento rilevanti con gli opportuni poligoni frontiera, la fase successiva prevederà l'aggiornamento dei dati preesistenti nella base di dati in input per le date aree di cambiamento rilevate.Parlando di aggiornamento di dati spaziali, oggigiorno, sono ancora presenti moltissime limitazioni dovute soprattutto alla mancanza di metodi o processi che permettano di aggiornare automaticamente dati relativi a determinate zone terresti ed al fine di proporre un supporto a questa fase si descriveranno due innovativi processi di restituzione automatica 3D, sviluppati in quest'ultimo anno e presenti in letteratura, applicabili alle zone di cambiamento rilevate dal nostro metodo e utili per poter acquisire automaticamente quanti più dati significati possibili correlati agli oggetti presenti nell'area da aggiornare, associando al generico poligono delimitante l'area di cambiamento ritrovata un modello 3D accurato e preciso nei dettagli rappresentante l'area stessa.Infatti, l'obiettivo qui sarà proprio quello di poter restituire con accuratezza, grazie ai metodi di restituzione automatica 3D che vedremo successivamente, un modello 3D, da cui poter estrarre automaticamente, attravero operazioni come classificazione o segmentazione, dati utili associabili agli oggetti presenti per la data zona di cambiamento rilevata per cui stiamo effettuando l'aggiornamento dei dati preesistenti che ci aiuteranno proprio ad automatizzare l'aggiornamento di parti di dati che verrà svolto.Questa fase di ricostruzione ed associazione alle aree di cambiamento ritrovate, per cui effettuare l'aggiornamento, di un nuovo modello 3D accurato e dettagliato, da cui poter estrarre automaticamente dati significativi relativi agli oggetti presenti, permette di ridurre l'intervento manuale richiesto per svolgere l'aggiornamento dei dati, di semplificare l'operazione stessa di aggiornamento e di abbassare il tempo necessario richiesto per completare quest'ultima. Il processo applicato per supportare l'aggiornamento dei dati in questa fase è il seguente:

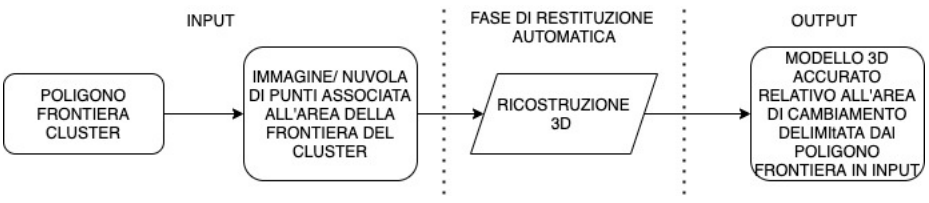


Figure 26: Supporto al processo di aggiornamento dei dati

I metodi di ricostruzione 3D applicabili in questa fase alle aree di cambiamento ritrovate possono essere differenti, noi ci soffermeremo sulla descrizione di due metodi innovativi, sviluppati nell'ultimo anno e che apportato delle migliorie allo stato

dell'arte della ricostruzione 3D e restituzione automatica in termini di precisione e accuratezza dei risultati ottenuti .Il primo metodo che verrà descritto dettagliatamente è un framework, un' innovativa libreria chiamata PyTorh3D creata appositamente per accellerare il processo di riconoscimento e restituzione 3D, il secondo è invece un metodo innovativo basato sull'utilizzo di una rete encoder-decoder abbinata ad una rete neurale per performare i task sempre di ricostruzione e restituzione 3D automatica.

3.2.1 PyTorch3D

PyTorch3D è un'innovativa libreria open-source basata su PyTorch[1] creata dal team di Facebook AI Research e composta da operatori modulari, efficienti e differenziabili per il Deep Learning di oggetti 3D. Tale libreria dispone di un renderizzatore modulare differenziabile per mesh,immagini e nuvole di punti molto più performante in termini di scalabilità (possibile processare grandi immagini o mesh), velocità e memoria degli altri presenti oggi giorno sul web ottenendo miglioramenti sino al 10x e permettendo anche approcci di analisi per sintesi. PyTorch3D verrà anche utilizzata per aggiornare e migliorare lo stato dell'arte delle previsioni non supervisionate di mesh 3D e di nuvole di punti da immagini 2D su ShapeNet[12] mantenendo un elevato throughput computazionale. Inoltre è utile ricordare al lettore che il progetto Pytorch3D è nato poichè il Deep Learning 3D rimane ancora oggi poco studiato e conosciuto rispetto al 2D a causa delle sfide ingegneristiche previste, infatti nel campo 3D avremo la presenza di una vastità e varietà di *dati eterogenei* i quali renderanno difficile implementare in modo efficiente le operazioni in batch utilizzando le primitive fornite dai toolkit standard di deep learning come PyTorch[1] ed inoltre ogni operazione che riguarderà il processo di dati 3D o l'utilizzo di metodi atti a processare dati 3D dovrà essere rivisitata al fine di poter calcolare in modo efficiente i gradienti ed essere introdotta nelle pipiline del Deep Learning.

3.2.1.1 PyTorch3D:componenti e caratteristiche

Come già accennato PyTorch3D è una libreria open-source basata su PyTorch e composta da:

- operatori modulari, efficienti, veloci e differenziabili
- kernel CUDA opportunamente personalizzati al fine di minimizzare l'utilizzo della memoria
- strutture dati riutilizzabili per la gestione di batch di nuvole di punti e mesh. Esse permetteranno conversione tra diverse tensor-based representations (lista, impacchettata, imbottita) necessarie per varie operazioni, e quindi, agli operatori di poter supportare batch di dati eterogenei
- motore di rendering modulare, efficiente e differenziabile per meshes e nuvole di punti. Tale motore proietta i dati 3D su immagini 2D permettendo le operazioni di analisi per sintesi[2] e di rendering inverso[3]

Il renderizzatore (modulare) di PyTorch3D utilizzato è ispirato a quello di SoftRas[4], il quale, propone un rendering differenziale in cui l'operazione di rasterizzazione è vista come un processo probabilistico dove il colore di ogni pixel dipende da più facce della mesh.A differenza di SoftRas, però, nel renderizzatore di PyTorch3D, gli *intermedi* calcolati dal processo di rasterizzazione verranno riprogettati ed esibiti. L'uso del rendering differenziale permette sia la predizione non supervisionata della forma 3D mediante perdite di riproiezione 2D che la personalizzazione, da parte degli utenti, delle pipiline di rending tramite gli shaders di Pytorch3D[1].

3.2.1.2 PyTorch3D:Funzionalità e prestazioni

In questa sezione verrà presentato un benchmark con le mesh di ShapeNetCoreV1[12] (le nuvole di punti verranno create campionando uniformemente dalle superficie delle mesh) sulla velocità e l'utilizzo della memoria dei principali operatori di PyTorch3D, confrontandoli con PyTorch e con le implementazioni open-source esistenti. Sarà osservabile che PyTorch3D presenterà delle migliorie sia su velocita che memoria, del 10X rispetto agli altri metodi. Tutti i risultati sono mediati su 5 batch casuali e 10 esecuzioni per batch, e sono eseguiti su una GPU V100.

Operatori 3D

- *Chamfer loss*, metrica che quantifica la corrispondeza tra la nuvola di punti P e Q:

  $$Lcham(P,Q) = \left|Q\right|^{-1} \sum_{(p,q)\in \wedge P,Q} \left\|(p-q)\right\|^2 + \left|P\right|^{-1} \sum_{(p,q)E\wedge P,Q} \left\|(q-p)\right\|^2$$

  dove $\wedge P,Q = (p, argmin_q \left\|p-q\right\|)$: $p \in P$ è l'insieme delle coppie *(p,q)* tali che $q \in Q$ è il vicino più vicino di $p \in P$.

  Come visibile dalla Figura 27 sottostante, confrontando la *Chamfer Loss*, in termini di tempo e memoria richiesta, di PyTorch3D con quella di PyTorch è visibile che l'approcio banale esaurisce la memoria per $|Q| > 10k$, mentre l'approccio innovativo di PyTorch3D si adatta perfettamente alle grandi dimensioni delle nuvole di punti ottenendo un risparmio di tempo e memoria di circa 12X rispetto a PyTorch.

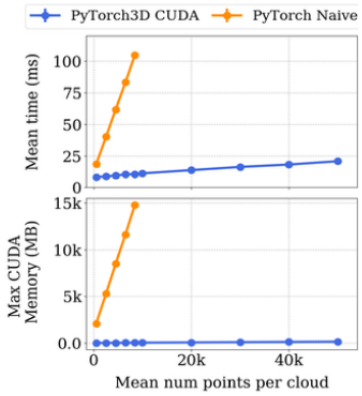


Figure 27: Chamfer loss.Lcham(P, Q) per nuvole di punti con |P| = 1000 ed eterogeneo, ed un valore |Q| variabile.

- *Graph convolution*[5], metrica comunemente utilizzata per processare mesh 3D. Siano dati i vettori di feature f_v per ogni vertice v allora si calcoleranno i nuovi vettori di feature attraverso la seguente formula:

$$f'_v = W_0 f_v + \sum_{u \in N(v)} W_1 f_v$$

  Dove *N(v)* rappresentano i vicini di v e W_0, W_1 sono le metrici dei pesi apprese durante il processo. Di seguito la Figura 28 mostrerà il confronto, in termini di tempo richiesto e memoria utilizzata, tra PyTorch3D e PyTorch per il calcolo della Graph Convolution. Da notare che PyTorch3D implementerà la convoluzione grafica tramite un kernel CUDA fuso ed otterrà un miglioramento del 30% rispetto all'implementazione pura di PyTorch.

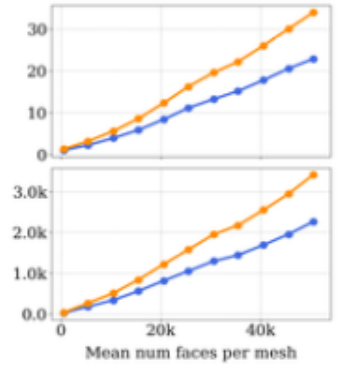
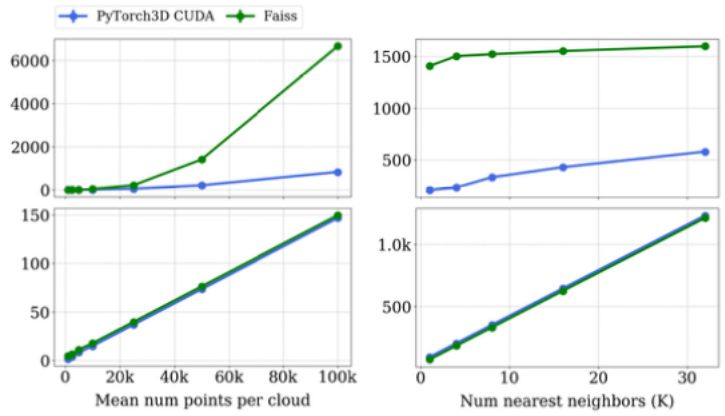


Figure 28: Graph convolution su batch di mesh eterogenee con features a 128-dimensioni.

- *K Nearest Neighbors*, metrica utilizzata per calcolare i vicini più vicini per i punti D-dimensionali. Il KNN su PyTorch3D è implementato tramite dei kernel CUDA personalizzati che permettono l'elaborazione di batch eterogenei,

inoltre è reso ottimizzato per valori di $D \leq 4$ e $K \leq 32$.La Figura 29, di seguito, confronta il KNN di PyTorch3D con il KNN di Faiss[6], una libreria GPU veloce per il *KNN*.Quest'ultimo non gestisce il batching di dati eterogenei ma è progettato e ottimizzato per valori di D abbastanza alti e per elaborare miliardi di punti.



(c) KNN (varying $|P|$) (d) KNN (varying K)

Figure 29: KNN di PyTorch3D vs Faiss,si considerano batch omogenei 3D di nuvole di punti P and Q, con |P | = |Q|.In (c) K = 1, and in (d) |P |=|Q|=50k;in entrambi l'utilizzo della memoria è ottimale.

I risultati ottenuti mettono in luce PyTorch3D con un miglioramento rispetto a Feiss[6] del 5X per i problemi del 3D in batches di dati eterogenei.

Renderizzatore differenziabile di mesh

Un renderizzatore consente di ottenere una rappresentazione /immagine digitale (detta render) attraverso l'operazione di rendering, la quale indica proprio il processo di "resa" grafica ovvero di generazione di un'immagine digitale a partire da una descrizione matematica di una scena tridimensionale.Il generico input per l'operazione di rendering è composto da camera, geometrie , materiali , luce e texture, le quali compongono le informazioni della scena tridimensionale.Mentre un renderizzatore differenziale permette l'integrazione dell'operazione di rendering nelle pipiline del Deep Learning[7][8] propagando i gradienti degli oggetti 3D all'indietro dalle immagini alle informazioni della scena tridimensionale interessata[9].Il renderizzatore utilizzato da PyTorch3D, lavora su batch eterogenei di mesh triangolari. Esso, come già accennato in precedenza, possiede tre caratteristiche fondamentali:

- *differenziabilità*: permette di calcolare i gradienti rispetto a tutti gli input
- *efficienza*: è rapido e riesce ad elaborare grandi mesh e immagini (scalabilità)
- *modularità*: gli utenti possono facilmente sostituire i componenti del renderer per personalizzare le sue funzionalità al loro caso d'uso e sperimentare formulazioni alternative.

Al fine di poter migliorare notevolmente efficienza e modularità e quindi apportare delle migliorie rispetto ai renderizzatori esistenti[10][11], è opportuno procedere ad un'attenta progettazione, come mostrato nella Figura 30, delle due componenenti principali del renderizzatore di PyTorch3D, *rasterizzatore* che seleziona le facce della mesh che influenzano ogni pixel e *shader* che calcolano il colore di ogni pixel.

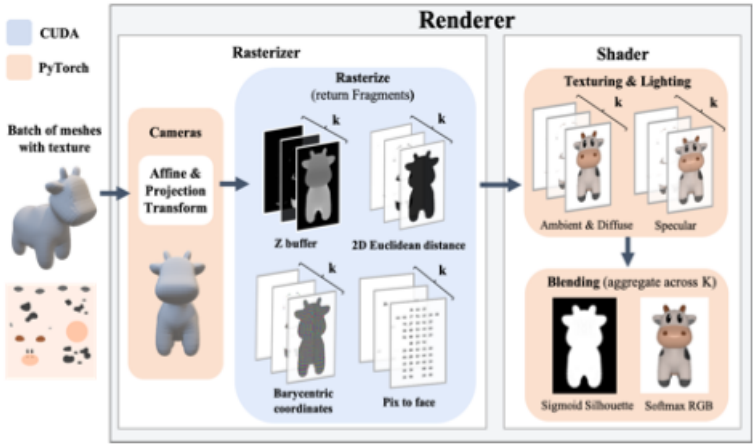


Figure 30: Pipeline del rendering modulare di PyTorch3D

La descrizione relativa alla progettazione del rasterizzatore e dello shader di PyTorch3D non saranno presenti in questo documento, in quanto, al di fuori degli obiettivi di tesi proposti, ma andremo ad analizzare i risultati ottenuti confrontando il renderizzatore differenziabile di PyTorch3D con quello di SoftRas[4] così da permettere al lettore di canstatare le migliorie apportate dal nuovo renderizzatore di PyTorch3D nel deep learning 3D e nella restituzione automatica 3D. La figura sottostante, Figure 31, mostra i risultati ottenuti dal confronto con SoftRas.

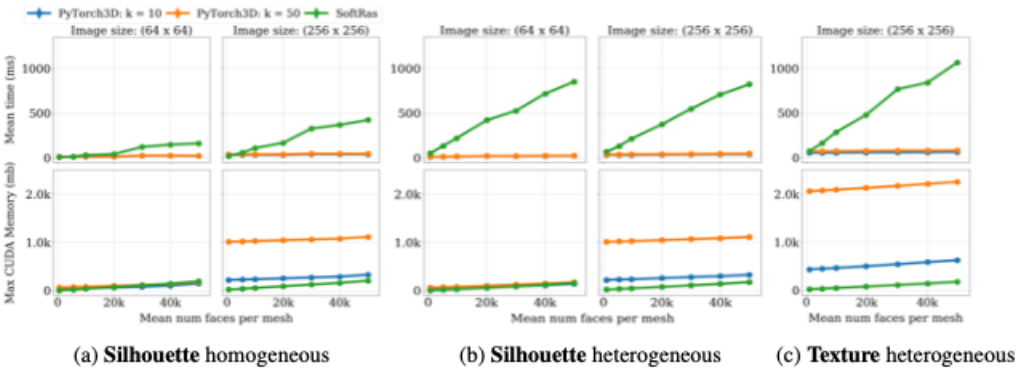


Figure 31: Benchmarks di silhouette rendering e textured rendering per PyTorch3D and SoftRas[4].

Si sono implementati, utilizzando texture per-vertex e Gouraud shading, degli shader per simulare, con PyTorch3D, i *silhoutte renderign* e i *texture mesh rendering* di SoftRas[4].E' stato osservato che PyTorch3D è notevolmente più veloce

per grandi mesh, immagini ad alta risoluzione e batch eterogenei. Il renderizzatore di PyTorch3D utilizza, però, più memoria GPU rispetto a SoftRas in quanto esso memorizza esplicitamente i dati Fragment. Malgrado questo, l'utilizzo assoluto della memoria(2GB per la texture a 2562) rimane più piccolo rispetto alla capacità delle moderne GPU (32GB per V100), quindi è possibile affermare che la migliore modularità di PyTorch3D compensi l'alto utilizzo della memoria.

Renderizzatore differenziabile di nuvole di punti

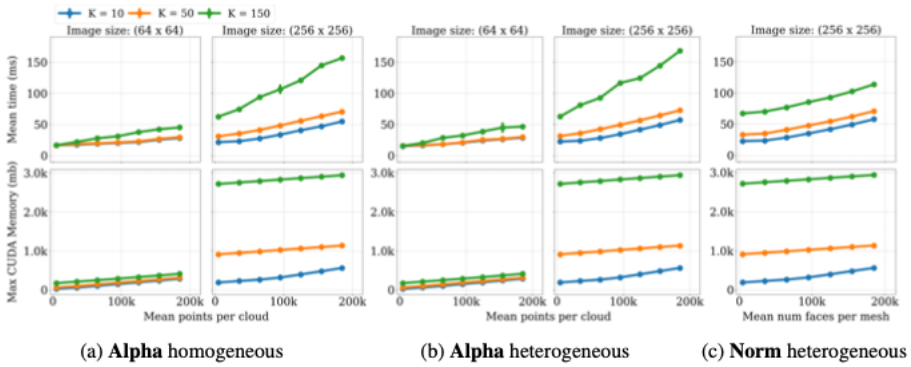
PyTorch3D fornisce, oltre al renderizzatore per mesh anche un renderizzatore, modulare ed efficiente, per nuvole di punti. Tale renderer è molto simile a quello utilizzato per le mesh, formato da un *rasterizzatore* il quale permettere di calcolare i K punti più vicini ad ogni pixel lungo l'asse z e da *shader* che consumano i dati forniti dal rasterizzatore e computa il colore di ogni pixel. Tale renderizzatore, ovviamente, permette di processare batch di dati eterogenei ed inoltre per il suo studio ed analisi si sono utilizzati shader per il rendering delle nuvole di punti con silhoutte e texture.

Negli esperimenti proposti si considerano due metodi di "unione", *Alpha-compositing* e *Normalized weighted sums*. Supposto che un pixel sia sovrapposto a K punti con opacità $\alpha_1, ..., \alpha_K \in [0, 1]$ e che codesti punti siano associati ai vettori di features $f_1, ..., f_K \in R^D$ allora sarà possibile calcolare le features (booleane per il rendering della silhoutte, colori RGB per il rendering delle texture oppure features neurali) nei segunti modi:

- $f\_Alpha = \sum_{i=1}^{K} (\alpha_i \prod_{j=1}^{w} 1 - \alpha_j) f_i$, dove $w = i - 1$.
- $f\_Norm = (\sum_{i=1}^{K} \alpha_i f_i) / (\sum_{i=1}^{K} \alpha_i)$

Sia f_Alpha che f_Norm sono funzioni di "unione" differenziabili.Esse permettono la propagazione all'indietro dei gradienti delle features dei pixel alle caratteristiche dei punti e all'opacità.

Di seguito verranno valutate le prestazioni del renderizzatore di nuvole di punti, utilizzato da PyTorch3D, in termini di rendering di silhouttes delle due funzioni di "unione", dove i punti delle nuvole sono stati ottenuti campionando punti dalla superficie di mesh randomiche di ShapeNet[12].Negli esperimenti proposti si è variata la dimensione della nuvola di punti, il numero di punti per pixel (K = 10, 50, 150), e la dimensione dell'immagine (64, 256).



(a) **Alpha** homogeneous (b) **Alpha** heterogeneous (c) **Norm** heterogeneous

Figure 32: Benchmarks per il render di nuvole di punti di PyTorch3D con l'utilizzo di Alpha weighted e Norm weighted .

Il renderizzatore per nuvole di punti, utilizzato da PyTorch3D, risulta essere, dai dati ottenuti, efficiente e migliore in termini di prestazione del renderer utilizzato per processare mesh. Come visibile dalla figura sovrastante, Figure 6, l'operazione di rendering su un batch di 8 nuvole di punti aventi circa 200k di punti ciascuno richiede 75 ms e utilizza poco più di un 1GB di memoria GPU rendendo fattibile la differenziabilità nel renderer stesso su reti neurali. Inoltre confrontanto (a) e (b), quindi l'applicazione del rendering, utilizzando *f_Alpha*, su batch omogenei ed eterogenei di simile dimensione si avranno prestazioni comparabili.Mentre, confrontando (b) e (c), che utilizzano le funzioni (a) *f_Alpha* e (b) *f_Norm* si nota un utilizzo simile di memoria ma *f_Norm* risulta essere 25% più veloce per valori grandi di K.

3.2.1.3 PyTorch3D:Esperimenti

In questa sezione verranno esaminati alcuni esperimenti relativi alla predizione non supervisionata della forma 3D di PyTorch3D, tralasciando il calcolo della forma 3D attraverso la supervisione 2D in quanto operazione fattibile grazie all'utilizzo del rendering differenziale. I risultati ottenuti e osservabili verranno confrontati, in un secondo momento, con SoftRas[4] dimostrando che PyTorch3D possiede una velocità e una predizione della forma dell'oggetto migliori. Inoltre, l'utilizzo del nostro renderizzatore differenziabile ci permetterà di poter scalare su una mole di dati molto più ampia rispetto ad altri renderer e quindi la possibilità di elaborare grandi e complesse mesh tramite PyTorch3D ha permesso un grande passo avanti nell'aggiornamento dello stato dell'arte per la predizione di forme 3D non supervisionate.

Durante lo svolgimento degli esperimenti, i modelli utilizzati predicono la forma 3D di un oggetto (mesh/nuvola di punti) da una singola immagine RGB senza ricevere, durante l'addestramento, alcuna supervisione 3D, ma basandosi sulle perdite di riproiezione[9] ottenute dal rendering differenziale.

Dataset

Gli esprimenti ed i risultati proposti sono stati ottenuti lavorando su ShapeNetCoreV1[12], utilizzando le immagini renderizzate e le suddivisioni train/test. Le

immagini sono 137×137, e ritraggono istanze di 13 categorie di oggetti da vari punti di vista. Inoltre, sono presenti circa 840K immagini di training e 210K immagini di test ed il 5% delle immagini di training verranno riservate per il processo di validazione.

Metriche utilizzate

Per la valutazione delle mesh 3D si utilizzano dei metodi già conosciuti in letteratura[13][14]. Si sono campionati 10k di punti, in modo random ed uniforme, dalle superficie delle mesh previste (sono quelle che verranno predette) e dalle mesh vere (mesh esatte, sono quelle che non verranno predette) e successivamente verranno confrontati utilizzando la *distanza di Chamfer*, la *consistenza normale*, e lo score F_1^θ per diverse soglie di distanza θ.

Utilizzo del rendering differenziale per la predizione di mesh

Analizzeremo i risultati ottenuti dal task di predizione di mesh di oggetti 3D da immagini 2D con supervisione di silhoutte 2D. Per la predizione di mesh si è utilizzata una configurazione di addestramento a due viste dove per ogni immagine oggetto presente sul minibatch verrà inclusa, anche, una seconda vista ottenuta applicando una trasformazione casuale. Durante il test, tutti i modelli utilizzati prendono come input una singola immagine e restituiscono in output la predizione dell'oggetto 3D sotto forma di mesh in coordinate di telecamera. I modelli utilizzati durante gli esperimenti di PyTorch3D sono:

- *Sphere FC*, modello matematico che impara a deformare una sfera iniziale avente 642 vertici e 1280 faccie.Qui l'immagine in input è codificata attraverso una backbone *CNN* seguita da due layer completamente connessi i quali generano previsione di offset per ogni vertice presente nella sfera iniziale

- *Sphere GCN*, modello utilizzante la *Graph Convolution*, molto simile a *Sphere FC* in quanto impara a deformare una sfera (mesh sferica) di grandi dimensioni.Qui ogni vertice raggrupperà le proprie features dall'output della backbone indicizzata dalla sua proiezione 2D e successivamente applicando un'insieme di *Graph Convolution* sulla mesh sferica iniziale sarà possibile predire gli offset per ciascun vertice

- *Voxel GCN*, modello diverso dai primi due i quali prevedevano solo forme simili a sfere.Il modello *Voxel GCN* implementato utilizza predizioni voxel grossolane per catturare le topologie degli oggetti per l'instanza considerata e successivamente raffina il risultato attraverso l'utilizzo di una sequenza di *Graph Convolution* producendo una predizione di alta qualità. Questo modello è stato implementato per dimostrare la flessibilità di PyTorch3D nel predire diverse topologie di forme e lotti eterogenei ed è basato sul modello sviluppo in [13].

Tutti i modelli considerati per gli esperimenti utilizzano una backbone ResNet50 [15], vengono inizializzati con i pesi di ImageNet ed in input prenderanno immagini

137x137 da [16]. Per quanto concerne l'apprendimento, i modelli utilizzeranno tutti Adam, con un learning rate sui 10-4 per 25 epoche.

L'immagine proposta di seguito, Figura 33, riporta il confronto di prestazioni del renderizzatore di mesh utilizzato da Pytroch3D e quello utilizzato da SoftRas[4].

Model	3D Superv.		Renderer		Metrics					Mesh Size	
Net	Vox	Mesh	Size	Engine	Ch. (↓)	Nrml	$F_1^{0.1}$	$F_1^{0.3}$	$F_1^{0.5}$	$\|V\|$	$\|F\|$
Sphere FC	✗	✗	64	SoftRas	1.475	0.691	25.5	68.4	82.3	642	1280
	✗	✗		PyTorch3D	0.989	0.696	26.4	69.9	83.5	642	1280
	✗	✗	128	SoftRas	0.346	0.700	26.1	70.6	85.2	642	1280
	✗	✗		PyTorch3D	0.313	0.699	**27.6**	72.5	86.6	642	1280
Sphere GCN	✗	✗	64	SoftRas	0.316	**0.713**	24.4	70.2	85.8	642	1280
	✗	✗		PyTorch3D	0.296	0.703	24.8	71.3	86.5	642	1280
	✗	✗	128	SoftRas	0.301	0.709	26.1	71.9	86.5	642	1280
	✗	✗		PyTorch3D	0.293	0.709	26.6	72.6	86.9	642	1280
High Res Sphere GCN	✗	✗	128	PyTorch3D	**0.281**	0.696	26.7	**73.8**	**87.8**	2562	5120
Voxel GCN	✓	✗	64	SoftRas	0.293	0.656	24.5	71.1	87.2	1947$_{\pm 923}$	3895$_{\pm 1851}$
	✓	✗		PyTorch3D	**0.267**	0.675	26.1	73.3	**88.5**	1932$_{\pm 935}$	3866$_{\pm 1873}$
	✓	✗	128	SoftRas	0.276	0.675	**26.2**	72.6	87.9	1918$_{\pm 928}$	3837$_{\pm 1860}$
	✓	✗		PyTorch3D	0.277	**0.687**	**26.2**	**73.4**	87.8	1951$_{\pm 949}$	3903$_{\pm 1901}$
Voxel Only [14]	✓	✗	n/a	n/a	0.916	0.595	7.70	33.1	54.9	2433$_{\pm 925}$	4877$_{\pm 1856}$
Mesh R-CNN [14]	✓	✓	n/a	n/a	0.171	0.713	35.1	82.6	93.2	2292$_{\pm 902}$	4598$_{\pm 1812}$

Figure 33: Ricostruzione mesh tramite silhouette rendering su ShapeNet di PyTorch3D e SoftRas. Si comparano i risultati con lo stato dell'arte di Mesh R-CNN.I modelli sono stati allenati con supervisione di voxel e mesh.In blu sono evidenziati i migliori risultati senza supervisione 3D e in rosso i migliori con supervisione voxel.

I risultati ottenuti mostrano che :

- PyTorch3D, rispetto a SoftRas[4] presenta prestazioni migliori o pari in tutti i modelli utilizzati.
- *Sfere GCN* risulta essere migliore rispetto a *Sfere FC* al fine di ottenere una predizione della forma di qualità.Tale affermazione è visibile dalla figura seguente, Figura 34, dove qualitativamente la predizione sulla forma fatta da *Sfere GCN* risulta essere di migliore qualità rispetto a quella di *Sfere FC*.

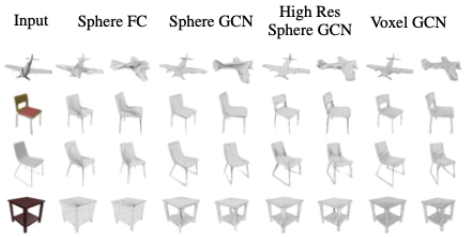


Figure 34: Predizione mesh tramite silhouette su ShapeNet.Si mostra l'immagine di input, la predizione ed una vista aggiuntiva per ogni modello.

- *High Resolution Sfere GCN* migliora drasticamente sia in termini qualitativi che quantitativi *Sfere GCN* mostrando l'alta scalabilità di PyTorch3D.
- *Voxel GCN* supera drasticamente, in termini di prestazioni *Voxel Only*.Anche se utilizzano entrambi la medesima supervisione 3D, *Voxel Only* producendo delle previsioni grossolane non riesce a catturare le forme sottili a differenza di *Voxel GCN* che come mostrato in Figure 8, migliora tutte le metrice e riesce a ricostruire strutture fini e complesse.
- PyTorch3D, grazie alla propria efficienza, ha migliorato notevolmente il tempo richiesto per il training dei dati.Se confrontanto con quello di SoftRas risulta essere 2 volte più veloce.

Successivamente, sono stati analizzati altri risultati ottenuti variando il valore K, ovvero il numero di facce più vicine per ogni pixel ritrovate dal nostro renderizzatore.E' stato dimostrato che i miglior risultati riguardanti le predizione per i modelli *Sfere GCN* e *Voxel GCN* sono stati ottenuti mantenedo un valore costante di K, nè troppo piccolo, nè troppo grande.

Infine, sono stati analizzati risultati ottenuti dalla sperimentazione della predizione e restituzione delle texture degli oggetti di una generica instanza considerata.Al fine di poter ottenere codesti risultati, i modelli precedentemente citati sono stati estesi per poter predire i valori *(r,g,b)* per ogni vertice della mesh considerata utilizzando il rendering testurizzato. Le due immagini sottostanti, Figura 35 e Figure 36, mostrano i risultati quantitativi e qualitativi ottenuti dall'analisi delle texture.

Model	Renderer		Metrics						
Net	Size	Shading	Chamfer (↓)	Normal	$F_1^{0.1}$	$F_1^{0.3}$	$F_1^{0.5}$	L_1^{fg} (↓)	L_1^{bg} (↓)
Sphere GCN	64	Flat	0.315	0.689	24.1	**70.9**	86.3	0.0217	0.0031
		Phong	0.309	**0.703**	24.1	70.2	85.9	**0.0173**	**0.0023**
		Gouraud	**0.302**	0.702	**24.4**	**70.9**	**86.6**	0.0180	0.0024
Voxel GCN	64	Flat	**0.270**	0.678	**26.4**	**73.0**	**88.2**	0.0128	**0.0020**
		Phong	0.272	**0.694**	25.7	72.4	87.9	**0.0126**	0.0021
		Gouraud	0.302	0.687	24.3	69.8	86.2	0.0128	0.0021

Figure 35: Ricostruzione mesh e texture tramite textured rendering su ShapeNet.Le ultime due metrice indicano la precisione di ricostruzione dell'oggetto in primo piano e dello sfondo.

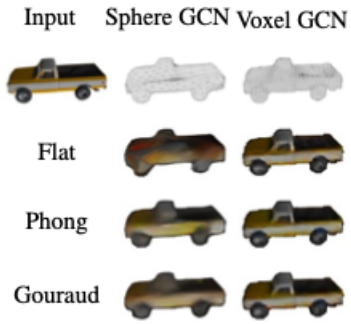


Figure 36: Texture mesh rendering

L'analisi dei risultati ottenuti constata un'alta qualità di ricostruzione delle texture è quindi possibile affermare che il renderizzatore differenziabile per mesh di PyTorch3D abbinato ai modelli *Sfere GCN* e *Voxel GCN* possiede sia un'alta qualità di ricostruzione per le forme (come visibile in Figura 33) che per le texture.

Utilizzo del rendering differenziale per la predizione di nuvole di punti

Al fine di poter mostrare l'efficacia del rendirizzatore di nuvole di punti di PyTorch3D verranno addestrati modelli di nuvole di punti non supervisionati. Il modello utilizzato per gli esperimenti è *Point Align*, esso deforma 10k di punti, ottenuti da una sfera, tramite il raggruppamento delle caratteristiche della backbone e la predizione degli offset per ogni punto considerato. E' dimostrabile che il modello *Point Align* abbinato al renderizzatore differenziale di PyTorch3D per nuvole di punti migliora leggermente le metriche sulle predizione della forma rispetto al renderizzato per mesh e che i migliori risultati, relativi alle metriche considerate, sono ottenuti mantenendo un valore costante di K, nè troppo basso, nè troppo alto. Inoltre, come per il renderizzatore di mesh, si sono ricostruite le texture prevedendo, tramite rendering texturizzato, i valori *(r,g,b)* per ogni punto considerato.

La figura sottostante, Figura 37, mostra i risultati qualitativi delle predizione delle nuvole di punti sia per la forma che per le texture.

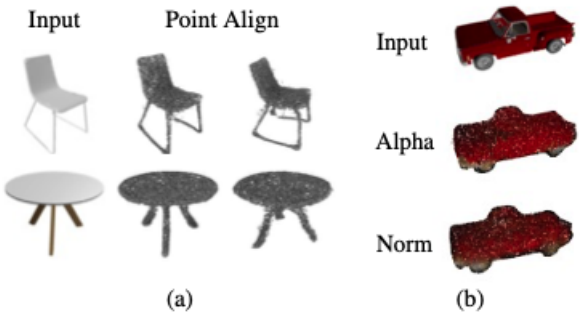


Figure 37: Predizione nuvola di punti tramite (a)silhoutte e (b)texture rendering su ShapeNet

Infine, le performance del modello *Point Align* utilizzato su PyTorch3D sono state confrontate con le performance di *PSG*[17], un modello supervisionato utilizzato nel 3D.Analizzando i risultati mostrati in Figure 12, *Point Align* migliora significamente la metrica F_1, ma peggiora, essendo un modello non supervisionato, la distanza di *Chamfer*, in quanto *PSG*[17] minimizza tale metrica tramite supervisione 3D.

Net	3D superv	Ch.(↓)	F_1^τ	$F_1^{2\tau}$
PSG [11]	✓	**0.593**	48.6	69.8
Point Align	✗	0.647	**61.0**	**74.6**

Figure 38: Confronto tra il modello supervisionato di nuvola di punti, Point Align utilizzato da PyTorch3D e PSG

Con questo concludiamo la panoramica generale di PyTorch3D. Tale "libreria" sarà uno dei fondamenti del futuro Deep learning 3D, dovrà essere migliorata ed approfondita, anno dopo anno , dalla community, al fine di poter accellerare l'intersezione tra 3D e Deep Learning. Per un maggiore approfondimento riguardante PyTorch3D si rimanda il lettore a [1].

3.2.2 3D-RVP

3D-RPV è un metodo di ricostruzione 3D da una singola vista di profondità dove *R*, *P* e *V* stanno rispettivamente per *ricostruzione*, *point-cloud* e *voxel*. Si tratta di un innovativo metodo di ricostruzione 3D, a due fasi, ad alta risoluzione dove vengono utilizzate due reti, la prima è una rete di codifa-decodifica 3D che permette l'apprendimento e la produzione di risultati di predizioni grossolane, mentre la seconda è una rete utilizzata per la predizione di punti in cui è stato implementato un opportuno algoritmo di suddivisione iterativa al fine di poter predire, in modo ottimale, le etichette dei punti selezionati. Nella prima fase, 3D-RPV converte un'immagine di profondità in una griglia di voxel e la inserisce come input nella rete codifica-decodifica 3D.La rete codifica la griglia di voxel in un vettore e successivamente decodifica tale vettore alla forma 3D più probabile.L'output ottenuto avrà una risoluzione di 64x64x64, inoltre ad ogni voxel verrà assegnato un valore, tra 0 e 1, che indica la probabilità che sia occupato, se tale valore fosse vicino a 0.5 allora l'incertezza è alta. Nella fase successiva 3D-RPV campiona, dalla distribuzione ottenuta, i voxel aventi incertezza alta, tramite un algoritmo di interpolazione ottiene, dalle mappe delle caratteristiche della rete encoder-decoder, le caratteristiche puntuali di tali voxel e successivamente con l'utilizzo di un point head prevede la probabilità che siano occupati.Alla fine, combinando i risultati ottenuti dalle due rete otterremo un modello 3D preciso ed accurato. Quindi, i contributi sostanziali apportati da *3D-RVP* nello stato dell'arte del Deep Learning 3D e che esamineremo attraverso esperimenti e risultati ottenuti sono:

- rete codifica-decodifica 3D con apprendimento residuo al fine di poter ricorstruire, da una singola vista di profondita, un modello 3D completo.Codesta rete è facilmente ottimizzabile tramite l'aggiunta di una struttura residua sia al codificatore che e decodificatore
- rete di predizione puntuale al fine di poter rielaborare il modello 3D ottenuto precedentemente dalla rete di codifica-decodifica 3D e ottenere risultati ad alta risoluzione. L'implementazione di tale rete permetterà oltre ad ottenere modelli ad alta risoluzione anche di ridurre l'utilizzo di memoria
- superamento dell'attuale stato dell'arte del Deep Learning 3D, apportando un miglioramento di circa il 2,7% in termini di metrica di intersezione-su-unione(IoU)

3.2.3 Metodologia utilizzata da *3D-RVP*

In questa sezione verrà descritto il funzionamento di *3D-RVP* attraverso l'analisi delle reti che lo compongono. L'architettura del metodo viene mostrata nella figura sottostante, Figure 39, dove è osservabile proprio la combinazioni delle due reti precedentemente citate, quella di *codifica-decodifica 3D* e quella per la *predizione dei punti*. Inoltre, è opportuno per il lettore sapere che i modelli trattati ed elaborati dalle due reti si presentarenno sotto forma di nuvole di punti.

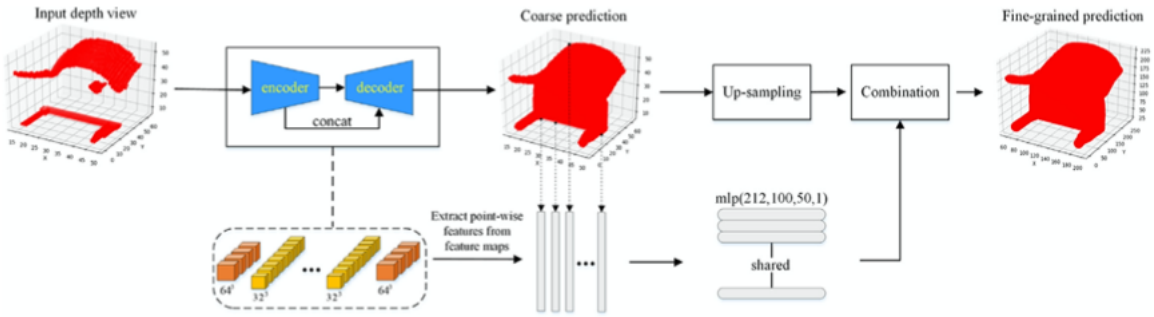


Figure 39: Architettura 3D-RPV

Rete Encoder-Decoder 3D

La rete encoder-decoder 3D utilizzata da *3D-RVP* è formata da un encoder e da un decoder con un apprendimento residuo come backbone. Considerando una generico modello 3D e scannerizzandolo da un singolo punto di osservazione,*3D-RVP*, elabora un'immagine di profondità, la quale viene convertita in una griglia di voxel utilizzata come input della rete *encoder-decoder 3D* che funge da rete neurale. L'encoder si occupa dell'estrazione delle caratteristiche della griglia di voxel in input al fine di ottenere una rappresentazione latente, mentre il decoder predice la forma 3D completa più probabile.La forma 3D ottenuta come output dalla combinazione dell'*encoder* e del *decoder* risulta essere accurata, ma grossolana. La figura successiva, Figura 40, mostra nel dettaglio l'archiettettura interna sia dell'encoder (lato sinistro) che del decoder (lato destro) formati entrambi da 18 blocchi.Qui ogni blocco costituisce un layer della rete neurale.

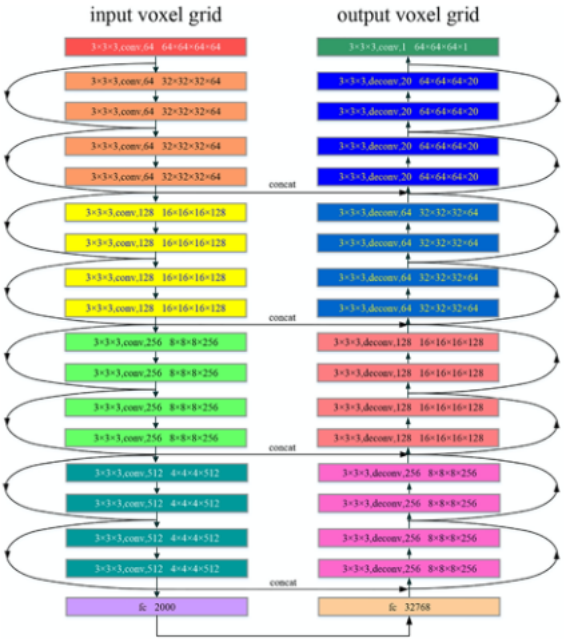


Figure 40: Architettura interna della rete encoder-decoder 3D

I valori associati al singolo layer sono utili al fine di performare l'operazione di convoluzione o deconvoluzione, essi rappresentano rispettivamente la dimensione del kernel di un layer di convoluzione 3D (*3x3x3*), l'operazione da performare (*conv* o *deconv*), il numero di canali di output (*64*) e la dimensione dell'output (*64x....x64*, valore variabile).Inoltre, la funzione di attivazione utilizzata nei layer è la *ReLu function* eccetto per l'ultimo layer del decoder dove viene utilizzata la *sigmoid function*.

L'encoder di *3D-RVP* prende in input,come accennato in precedenza, la *griglia di voxel*, il primo layer converte il numero dei canali di uscita a 64 tramite l'applicazione dell'operazione di convoluzione, i successivi 16 layers rappresentano una rete residua(*ResNet*) suddivisa in 4 sotto-blocchi di colore diverso dove in ogni sotto-blocco il numero dei canali di uscita viene raddoppiato e la risoluzione delle mappe delle features dimezzata al fine di incrementare il campo recettivo e non perdere nessuna informazione cruciale durante l'elaborazione.L'ultimo layer è un layer completamente connesso formato da 2000 nodi.

Il decoder prende in input l'output dell'encoder, il primo layer, un layer completamente connesso, converte il vettore latente (rappresentazione ottenuta dall'encoder) di 2000 dimensioni in uno a 32.768 dimensioni, i successivi 16 layer come nel caso precedente rappresentano una rete residua (*ResNet*) suddivisa in 4 sotto-blocchi differenziati dal colore, i quali performano l'operazione di deconvoluzione al fine di incrementare la risoluzione delle mappe delle features.L'ultimo layer, invece, permette la predizione grossolana della forma 3D completa tramite l'operazione di convoluzione. *Encoder* e *Decoder* sono connessi mediante l'operazione di *concatenazione*, questo avviene in quanto il campo recettivo, nei due, risulta essere variabile, infatti, se sul primo il campo recettivo si presenta come "piccolo" possedendo più informazioni locali, nel secondo il campo recettivo è più "grande" permettendo di mantenere più informazioni globali.Quindi l'operazione di concatenazione tra *encoder* e

decoder verrà performata su *3D-RVP* al fine di aggiunge all'output caratteristiche più dettagliate possibili.

Infine, *3D-RVP* aggiunge alla *rete encoder-decoder 3D* una struttura residua e seguendo [18] implementa l'apprendimento residuo su ogni sotto-blocco dell'encoder e del decoder permettendo l'ottimizzazione della rete in modo semplice anche qualora la rete neurale diventi ampliamente complessa.Ricordiamo al lettore che quando la rete neurale incomincia ad espandersi e diventare complessa sarà possibile estrarre features sempre più dettagliate migliorando notevolmente la precisione del metodo ma occorrerà sicuramente un metodo per ottimizzare la rete in modo semplice, ecco il motivo dell'utilizzo di una struttura residua.

Continuando con la descrizione , ogni sotto-blocco della *rete encoder-decoder 3D* che beneficia dell'apprendimento residuo possiederà due connessioni a taglio corto, il valore associato alla generica connessione a taglio corto e quindi l'output della struttara residua stessa è definito come y e verrà calcolato come segue:

$$y = F(x, W_i) + W_s x$$

con x input della struttura residua, $F(x, W_i$ layer multipli di convoluzione aventi W_i come parametri e W_s proiezione lineare utilizzata per matchare le dimensioni.

Rete di predizione dei punti

La *point prediction network* o anche detta *rete di predizione dei punti* è stata implementata all'interno di *3D-RVP* al fine di ottenere, dal modello grossolano 3D precedentemente predetto dalla *rete encoder-decoder*, un modello 3D più accurato ed ad alta risoluzione. La rete che analizzeremo in questa sotto sezione agisce solamente sui voxel/punti presenti lungo i confini del modello 3D ottenuto applicando l'operazione di interpolazione diretta, finalizzata ad ottenere un modello a maggiore risoluzione, al modello 3D grossolano in input.Il modello ottenuto successivamente all'applicazione dell'interpolazione diretta sarà sicuramente accurato internamente, ma grossolano lungo i confini in quanto i voxel/punti localizzati lungo il perimetro presentano variazioni notevoli coi propri vicini.Quindi, la *point prediction network* seleziona i voxel/punti lungo il confine della figura e per ognuno di essi predice le opportune etichette, definendo anche la probabilità che siano occupati. Dalla combinazione del modello 3D grossolano ottenuto dalla rete *rete encoder-decoder* e delle etichette ottenute dalla *rete di predizione dei punti* si otterrà un modello 3D, in formato nuvola di punti, sicuramente più accurato e avente alta risoluzione.

Come fa *3D-RVP* a ritrovare la probabilità di occupazione e quindi le etichette per un generico voxel/punto? E' opportuno per il lettore sapere che la rete di predizione utilizzata da *3D-RVP* è composta da:

- strategia per la selezione dei voxel/punti dove agire
- modulo utilizzato per l'estrazione delle features del punto selezionato
- un point head

In primis, la strategia di selezione permetterà di selezionare dal modello grossolano in input i voxel/punti con incertezza alta, cioè quei voxel/punti che presentano un valore associato prossimo allo 0.5.L'incertezza relativa all'occupazione di un voxel/punto, definita come V^*, può essere calcolata come segue:

$$V^* = 1 - |V_p - 0.5|$$

dove V_p rappresenta una predizione sulla probabilità che il voxel sia occupato.

Inoltre, le strategie di selezione utilizzate da *3D-RVP* per le fasi di inferenza e training della rete neurale saranno diverse, infatti, nell'inferenza verrà utilizzato un metodo di suddivisione adattivo ed iterativo dove verrà interpolando il modello grossolano ottenuto dalla rete precedente , verrà definito un threshold $\alpha(0.9)$, verranno selezionati tutti quei voxel/punti aventi valore di incertazza maggiore di $\alpha(0.9)$, per codesti punti verranno calcolate le coordinate del centro salvate come punti selezionati, e solo a questo punto, sui punti selezionati , verranno predette le opportune etichette.Successivamente alla predizione delle etichette il processo verrà ripetuto iterativamente finquando la risoluzione del modello non soddisfa i requisiti di precisione e alta risoluzione a cui *3D-RVP* mira. Mentre durante la fase di training della rete neurale, *3D-RVP* utilizza una strategia non iterativa basata su campionamento random.Al fine di selezionare M voxel/punti con alta incertezza verranno campionati in primis kM (k>1) punti dalla distribuzione uniforme(modello/forma 3D) ottenuta dalla *rete encoder-decoder*, per tali punti verranno calcolati i valore di incertezza mediante interpolazione trilineare sui valori grossolani predetti precedentemente e successivamente si selezioneranno da questi punti βM ($\beta \in (0, 1]$) punti aventi incertezza più alta.Infine, i rimanenti $(1-\beta)M$ punti verranno campionati randomicamente sempre dalla una distribuzione uniforme. Dopo aver selezionato i punti aventi incertezza alta, *3D-RVP* utilizzerà il modulo di estrazione delle features dei punti per ottenere dalle mappe delle features in input le caratteristiche dei punti selezionati.Il modulo implementato combina features dettagliate, ottenute dall'output del primo layer e del primo modulo residuo dell'encoder, e features grossolane, ottenute invece dall'output degli ultimi due moduli residui del decoder, con 212 canali ottenendo le caratteristiche puntuali per tutti i punti selezionati. A questo punto, *3D-RVP*, tramite l'utilizzo di un point head e la conoscenza delle caratteristiche puntuali precedentemente calcolate predirà le opportune etichette per i punti selezionati e combinando il modello grossolano con le nuove etichette otterrà una predizione del modello più dettagliato. Quindi, il modello in uscita dalla rete *point prediction network* sarà un modello 3D accurato ed ad alta risoluzione, ottenuto migliorando il modello 3D grossolano outputtato dalla *rete encoder-decoder* tramite rietichettatura di quei voxel/punti che presentavano incertezza alta di occupazione. Inoltre, oltre le reti descritte precedentemente, *3D-RVP*, utilizza, nella fase di rietichettatura dei punti incerti, una rete neurale *MLP* a 4 layer con l'unico scopo di condividere parametri proprio tra i diversi punti.

Loss Function

In *3D-RVP* la funzione di perdita, *loss function*, è data dalla somma della funzione di perdita della *rete encoder-decoder* e dalla funzione di perdita della *rete di predizione dei punti*.

Per il calcolo della *loss function* relativa alla *rete encoder-decoder* si utilizzerà una funzione di perdita cross-entropy che tenderà a risolvere il problema relativo alle etichette di categoria sbilanciate presenti nella griglia di voxel aggiungendo un parametro che penalizzerà maggiormente la situazione in cui il voxel avente valore 1 venga riconosciuto come 0.La funzione l_1, definita come la funzione di perdita della griglia di voxel non è altro che una media delle funzioni di loss dei singoli voxel e può essere calcolata come segue:

$$l_1 = -1/N \sum_{i=1}^{N} [\alpha y_i^* log y_i + (1-\alpha)(1-y_i^*) log(1-y_i)]$$

dove i rappresenta l'indice del voxel, $y_i \in (0,1)$ rappresenta la probabilità di predizione che l'i-esimo voxel sia occupato, y_i^* è invece il vero valore di occupazione dell'i-esimo voxel, N è il numero di voxel presenti nell griglia di voxel ed infine α rappresenta il paramametro di penalità settato a 0.85.

Invece, per quanto riguarda la funzione di perdita della *rete di predizione dei punti*, definita come l_2, verrà calcolata solo su M punti selezionati nel seguente modo:

$$l_2 = -1/M \sum_{i=1}^{M} [\alpha z_i^{\sim} log z_i + (1-\alpha)(1-z_i^{\sim}) log(1-z_i)]$$

dove i rappresenta l'indice del punto selezionati, z_i è la probabilità di predizione che l'i-esimo voxel della griglia sia occupato e $z_i^{\sim}$ l'etichetta stimata per z_i.

Quindi come detto inizialmente, in *3D-RVP* la funzione di perdita per l'intero modello è calcolata come somma delle funzioni di *loss* delle due reti:

$$loss = l_1 + \lambda l_2$$

dove λ non è altro che un parametro di regolarizzazione che permette di bilanciare le due funzioni di perdita l_1 e l_2 settato a 0.1. Inoltre, ricordiamo al lettore che tutti i parametri utilizzati nella rete neurale sono aggiornati da *3D-RVP* in modo dinamico tramite un algoritmo di backpropagation.

3.2.4 Esperimenti e risultati di 3D-RPV

In questa sezione verranno analizzati le migliorie apportate dal metodo *3D-RVP* nel campo della ricostruzione 3D con l'utilizzo del deep learning. Verranno descritti ambiente di lavoro, dataset e metriche di valutazioni adoperate durante gli esperimenti proposti e successivamente si evidenzieranno, grazie ad un semplice confronto di risultati ottenuti per il metodo proposto e per metodi simili, i benefici apportati in termini di performance da *3D-RVP*.I metodi adoperati per il confronto sono:

- 3D-EPN[19]
- 3D-RG[20]
- 3D-HSC[21]
- 3D-RecGAN++[22]

I risultati di questi 4 metodi, utilizzati per il confronto, sono stati ottenuti ed estratti da [22].

Ambiente di lavoro

Gli esperimenti condotti al fine di verificare le perfomance di *3D-RVP* sono stati su svolti su una workstation avente CPU Intel Xeon E5-2630v4 ed un processore grafico Titan V e sono di due tipi, il primo è un addestramento per categoria,mentre il secondo è un addestramento multi-categoria.

Dataset

Il dataset utilizzato in *3D-RVP* per il training della rete è quello adoperato in [22], estratto da ShapeNet e contenente singole viste di profondità e il corrispondente modello 3D completo.Verranno utilizzate 4 categorie per il training quali panca, sedia, divano e tavolo, per ognuno di queste categorie si sono scelti da ShapeNet, in modo random, 213 modelli e per ogni modello si sono campionati 5 diversi angoli di osservazione per rollio, beccheggio e imbardata.I dati utilizzati per i test,invece, sono generati da 4 categorie uguali al set utilizzato per il training della rete e per ogni categoria vengono selezionati da ShapaNet 37 modelli CAD al fine di ottenere due gruppi di prova, uno scansionato da 125 angoli di osservazione diversa, chiamato SV, ed un altro scansionato da 216 angoli di osservazioni chiamato CV.

Metriche utilizzate

Al fine di valutare la qualità della ricostruzione di *3D-RPV* ed anche degli altri approcci con cui esso verrà confrontato si utilizzano due metriche:

- intersection-over-union (IoU):per una griglia di voxel IoU è calcolabile attraverso la seguente formula:

  $$IoU = \sum_i [I(y_i > \gamma) * I(y_i^*)] / \sum_i [I(I(y_i > \gamma) + I(y_i^*))]$$

  dove i rappresenta l'indice dell'i-esimo voxel, y_i è il valore di predizione per l'i-esimo voxel,y_i^* è il reale valore associato all'i-esimo voxel, γ è il valore di soglia e $I(x)$ è una funzione uguale ad 1 se x è maggiore o uguale ad 1 altrimenti è uguale a 0.Il valore ottenibile di IoU è compreso tra 0 ed 1, più è grande questo valore migliore è la qualità della ricostruzione 3D.

- standard cross-entropy (CE) loss: rappresenta una funzione di perdita e per una griglia di voxel questa metrica è calcolabile attraverso la seguente formula

  $$CE = -1/N \sum_i [y_i^* log y_i + (1 - y_i^*) log(1 - y_i)]$$

  dove i, y_i e y_i^* sono definibili come per la IoU, N rappresenta il numero di voxel presenti nella grigia di voxel.

Risultati ottenuti

Al fine di ottenere i risultati di performance relativi alla qualità della ricostruzione per *3D-RVP* si sono condotti due esperimenti sia sui dati di test SV che sui dati di test CV addestrando, nel primo esperimento, le reti neurali su ogni categoria dei dati di training, quindi sulla categoria sedia,panchina, divano e tavolo, mentre nel secondo esperimento le reti neurali sono state addestrate su un set di dati di training multi-categoria, in cui troviamo sempre la categoria sedia, tavolo, divano e tavolo.Vedremo i valori di metriche ritornate dalle rete neurali utilizzate al fine di constatare l'efficacia e la qualità di ricostruzione di *3D-RVP*. Le Figure 41 e 42 sottostanti mostrano i risultati di Iou e CE loss ottenuti, dai modelli utilizzati per il confronto, dalla rete encoder-decoder di *3D-RVP* e da *3D-RVP* utilizzando i dati di test SV e CV.

Methods	IoU				CE			
	Bench	Chair	Couch	Table	Bench	Chair	Couch	Table
3D-EPN[9]	0.423	0.488	0.631	0.508	0.087	0.105	0.144	0.101
3D-RG[44]	0.227	0.317	0.544	0.233	0.111	0.157	0.195	0.191
3D-HSC [13]	0.441	0.426	0.446	0.499	0.045	0.081	0.165	0.058
3D-RecGAN++ [49]	0.580	0.647	0.753	0.679	0.034	0.060	**0.066**	0.040
EDnet (ours)	0.577	0.654	0.750	0.663	0.033	0.062	0.069	0.042
3D-RVP (ours)	**0.598**	**0.668**	**0.760**	**0.696**	**0.032**	**0.060**	0.067	**0.039**

Figure 41: risultati ottenuti sui dati di test SV relativi alle metriche IoU e CE loss

Methods	IoU				CE			
	Bench	Chair	Couch	Table	Bench	Chair	Couch	Table
3D-EPN[9]	0.408	0.446	0.572	0.482	0.086	0.112	0.163	0.103
3D-RG[44]	0.185	0.278	0.475	0.187	0.108	0.171	0.210	0.186
3D-HSC [13]	0.439	0.426	0.455	0.482	0.047	0.090	0.163	0.060
3D-RecGAN++ [49]	0.531	0.594	**0.646**	0.618	0.041	0.074	**0.111**	0.053
EDnet (ours)	0.537	0.611	0.639	0.631	0.038	0.076	0.140	0.049
3D-RVP (ours)	**0.554**	**0.621**	0.643	**0.656**	**0.037**	**0.074**	0.138	**0.047**

Figure 42: risultati ottenuti sui dati di test CV relativi alle metriche IoU e CE loss

Dalle tabelle e dai risultati, in entrambi i casi, quindi utilizzando sia il set di test SV, che CV, si ottengono degli ottimi risultati per entrambe le metriche considerate.Nello specifico si migliora utilizzando SV, la *intersection over union* dello 2.3% rispetto gli altri metodi e si riduce dello 1.7% la perdita CE dimostrando l'efficacia del metodo.Mentre utilizzando i dati di test CV è possibile osservare una grande capacità di generalizzazione da parte di *3D-RVP*, infatti, 3D-RPV migliora, come visibile dalla figura 42, la IoU di circa il 3.6% rispetto *3D-RecGAN++* e di circa 2.3% rispetto la propria rete encoder-decorder.Anche il secondo esperimento, condotto allenando le rete neurali attraverso set di training multi-categoria, ha messo alla luce dei risultati più che ottimi per il metodo con migliorie, utilizzando entrambi i set di test SV e CV, sia sulla metrica IoU che sulla metrica di perdita CE.Inoltre, per questo secondo esperimento è invece più utile osservare i risultati qualitativi, mostrati nelle Figure 43 e 44, associati alle ricostruzioni 3D ottenute dal metodo sia per i dati di test SV che per quelli CV da cui si può notare, in entrambi i casi, che la *rete encoder-decoder* utilizzata da *3D-RVP* riesce ad imparare e predire

una forma 3D completa, ma i confini della forma predetta rimangono grossolani a differenza invece della forma 3D ottenuta e predetta alla fine dalla rete neurale di *3D-RVP* stesso che possiede molti dettagli anche lungo i confini.Questo dimostra, per l'ennesima volta, la grande generalizzazione di questo metodo di ricostruzione 3D.

Figure 43: risultati qualitativi della ricostruzione 3D sui dati di test SV.Dall'alto in basso, immagine di profondità, predizione grossolana ottenuta dalla rete encoder-decoder, predizione dettagliata ottenuta dalla rete neurale, forma originale e reale

Figure 44: risultati qualitativi della ricostruzione 3D sui dati di test CV.Dall'alto in basso, immagine di profondità, predizione grossolana ottenuta dalla rete encoder-decoder, predizione dettagliata ottenuta dalla rete neurale, forma originale e reale

Quindi *3D-RVP* riesce a ricostruire e restituire forme 3D, anche complesse, con alta qualità e con un alto grado di accuratezza.Inoltre, anche questo metodo, come PyTorch3D[1], migliora lo stato dell'arte relativo alla ricostruzione 3D e permette di ottenere predizioni di modelli 3D molto dettagliati e simili alla realtà. Questo chiude la spiegazione dettagliata di *3D-RPV* e quindi dei due metodi che possono

essere utilizzati in questa fase di ricostruzione, essi sono ampliamente trattati in letteratura e si rimanda il lettore a [1] e [23] per maggiori approfondimenti.E' intuibile che comunque, grazie ai risultati descritti, quindi al valore elevato di accuretezza posseduto dai modelli restituiti in output, ai dettagli presenti nel modello di output e alle metriche relative alla perdita di informazione, i due metodi proposti per la restituzione risultino essere tra i migliori per poter associare alle aree di cambiamento rilevate, per cui deve essere effettuato l'aggiornamento, un modello 3D molto simile alle condizioni reali dell'aree di interesse e da cui poter estrarre, attraverso operazioni come la classificazione automatica o la segmentazione automatica degli oggetti, informazioni significative relative agli oggetti presenti nell'area di cambiamento utili a ridurre l'intervento manuale richiesto durante l'aggiornamento dei dati, a semplificare la fase stessa di aggiornamento e ad abbassare i tempi necessari richiesti per completare quest'ultima.Si ribadisce, che comunque, questi due metodi presentati vengono utilizzati solamente come metodi per poter supportare la fase di aggiornamento dei dati per le aree di cambiamento ritrovate e che sulle nuvole di punti o più in generale sui modelli 3D ottenuti attravero l'applicazione dei metodi *PyTorch3D* o *3D-RVP* è possibile svolgere diverse computazioni, anche viste durante la spiegazione del metodo proposto e citate più volte durante la trattazione e spiegazione di questa fase di restituzione 3D ed aggiornamento dei dati, che permettano letteralmente di ridurre l'intervento umano richiesto durante l'aggiornamento dei dati e di semplificare l'aggiornamento dei dati evitando inutili confronti o osservazioni visive dell'area di interesse da aggiornare attraverso l'ottenimento di informazioni/-dati utili correlabili agli oggetti presenti nell'area di cambiamento rilevata o più in generale alla zona da aggiornare.E' il caso di un'operazione di classificazione automatizzata, applicata alla nuvola di punti o al modello 3D ottenuto da *PyTorch3D* o da *3D-RVP* rappresentante la generica aree di cambiamento rilevata, la quale permetterà di ottenere automaticamente informazioni riguardanti il tipo di oggetto oppure un'operazione di segmentazione automatizzata degli oggetti che invece permetterà di conoscere automaticamente la posizione e la delimitazione dell'oggetto nell'area di cambiamento rilevata dal nostro metodo.Ovviamente altre operazioni, oltre quelle di classificazione automatica e segmentazione, possono essere applicate ai modelli 3D restituiti dai metodi di ricostruzione 3D sempre con l'obiettivo, come nel caso della classificazione automatica e della segmentazione, di poter automaticamente ottenere informazioni/dati utili relativi all'area di cambiamento rilevata per cui effettuare l'aggiornamento dei dati o più nel dettaglio ottenere automaticamente dati utili relativi agli oggetti presenti nell' area di cambiamento rilevata dal metodo e considerata che permettano di automatizzare l'aggiornamento di una parte di dati, presenti nel database associato alla nuvola di punti meno recente, per la generica area di cambiamento rilevata considerata e quindi di ridurre l'intervento umano richiesto durante la fase di aggiornamento.

Con la conclusione di questo capitolo terminiamo la spiegazione dettagliata in tutte le sue fasi del metodo proposto.Nel capitolo successivo verranno invece mostrate la sperimentazione condotta, solamente sulle parti essenziali e cruciali del metodo spiegato, in un contesto di reale applicabilità e delle verifiche tenute sui risultati finali ottenuti da questa sperimentazione da cui è stato possibile dedurre il grado di bontà posseduto dal metodo proposto e realizzato.

4 Esperimenti condotti e risultati ottenuti relativi al metodo proposto

In questo capitolo viene descritta la sperimentazione eseguita solamente sulle parti essenziali del metodo proposto, tralasciando le fasi di verifica della classificazione, di ricostruzione 3D ed aggiornamento finale dei dati a causa della mancanza dei dati necessari per attuarle, al fine di testare l'applicabilità di quest'ultimo in contesti pratici e poter verificare la veridicità del rilevamento delle aree di cambiamento ritrovate, il numero di punti significativi al cambiamento non considerati nelle aree di cambiamento finali ritrovate ed il grado di risoluzione posseduto dal metodo stesso.Per questo, verranno mostrati e descritti, in modo sequenziale, i modelli in input utilizzati, in formato nuvola di punti, i parametri scelti e fissati durante le varie fasi della sperimentazione del metodo, i dati utilizzati ed ottenuti, ancora, in ogni fase del metodo sperimentato, ed infine, le verifiche condotte sui risultati finali ottenuti in output dalla sperimentazione eseguita, da cui si è valutata e definità la bontà del metodo realizzato.

4.1 Descrizione dei parametri e dei dati utilizzati ed ottenuti in ogni fase del metodo proposto durante la sperimentazione condotta

4.1.1 Nuvole di punti in input

Come abbiamo più volte ribadito nel capitolo precedente, il metodo proposto lavora su modelli 3D in input, in formato nuvola di punti, ottenuti da sistemi aerei o da MMS, quindi su dati 3D puramente terrestri e per questo si è scelto di studiare e speriamentare il metodo su una zona terrestre di grande interesse che rappresenta un pezzo strada o di rete viaria/stradale dove possono essere presenti ed avvenuti diversi cambiamenti significativi associati alla strada stessa, ai cartelli stradali presenti, ma anche alle costruzioni presenti nell'intorno abitativo della strada considerata.Le due nuvole di punti, a nostra disposizione, utilizzate per la speriamentazione sono raffigurate in Figura 44 e 45.Si può notare che esse avranno in comune solamente la parte evidenziata mediante un rettangolo e che il metodo di change detection proposto verrà applicato proprio in quest'area al fine di poter rilevare i cambiamenti avvenuti e verificare i risultati ottenibili e quindi il metodo stesso.

Figure 45: Vista dall'alto della nuvola di punti vecchia, definita come "prima"

Figure 46: Vista dall'alto della nuvola di punti più recente, definita come "dopo"

Ovviamente la Figura 45 è una nuvola di punti rappresentante la situazione temporale, associata alla zona di studio, vecchia, mentre la nuvola presente in Figura 46 rappresenta la situazione temporale più recente.Anche se non fosse molto visibile dalle due immagini, esse avranno una zona in comune, quella evidenziata da un rettangolo, dove nella situazione temporale definita come "prima" era presente un incrocio tra due strade, mentre nella situazione temporale più recente e quindi nella nuvola di punti definita come "dopo" è presenta ed osservabile, per la medesima zona, la presenza di una rotatoria, l'aggiunta di segnaletica verticale e modifiche anche sull'intorno abitativo di quest'area. E' opportuno, qualora parlassimo di nuvole in input utilizzate, conoscere anche il numero di punti che li forma o quanto meno l'ordine di grandezza di suddette.Entrambe le nuvole sono nuvole molto dense formate dall'ordine di decine di milioni di punti, con esattezza, la nuvola di punti "prima" è composta da 12 milioni di punti, mentre la nuvola di punti "dopo" da 14 milioni di punti, ma la zona in comune possedute da entrambe, su cui verrà applicato il metodo, sarà una zona, fortunatamente, formata da solamente 5 milioni di punti. Quindi, ribadiamo che i risultati che sono stati ottenuti e che saranno osservabili e descritti in uno dei paragrafi successivi sono risultati associabili al change detection solo per la zona di studio evidenziata dal rettangolo.

4.1.2 Classificazione ottenuta ed utilizzata durante la sperimentazione

Come ormai sappiamo, prima di poter effettuare qualsiasi computazione e poter procedere fino agli step finali del metodo al fine di ottenere le aree di cambiamento rilevate con le opportune delimitazioni, necessitiamo delle informazioni relative alla classificazione di ogni punto, presente nella zona di interesse, della nuvola di punti più recente.Nella sperimentazione condotta si è utilizzata la classificazione dei punti, relativa alla zona evidenziata nel paragrafo precedente mediante un rettangolo, ottenuta mediante il software *Agisoft Metashape* visibile dalla Figura 47 sottostante.

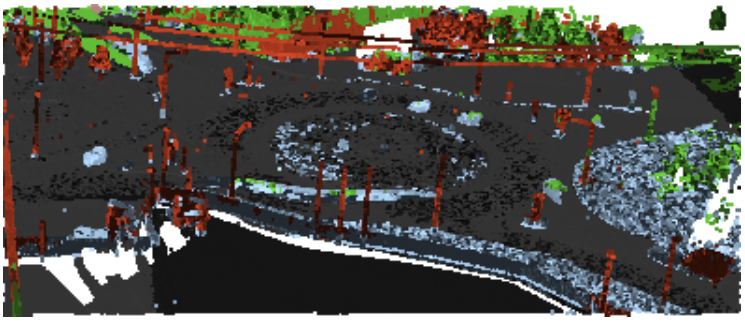


Figure 47: Classificazione utilizzata

La classificazione mostrata e che è stata utilizzata durante la speriamentazione è ottenuta considerando le 5 macro-classi di appartenenza, già citate nel capitolo precedente, per il generico punto, quali, *vegetazione*, *macchina*, *strada* e *costruzione* e *mare/fiume*.E' utile ribadire, come sappiamo, che ad ognuna di quest'ultime verrà associato un relativo valore reale intero(a cui è attribuito un colore) che rappresenta proprio la classe di appartenenza, *5*(verde) alla vegetazione, *2*(azzurro) ai punti classificati come macchina, *11*(grigio) ai punti classificati come strada ed il valore *6*(rosso) alle costruzione.Come è visibile dalla Figura 47 nel caso considerato non è presente un mare/fiume e quindi neanche punti classificati con la suddetta classe "mare/fiume" a cui veniva associato un valore intero pari ad 1 ed un colore blu scuro. Inoltre, si è scelto di utilizzare questa classificazione per la speriamentazione del metodo in quanto, come visibile, è molto accurata in termini di classificazione dei punti in relazione ai reali oggetti presenti nella zona di interesse/studio.Ovviamente tra tutti i punti considerati e classificati sono presenti, non in numero rilevante, anche punti classificati erroneamente.C'è da precisare che la classificazione errata di punti potrebbe incidere, anche se non in modo drastico considerando un numero non rilevante di punti, sulla fase di eliminazione dei punti stessi e quindi anche sui risultati finali ottenibili.E' possibile che alcuni punti definibili come "non falsi positivi" e rilevanti ai fine del cambiamento avvenuto siano stati classificati, in questa classificazione considerata, come dei "falsi positivi" e vengano eliminati, mentre altri punti definibili come reali "falsi positivi" siano stati classificati come "non falsi positivi" e saranno presenti nei risultati finali.Questi punti, classificati erroneamente, come già accennato, incideranno sui risultati ottenibili, ma non in modo significativo, in quanto, rispetto i punti classificati in modo giusto/ottimale rappresentano la minoranza.E' sempre utile, comunque, ribadire al lettore che, come già descritto nel capitolo precedente, il metodo ammette, nei risultati finali, la presenza, in numero irrilevante, di falsi positivi e riesce a ridurre il numero di punti classificati erroneamente, come "falsi positivi" o non, classificandoli alla giusta classe di appartenenza attraverso la fase di verifica della classificazione automatica ottenuta dal software *Metashape*, che purtroppo però, come già detto ad inizio capitolo, non è stata speriamentata/testata a causa della mancanza dei dati necessari per attuare questa fase, cioè del database preesistente contemporaneo alla nuvola di punti vecchia contenente l'*a-priori information* correlabile alla zona di studio/interesse, e di uno studio, condotto sempre sulla zona di interesse, da cui poter trarre dati ed informazioni a-priori rilevanti e relative agli oggetti presenti.Comunque, possedendo

un grado di accuratezza di classificazione alto, in relazione alla reale classificazione degli oggetti presenti nella zona di studio/interesse, si è scelto di utilizzare durante la sperimentazione del metodo proposto e per gli step successivi a questa fase la classificazione considerata ed ottenuta tramite il solo software *Agisoft Metashape*, su cui non è stata eseguita la fase di verifica.Quindi è intuibile, da quanto affermato poco prima, che la classificazione da utilizzata per gli step successivi, duranti la sperimentazione/applicazione del metodo, potrà presentare punti classificati erroneamente o punti definibili come "falsi positivi", ma in un numero irrilevante,in quanto essa deve essere il più consona possibile alla reale classificazione degli oggetti presenti nella zona di studio al fine di evitare il rilevamento di aree di cambiamento non significative, dovuto alla presenza, nei risultati finali ottenibili, di alcuni punti "falsi positivi" classificati attraverso delle classi significative al cambiamento stesso e quindi non eliminati, o il non rilevamento di aree di cambiamento significative dovuto, invece, alla non presenza, nei risultati finali ottenibili, di punti significativi al cambiamento avvenuto, in quanto classificati come "falsi positivi" e quindi non considerati come punti significativi al cambiamento avvenuto ed eliminati.Infine, da diversi test e sperimentazioni condotte proprio su questa fase relativa alla classificazione della nuvola di punti più recente, si è notato, come giusto che sia, dato che il metodo proposto si differenzia dagli altri metodi di change detection presenti in letteratura proprio per la conoscenza e l'utilizzo dell'informazione a-priori correlata alla zona di studio, quanto sia importante ed essenziale per il nostro metodo, ai fini dell'attendibilità e dell'accuratezza dei risultati finali relativi al cambiamento avvenuto, ottenere ed utilizzare,durante l'applicazione, solamente classificazioni accurate rispetto la realtà e, quindi, su cui è stata effettuata, anche se purtroppo non è stata possibile sperimentarla e testarla, la fase di verifica della classificazione attraverso l'utilizzo dell'informazione a-priori conosciuta e correlata alla zona di studio.Da tutto ciò si evince, una delle caratteristiche intrinsiche, potremmo definirlo anche un limite, del metodo proposto: l'elevata dipendenza dalla classificazione effettuata/ottenuta in questa fase ed utilizzata per gli step successivi del metodo stesso dei risultati finali ottenibili relativi al cambiamento avvenuto per la zona di studio/interesse.

4.1.3 Computazione del cambiamento ed ottenimento del file di testo rappresentante la zona di studio/interesse

Per computare i cambiamenti avvenuti tra le due nuvole di punti in input nella zona di studio/interesse considerata si è utilizzato, come gia descritto, il software *CloudCompare* con relativo toolCloud2Cloud Distance che permette di calcolare la distanza di scostamento lungo la coordinata z avvenuto tra i punti più vicini appartenenti alle due nuvole in input.I risultati ottenuti dal tool, in questa fase, sono visibili dall'immagine 48 sottostante, dove sempre per la data zona di interesse, ad ogni punto è stato associato un valore di distanza, rappresentante il cambiamento, visibile attraverso un colore.

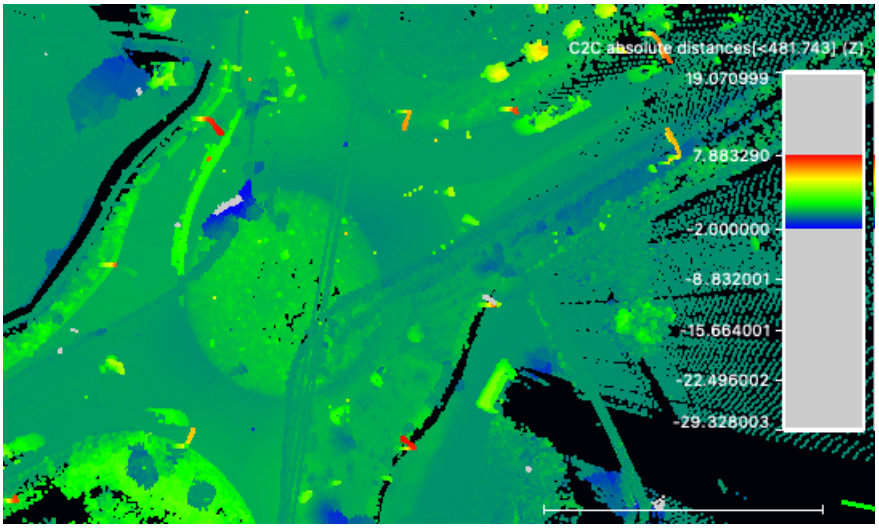


Figure 48: Distanze computate da CloudCompare

Le informazioni di distanza di scostamento relativa alla coordinata z computate per ogni punto della nuvola più recente dal tool in questo step e visibili dalla Figura 48, insieme alle informazioni di classificazioni associate ai punti nello step precedente e alle coordinate geografiche spaziali di ogni punto conosciute attraverso la nuvola di punti stessa, vengono estratte , tramite CloudCompare, in un file di testo rappresentante proprio la zona di studio/interesse.Il file di testo ottenuto in output in questa fase ha dimensione di circa 350 Mb e contiene 5.429.906 righe, ossia più di cinque milioni di punti.Ogni riga presente nel file ottenuto possiede la seguente struttura:

*x, y, z, *, *, *, classificazione, distanza*

Dove la quarta, quinta e sesta colonna, identificate da *, sono valori inutili associati alla distanza di scostamento di z calcolata, che vengono outputtati in automatico dal software CloudCompare durante l'estrazione degli attributi per la zona di studio/interesse e che non verranno presi in considerazione.

4.1.4 Definizione dei parametri utili all'eliminazione dei punti non significativi ai fini del rilevamento del cambiamento

Come sappiamo, successivamente ad aver ottenuto il file di testo rappresentante la zona di studio è utile estrarre da questa zona tutti quei punti poco significativi per il processo di change detection stesso e tutti quei i punti defibili come "falsi positivi".Al fine di operare queste eliminazioni si è scelto di considerare come "falsi positivi" i punti classificati attraverso la classe "vegetazione",quindi che possiedono valore di classificazione pari a *5* ed i punti classificati come "macchina" ossia tutti quei punti il cui attributo classificazione è pari a *2* ed inoltre si è scelto di considerare come non rilevanti, ai fini di un cambiamento avvenuto, i punti il cui valore di distanza di

scostamento associata è compreso tra $-0.20e0.25$ metri.La scelta di fissare proprio questo intervallo di valori di soglia è data da vari e precedenti esperimenti dove si ci è accorti che per i punti a cui veniva attribuito un valore di scostamento appartenente a quel definito intervallo non risultava essere avvenuto nessun cambiamento rilevante nel tempo. Comunque, Il file ottenuto in output dalla fase di eliminazione dei punti durante la sperimentazione, prende il nome di "filtrati", ha dimensione pari a circa 80 Mb e contiene 1.783.695 righe, ossia 1.783.695 punti significativi ai fini del rilevamento del cambiamento.Quindi, per la data zona iniziale di studio considerata,formata da più di 5 milioni di punti iniziali, sono stati scartati ed eliminati 3.646.211 punti, circa il 67% di quelli totali, non significativi ai fini del cambiamento avvenuto.

4.1.5 Definizione dei parametri utili alla clusterizzazione e all'ottenimento dei poligoni frontiera

Dal file di testo "filtrati" ottenuto precedentemente e contenente 1.783.695 punti sono stati considerati e clusterizzati solamente i primi 500.000 mila punti.La scelta di clusterizzare solamente 500.000 mila punti non inciderà nè sui risultati finali ottenuti, nè sulle metriche che vedremo, ma è più legata al tempo computazionale di esecuzione richiesto per effettuare e completare l'operazione di clusterizzazione di tutti i punti considerati.Purtroppo, il tempo di esecuzione richiesto dall'algoritmo sviluppato per operare la clusterizzazione dei punti è uno dei limiti presenti nel metodo proposto in quanto l'algoritmo sviluppato richiede un tempo computazionale pari ad $O(n^2)$ dipendente dal numero di punti n in input da clusterizzare, i quali, essendo solitamente nell'ordine di milioni porta il tempo richiesto per completare l'operazione ad essere abbastanza alto.Inoltre esso dipende anche dall'hardware adoperato per performare l'operazione. Comunque, la clusterizzazione, in fase di sperimentazione, è stata effettuata considerando il parametro di distanza, che definisce se due punti sono abbastanza vicini per poter appartenenre al medesimo cluster, pari a 0.20 metri e si sono ottenuti in output, dati 500.000 mila punti, 3854 cluster di diverse dimensioni, rappresentanti le aree di cambiamento per la zona di studio, in formato file di testo dove la generica riga che definisce il punto appartenente al cluster possiede la seguente struttura:

$$x,y,z$$

ossia le coordinate relative al punto considerato.Inoltre, al generico file outputtato in questa fase di clusterizzazione è associato come nome un numero intero reale crescente, che parte dal valore 0 e che viene incrementato man mano che i cluster vengono creati.L'output ottenuto in fase di speriamentazione è il seguente:

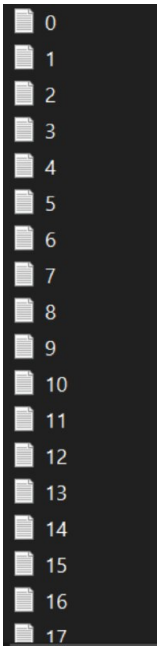


Figure 49: File di testo dei cluster ottenuti nella fase di clusterizzazione durante la sperimentazione

Una sfilza di file di testo rappresentanti i cluster ritrovati, a cui vengono associati come nome un valore reale intero crescente. Successivamente ad aver ottenuto il generico cluster, come sappiamo, la fase successiva prevede di trovare, solamente per i cluster rilevanti, il poligono frontiera delimitante l'area di cambiamento associata al cluster ritrovato.La rilevanza del cluster è stata definita e controllata, durante questa sperimantazione, fissando il valore di threshold di rilevanza a 450 punti, che è proprio, per la zona studiata, il numero di punti che formano il cartello stradale più piccolo per cui poter rilevare un cambiamento significativo.Impostando questo valore di threshold, si sono ottenuti, in questa fase del metodo, solamente 26 cluster, sui 3854 cluster disponibili, definibili come "rilevanti" per cui sono stati calcolati ed outputtati in formato file di testo i poligoni frontiera che delimitano le aree di cambiamento significative, associate al generico cluster considerato, ritrovate dal metodo proposto e per cui aggiornare i dati presenti nella base di dati preesistente.Al generico file di testo ritornato in output in quest'ultima fase, come visibile dalla Figura 50 sottostante, è stato associato come nome, il nome "poligonofrontieracluster" più il nome associato al cluster per cui si è computo il poligono, ottenendo così delle relazioni tra i file cluster ed i file contienente i poligoni frontiera ottenuti.Quindi le coppie (cluster, clusterfrontiera) rappresentano i risultati finali ottenuti ed outputtati dal metodo proposto e quindi anche le aree di cambiamenti rilevate dal metodo per la zona di studio/interesse.

Questo conclude la panoramica dei parametri e dei dati utilizzati ed ottenuti nelle varie fasi del metodo proposto durante la speriamentazione effettuata, c'è ancora da dire che i risultati finali ottenuti da questa speriamentazione, quindi le coppie di file di testo (cluster, clusterfrontiera), sono tutte state analizzate e controllate, tramite verifiche manuali e visive, al fine di poter valutare il metodo realizzato.Tali verifiche

vengono ampliamente descritte nel capitolo successivo.

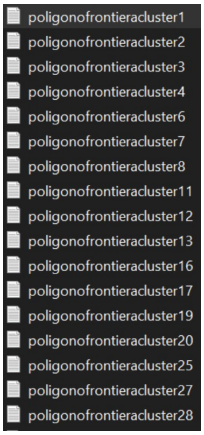


Figure 50: File di testo ottenuti durante la sperimentazione dell'ultima fase del metodo proposto contenenti i poligoni frontiera opportunamente computati per i cluster definibili come rilevanti

4.2 Verifiche effettuate sui risultati ottenuti e valutazione del metodo proposto

Da alcune verifiche manuali e visive svolte sui risultati ottenuti in output dall'ultima fase considerata, e quindi sui 26 file di testo ottenuti e rappresentanti i poligoni frontiera associati alle generiche aree di cambiamento rilevate, si sono ottenuti dei risultati valutativi per il metodo proposto. Le verifiche svolte permetteranno, infatti, di definire la "bontà" del metodo sviluppato andado ad analizzare e studiare:

1. *quanto siano veritieri i risultati ottenuti in relazione ai reali cambiamenti avvenuti avvenuti per la zona di studio*

2. *il numero di outlies presenti nei risultati finali ottenuti*

3. *il grado di risoluzione posseduto dal metodo stesso*

Per quanto riguarda la prima valutazione c'è da dire che andremo ad osservare quanto siano veritieri i risultati relativi alle aree di cambiamento ritrovate ed ottenute dal metodo proposto in relazione ai reali cambiamenti avvenuti nella zona di studio.Per verificare questa performance del metodo si sono analizzati qualitativamente i risultati ottenuti relativi alle 26 aree di cambiamento ritrovate, e si è osservato se per queste aree di cambiamento ritrovate sia avvenuto realmente un cambiamento significativo tra le due situazioni temporali conosciute tramite le due nuvole di punti in input.Le analisi mostrano che il nostro metodo presenta un alto grado di veridicità dei risultati ottenuti, in quanto, il 92.3% dei poligoni frontiera computati e delimitanti i generici cluster rilevanti, ossia le generiche aree di cambiamento rilevate per cui aggiornare i dati, rappresentano delle aree cambiamento significative dove è realmente avvenuto un cambiamento tra la situazione temporale vecchia e

la situazione temporale più recente.Tale risultato è osservabile attraverso le Figure sottostanti che mostrano i bounding box di alcuni dei poligoni frontiera ritrovati, il cambiamento avvenuto per questa delimitazione e quindi l'effettivo ritrovamento di aree di cambiamento significative per cui aggiornare i dati presenti.

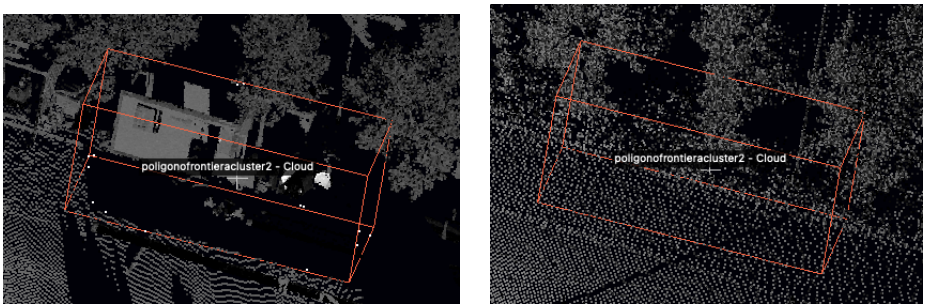


Figure 51: Bounding box del poligono frontiera associato al cluster rilevante numero 2 per cui è visibile il reale cambiamento avvenuto tra la situazione temporale vecchia(sinistra) e la situazione temporale più recente(destra)

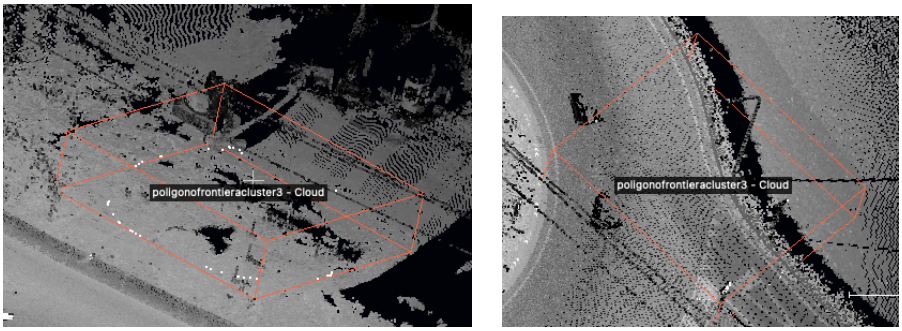


Figure 52: Bounding box del poligono frontiera associato al cluster rilevante numero 3 per cui è visibile il reale cambiamento avvenuto tra la situazione temporale vecchia(sinistra) e la situazione temporale più recente(destra)

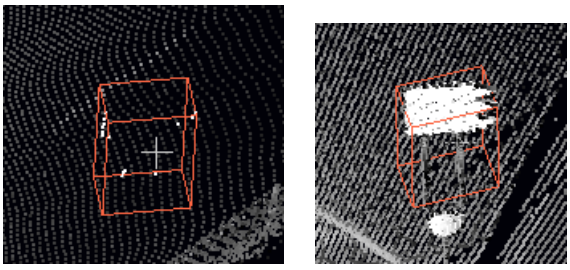


Figure 53: Bounding box del poligono frontiera associato al cluster rilevante numero 28 per cui è visibile il reale cambiamento avvenuto tra la situazione temporale vecchia(sinistra) e la situazione temporale più recente(destra)

Queste Figure rappresentano solamente 3 aree delle 26 aree di cambiamento rilevate, delimitate ed ottenute dal metodo.Come già accennato, il 92,3% di queste

26 aree rappresenta un cambiamento veritiero ed avvenuto nel tempo, purtroppo però sono presenti, nei risultati finali, 2 aree di cambiamento rilevate dove sono avvenuti dei cambiamenti poco significativi in quanto aree vegetative.Queste aree di cambiamento rilevate, poco significative, sono presenti nei risultati finali ottenuti dal metodo durante la sperimentazione a causa dell'errata classificazione dei punti che li formano dovuta alla mancata attuazione della fase di verifica della classificazione.Infatti, i maggiori errori che si potrebbero presentare qualora parlassimo di veridicità dei risultati ottenuti in relazione ai cambiamenti reali avvenuti per la zona di studio sono dovuti ad errori di classificazione dei punti e/o oggetti presenti nella nuvola più recente e come accaduto per la sperimentazione effettuata, alcune delle aree di cambiamento ottenute in output dal metodo potrebbero non essere significative o addirittura potrebbe accadere, sempre a causa dell'errata classificazione, che alcune delle aree di cambiamento più significative per la zona di studio non siano presenti nei risultati finali ottenibili.Comunque, è utile ribadire al lettore che grazie alle verifiche condotte sui risultati ottenuti è possibile definire che il grado di veridicità del metodo sviluppato, anche se dipendente dall'operazione di classificazione, si presenta molto alto e quindi che il metodo proposto riesce ad ottenere in output la maggior parte dei cambiamenti significati avvenuti per la zona di studio. Invece, l'analisi del numero di punti definibili come *outlies* permette di valutare, in questo contesto, quanti punti significanti ai fini del cambiamento avvenuto vengono scartati e non sono presenti nei risultati finali ottenibili rappresentanti le aree di cambiamento ritrovate dal metodo.Ricordiamo al lettore, che gli outlies sono punti significativi ai fini del cambiamento avvenuto, in quanto considerati nella fase di clusterizzazione, ma che essendo isolati rispetto i restanti punti da clusterizzare e non in numero sufficiente verranno raggruppati in cluster non rilevanti e successivamente eliminati e non considerati nei risultati finali outputtati.Valutare e definire questo valore è molto utile, in quanto ci permette di stimare, in percentuale, un valore di perdita di punti significativi correllabile al metodo stesso, infatti, nella sperimentazione condotta, si è osservato che i 26 cluster outputtati dal metodo, rappresentati le aree di cambiamento ritrovate e per cui sono stati calcolati gli opportuni poligoni frontiera, sono formate da un un totale di 484.682 punti, mentre il totale dei punti rilevanti ai fini del cambiamenteo e da clusterizzare, come definito nel paragrafo 2, era di 500.000 punti.Da questi valori è deducibile che 15.318 punti significativi sono stati scartati e non sono presenti nelle aree di cambiamento rilevate e ritornate in output dal metodo in quanto *outlies*.Portando in termini percentuali il numero di punti *outlies* ottenuti rispetto il numero totale di punti iniziali significativi da clusterizzare si ottiene una stima relativa alla perdita di punti significativi correlabile al nostro metodo.Nella speriamentazione condotta questo valore è di circa 3%, cioè 3%(di 500.000) dei punti totali significativi per il cambiamento avvenuto non è stato considerato e non è presente nei risultati finali ottenuti.Comunque dalla sperimentazione descritta e da altri test condotti sul metodo proposto è stato osservabile che in media solo tra l'1% e il 3% dei punti totali significativi per il cambiamento risulta essere un outliers.Questa media in percentuale ottenuta ci permette di affermare che il metodo proposto riesce a clusterizzare e ad ottenere risultati relativi al cambiamento avvenuto per circa il 97%-99% dei punti totali considerati significativi e da clusterizzare.Quindi, da queste percentuali, ottenute attraverso analisi

sul numero di punti *outliers* presenti durante gli esperimenti condotti, è possibile poter attribuire al metodo proposto e realizzato un basso valore di perdita relativo ai punti significativi al cambiamento, in quanto, come facilmente deducibile, durante l'applicazione del metodo solamente un numero irrelevante di punti, rispetto il totale di punti significativi al cambiamento, viene scartato e non considerato nelle aree di cambiamento ritrovate e ritornate in output dal metodo stesso. Infine, è utile valutare la *risoluzione* del metodo realizzato.Qualora parlassimo di risoluzione del metodo faremo riferimento alla possibilità, per il metodo stesso, di poter rilevare ed ottenere, in output, le aree di cambiamento, e quindi i cambiamenti avvenuti, relativi agli oggetti di più piccola dimensione, presente nella zona di studio/interesse, per cui è possibile rilevare un cambiamento.Diciamo subito che il metodo presenta un buon/ottimo grado di risoluzione e che è stato possibile verificare quest'ultimo analizzando in primis i rilsultati ottenuti dalla sperimentazione condotta e descritta precedentemente dove è stato possibile osservare l'ottenimento in output di un'aree di cambiamento, con opportuno poligono frontiera, relativo ad un cartello stradale che rappresenta proprio il più piccolo oggetto presente nella zona di studio per l'esperimento condotto.L'area di cambiamento ritrovata è visibile dalla Figura 53, la quale mostra la situazione vecchia dove non era presente il cartello, la situazione più recente dove invece è presente, per la medesima area, un nuovo cartello stradale ed il poligono frontiera ottenuto che delimita l'area di cambiamento rilevata e ritrovata dal metodo stesso.Oltre all'analisi dei risultati ottenuti in output dalla sperimentazione descritta nel paragrafo 2 di questo capitolo, per poter verificare e testare ancora la risoluzione del metodo è stato approciato un mini-esperimento, con gli stessi parametri utilizzati per la sperimentazione precedente, ma sui dati visibili nella Figura 54 sottostante, i quali mostrano una situazione temporale "prima" rappresentante una strada ed una situazione temporale più recente dove è visibile, per la medesima zona di studio,la presenza di un nuovo cartello stradale.

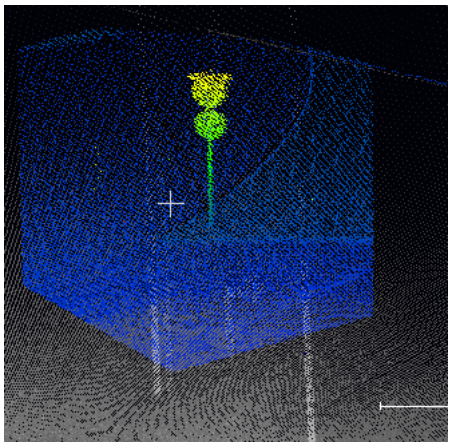


Figure 54: Zone di studio considerate per il mini-esperimento condotto al fine di testare la risoluzione del metodo.Nell'immagine troviamo, in scala di grigi la situazione temporale precedente relativa alla sola strada, a colori la nuova situazione temporale che mostra la presenza sulla medesima strada di un nuovo cartello stradale

Applicando il metodo proposto su questi dati, al fine di poter dedurre il buon/ot-

timo grado di risoluzione posseduto, il metodo deve obbligatoriamente ritrovare l'area di cambiamento associabile al cartello stesso.I risultati ottenuti in output dall'applicazione del metodo in questo contesto considerato, mostrano l'opportuno rilevamento dell'area di cambiamento associabile al nuovo cartello stradale presente per la zona studiata.Dalla Figura 55 è possibile osservare il bounding box del poligono frontiera correlato all'area di cambiamento rilevata ed ottenuta dall'applicazione del metodo e contenetente proprio tutti i punti formanti il cartello stradale.Quindi, dai risultati ottenuti ed osservati sia per questo mini-esperimento che per la sperimentazione principale condotta è possibile definire, per il metodo realizzato, il buon/ottimo grado di risoluzione posseduto.

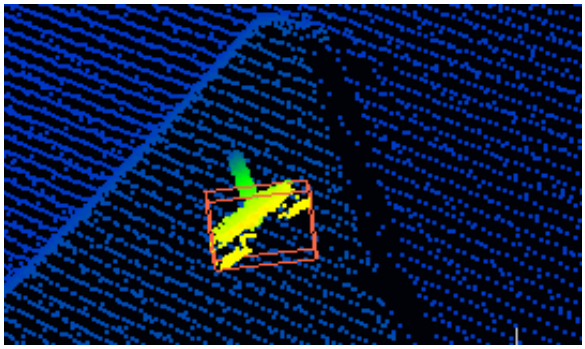


Figure 55: Bounding box del poligono frontiera delimitante l'area di cambiamento rilevata dal metodo relativa al nuovo cartello stradale, precedentemente non presente sulla strada considerata

Questo chiude la visione generale degli esperimenti condotti e delle verifiche effettuate sul metodo al fine di valutare la bontà di quest'ultimo.Possiamo affermare che, nonostante la presenza di limitazioni, le verifiche effettuare, hanno evidenziato la bontà del metodo stesso, in contesti pratici di applicazione, nel poter rilevare cambiamenti territoriali avvenuti nel tempo.Quindi dalle verifiche condotte e dalla bontà riscontrata ed associata al metodo stesso è possibie affermare che quest'ultimo si presenta come un metodo *efficace* per poter processare, utilizzando dati puramente terrestri, il task di *change detection* di zone territoriali.

5 Conclusioni e sviluppi futuri

L'obiettivo principale di questo lavoro, cioè definire un metodo che consenta, a partire da dati ottenuti da remote sensing, di rilevare le aree del territorio dove sono avvenuti cambiamenti è stato raggiunto.Il metodo è stato definito nelle sue linee generali e le fasi più importanti sono state verificate.La verifica, seppur parziale come abbiamo detto, ci consente di ritenere la strada intrapresa molto promettente.Abbiamo anche esplorato le possibilità di ulteriormente ridurre l'intervento manuale, durante la fase di aggiornamento dei dati per cui è avvenuto un cambiamento, utilizzando metodi di restituzione automatica degli oggetti contenuti nelle aree di cambiamento rilevate.Tali metodi di restituzione automatica (PyTorch3D, 3D-RVP) vengono applicate alle aree di cambiamento rilevate per poter ottenere un modello 3D accurato e dettagliato di quest'ultime, permettendo, anche attraverso operazioni come classificazione, segmentazione o altre, di ottenere informazioni utili ed utilizzabili durante l'aggiornamento dei dati.Tali dati utili ed estrapolabili dai modelli 3D accurati e dettagliati ottenuti dall'applicazione di PyTorch3D o 3D-RVP riducono l'intervento manuale richiesto, semplificando l'operazione di aggiornamento stessa e riducendone i tempi richiesti.

Inoltre, gli aspetti critici del processo sono fondamentalmente legati alla bontà della classificazione della nuvola di punti.Migliori risultati relativi alla classificazione saranno ottenuti applicando la fase di verifica della classificazione automatica ottenuta da *MetaShape*. Un'altro aspetto sicuramente migliorativo sarà dato da un miglior tempo di esecuzione richiesto per elaborare grandi quantità di punti (nuvole molto dense per grandi superfici).Il tempo di esecuzione richiesto per la clusterizzazione di tutti i punti aumenta all'aumentare dei punti da processare, per cui si consiglia di utilizzare durante le future verifiche hardware performanti e di ultima generazione.

Per quanto riguarda, invece, gli sviluppi futuri del lavoro, si tratta innanzitutto di verificare le fasi del processo non verificate o verificate parzialmente.Per svolgere queste verifiche si richiede la disponibilità di dati ben organizzati, in particolare per la fase di verifica della classificazione della nuvola di punti, che come si è detto più volte rappresenta un aspetto cruciale del metodo realizzato e richiede la disponibilità della base di dati contemporanea alla nuvola di punti vecchia per poter inserire la conoscenza a-priori e migliorare la classificaziona automatica ottenuta attraverso il software *MetaShape*. Siamo certi che questo migliorerà oltre l'accuratezza dei risultati finali ottenuti anche l'accuratezza della classificazione utilizzata.

Altro sviluppo futuro consiste nel verificare la fase di restituzione 3D e di aggiornamento dei dati per le aree di cambiamento rilevate.Questa verifica permetterà di osservare "come" e "quanto" l'associazione di un modello 3D accurato e dettagliato, ottenuto tramite l'applicazione di metodi di ricostruzione 3D alle aree di cambiamento rilevate, da cui poter estrarre informazioni relativi agli oggetti presenti, potrà effettivamente ridurre l'intervento manuale richiesto durante la vera e propria fase di aggiornamento dei dati.

References

[1]] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In NeurIPS, 2019.

[2] U. Grenander. Lectures in Pattern Theory I, II and III: Pattern Analysis, Pattern Synthesis and Regular Structures. 1976-1981.

[3] S. R. Marschner and D. P. Greenberg. Inverse rendering for computer graphics. Citeseer, 1998

[4] S. Liu, W. Chen, T. Li, and H. Li. Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction. In ICCV, 2019.

[5] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In ICLR, 2017.

[6] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with gpus. arXiv preprint arXiv:1702.08734, 2017.

[7] H. Kato, Y. Ushiku, and T. Harada. Neural 3D mesh renderer. In CVPR, 2018.

[8] S. Liu, W. Chen, T. Li, and H. Li. Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction. In ICCV, 2019.

[9] M. M. Loper and M. J. Black. OpenDR: An approximate differentiable renderer. In ECCV,2014

[10] W. Chen, H. Ling, J. Gao, E. Smith, J. Lehtinen, A. Jacobson, and S. Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. In NeurIPS, 2019.

[11] H. Kato, Y. Ushiku, and T. Harada. Neural 3D mesh renderer. In CVPR, 2018.

[12] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012, 2015.

[13] G. Gkioxari, J. Malik, and J. Johnson. Mesh R-CNN. In ICCV, 2019.

[14] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang. Pixel2Mesh: Generating 3D mesh models from single RGB images. In ECCV, 2018.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In CVPR, 2016.

[16] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3D-R2N2: A unified approach for single and multi-view 3d object reconstruction. In ECCV, 2016.

[17] H. Fan, H. Su, and L. J. Guibas. A point set generation network for 3d object reconstruction from a single image. In CVPR, 2017.

[18] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[19] 3D-EPN A. Dai, C.R. Qi, M. Niessner, Shape completion using 3D-encoder-predictor CNNs and shape synthesis, in: 30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6545–6554

[20] J. Varley, C. DeChant, A. Richardson, J. Ruales, P. Allen, Shape completion enabled robotic grasping, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 2442–2447.

[21] X. Han, Z. Li, H. Huang, E. Kalogerakis, Y. Yu, High-resolution shape completion using deep neural networks for global structure and local geometry inference, in: 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 85–93, https://doi.org/10.1109/ICCV.2017.19.

[22] B. Yang, S. Rosa, A. Markham, N. Trigoni, H. Wen, Dense 3D object reconstruction from a single depth view, IEEE Transactions on Pattern Analysis and Machine Intelligence 41 (2019) 2820–2834, https://doi.org/ 10.1109/TPAMI.2018.2868195.

[23] Zhao, Meihua, et al. "3D-RVP: A method for 3D object reconstruction from a single depth view using voxel and point." Neurocomputing 430 (2021): 94-103. 1.https://ihal.it/come-funziona-la-classificazione-delle-immagini/