UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

CONFRONTO E VALIDAZIONE DI TOOL DI ANNOTAZIONE PER TASK DI HUMAN PARSING IN DATI RGB-D

Relatore: Prof. Stefano Ghidoni

Laureando: Davide Rizzotti

Correlatori: Dott. Matteo Terreran
Dott. Leonardo Barcellona

ANNO ACCADEMICO 2021 - 2022

Data di Laurea 19 Luglio 2022

Sommario

Negli ultimi anni la ricerca ha mostrato un notevole interesse nel task di human parsing, che consiste nel riconoscere e segmentare le varie parti del corpo umano all'interno di in un'immagine. E' di cruciale importanza in tutte le applicazioni della robotica in cui è necessaria un'interazione - attiva o passiva - con esseri umani. I modelli per risolvere questo problema richiedono un'ingente quantità di dati. Nell'ambito 2D sono stati ottenuti ottimi risultati grazie alla presenza di dataset annotati di grandi dimensioni, mentre nell'ambito 3D i dataset disponibili sono pochi e di dimensione limitata. Per questi motivi, mentre la ricerca ha fatto notevoli passi avanti sviluppando diverse architetture per il task di human parsing in dati 2D, non si è riscontrato lo stesso successo in ambito 3D: lo sviluppo di applicazioni di collaborazione uomo-robot è fortemente penalizzato. In questo lavoro di tesi viene pertanto proposta un'analisi e comparazione dei principali tool di annotazione 3D disponibili, in base alla loro usabilità, qualità dell'annotazione ottenuta, tempo di annotazione richiesto, funzionalità disponibili e l'eventuale presenza di bug. L'obiettivo di questa analisi è fare luce sugli strumenti attualmente disponibili evidenziando i loro pregi e difetti in relazione al task di human parsing. Sono stati comparati 3 tool di due tipologie diverse (*pixel-wise* e a *3D bounding boxes*), testati su un insieme di pointclouds acquisite in laboratorio[1] per mezzo di una rete di telecamere RGB-D. Con ogni tool è stato creato un piccolo dataset di 5 pointclouds annotate. I risultati ottenuti mostrano che tutti i tool consentono di effettuare annotazioni precise delle pointcloud: in particolare l'annotazione pixel-wise fornisce segmentazioni più accurate in un minor tempo rispetto all'annotazione tramite bounding boxes.

[1]IAS-LAB dell'Università di Padova

Sommario

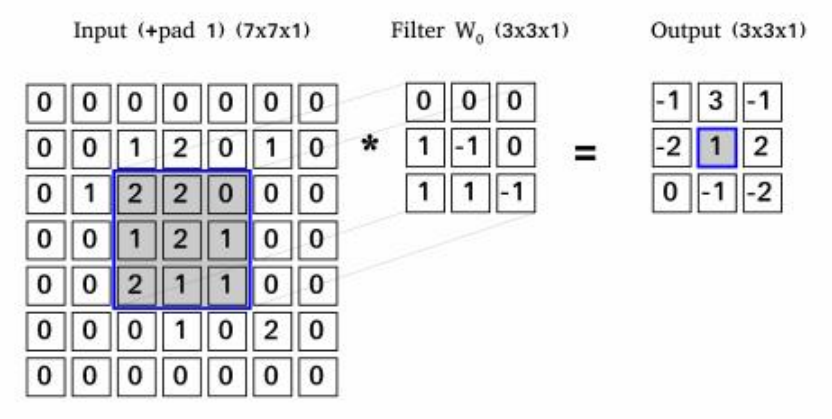
Terminologia

In questa sezione vengono definiti i vari termini tecnici presenti nel testo, al fine di evitare ambiguità durante la lettura. Alcune definizioni verranno riprese anche durante l'introduzione.

- **Pointcloud**: Insieme di punti nello spazio 3Ddefiniti in un certo sistema di riferimento, descritti da tre o più coordinate (X,Y,Z ed eventualmente R,G,B per l'informazione sul colore, o L per l'etichetta fornita dalla segmentazione).
- **Telecamera RGB-D**: Particolare tipo di telecamera che associa ad ogni pixel RGB dell'immagine la sua profondità, rispetto alla telecamera stessa.
- **Segmentazione**: Assegnazione di una particolare classe ad ogni pixel di un'immagine. La classe assegnata verrà chiamata anche "label" o "etichetta".
- **Human Parsing**: Processo di segmentazione di immagini contenenti una o più persone, nelle varie parti del corpo (testa, busto, mano destra etc.).
- **Dataset**: Insieme di dati, nel nostro caso pointclouds RGB-D manualmente annotate.
- **Tool di Annotazione**: Software utilizzato per eseguire la segmentazione manuale di immagini.
- **3D Bounding Box**: E' un parallelepipedo contenente un insieme di punti. Sono utilizzate nell'ambito della segmentazione per indicare l'appartenenza di un insieme di punti ad una certa classe.
- **Pixel-wise**: Letteralmente "*pixel per pixel*". Termine utilizzato per riferirsi ad un tool che consente l'annotazione pixel per pixel.
- **Rete neurale**: Modello matematico che si ispira al funzionamento del cervello umano, costituito da un insieme di neuroni disposti su più livelli (*layer*). I neuroni assumono un valore numerico che ne determina l'attivazione. Le connessioni biologiche fra i neuroni sono modellate tramite rami pesati: un peso positivo corrisponde ad una connessione eccitatoria, uno negativo ad una inibitoria.
- **Training Set**: Insieme di dati utilizzati per addestrare la rete neurale a risolvere un particolare problema.
- **Loss Function**: Funzione che fornisce una stima di quanto la rete sia competente a risolvere il problema in questione. Minore è il suo valore, maggiore è la performance della rete neurale.
- **Addestramento**: Aggiornamento progressivo dei parametri interni della rete (pesi) per minimizzare la loss function, e quindi ottenere prestazioni sempre migliori.

- **Convoluzione**: Operazione che permette di estrarre *features*(spigoli, bordi, forme partcolari) da un'immagine. E' una moltiplicazione matriciale fra una matrice di parametri variabili (filtro) e la matrice dell'immagine. L'output della convoluzione è detto *feature map*.

Figura 1: Convoluzione



- **Pooling**: Operazione applicata successivamente alla convoluzione. Permette di ridurre la dimensione delle *feature maps*, in modo tale da alleggerire la complessità computazionale. Il *Max Pooling*, ad esempio, restituisce in output il valore massimo in una porzione dell'immagine.

Figura 2: Pooling

1 Introduzione

1.1 Lo Human Parsing e le sue applicazioni

Per lo sviluppo di applicazioni di robotica che devono relazionarsi e interagire con l'ambiente circostante la percezione gioca un ruolo fondamentale nel permettere al robot di comprendere cosa c'è intorno a lui. Grazie alla segmentazione semantica, cioè quel compito della *computer vision* che si occupa di assegnare ad ogni porzione di un'immagine una determinata classe (es. albero, macchina, strada etc.), il robot può comprendere cosa lo circonda. Lo Human Parsing è un caso particolare di segmentazione semantica che mira a identificare le varie parti del corpo delle persone presenti in un'immagine (Figura 3). Si tratta di un task molto importante perchè ha una vasta gamma di potenziali

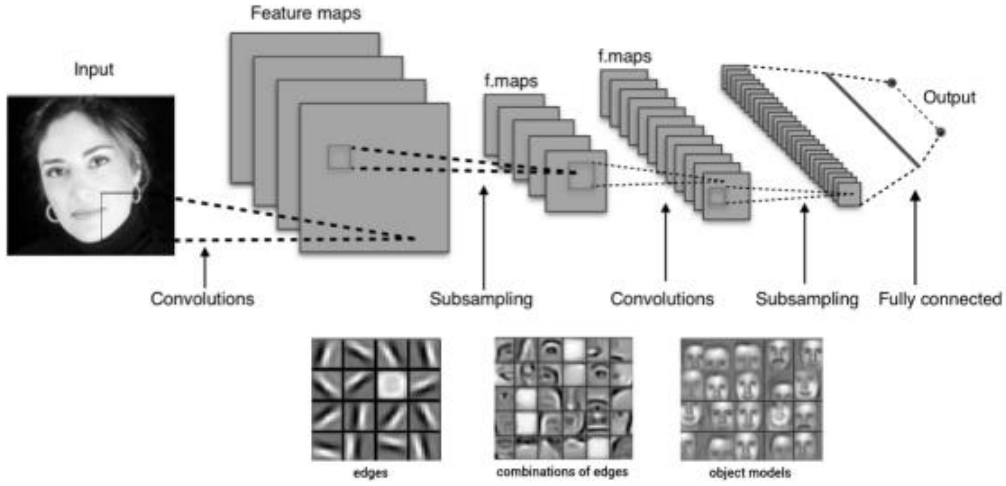
(a) Su immagine 2D

(b) Su pointcloud 3D

Figura 3: Esempi di output dello human parsing

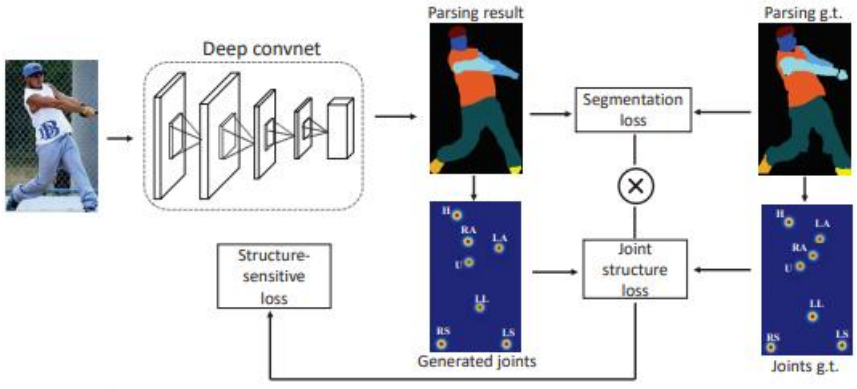
applicazioni: dall'identificazione delle persone e dei loro comportamenti, alla collaborazione uomo-robot, fino alla guida autonoma. Per questo motivo nell'ultimo decennio molti gruppi di ricerca si sono messi al lavoro per cercare delle soluzioni, sfruttando il crescente successo del deep learning. La quasi totalità degli approcci ([1], [2], [3], [4], [5], [6]) per risolvere il task di human parsing sono infatti approcci di questo tipo. Il deep learning è una branca del machine learning che utilizza reti neurali artificiali, un modello matematico che cerca di simulare il comportamento del cervello umano e che impara da una grande quantità di dati che viene fornita. Le reti neurali convoluzionali (CNN), in particolare, sono efficaci nell'estrarre dall'immagine di input delle features. Queste possono essere dei bordi o delle forme chiave utilizzate poi per la classificazione della stessa (Vedi Figura 4). Si intuisce come questo tipo di modello sia una componente fondamentale nella risoluzione dei task della computer vision, human parsing compreso. Tutte le architetture sviluppate per risolvere questo task, sebbene varino nella loro implementazione, si basano sull'apprendimento di parametri (raggruppati in *filtri*) che permettono l'estrazione efficace di features dall'immagine. L'apprendimento è un procedimento iterativo in cui la rete cerca di classificare e segmentare le immagini in un

Figura 4: Architettura di una CNN



training set (insieme di dati di addestramento). Ad ogni iterazione, l'output viene confrontato con la ground-truth, cioè la stessa immagine segmentata da un umano, e viene calcolata una stima della qualità della segmentazione sulla base di una funzione di costo che la rete cerca di minimizzare. In base a questa stima, vengono aggiornati i parametri interni della rete, per ottenere un risultato migliore alla prossima iterazione. E' un processo laborioso che richiede una notevole capacità computazionale, ma fornisce ottimi risultati se l'architettura utilizzata è ben progettata e se il dataset utilizzato per l'addestramento è grande e ben annotato. Alcune vengono descritte nel paragrafo dedicato allo stato dell'arte.

Figura 5: Ciclo di apprendimento supervisionato di una CNN [7]



1.2 2D e 3D

Nel corso di questo lavoro di tesi sarà ricorrente il confronto fra segmentazione di immagini 2D e 3D. Nelle immagini 2D ad ogni pixel sono associate due coordinate *(x,y)* per la posizione e un colore espresso in formato RGB. Le immagini 3D (ovvero dati RGB-D) presentano le stesse componenti di un'immagine 2D, a cui viene aggiunta una seconda immagine in scala di grigi (immagine *depth*) che conserva l'informazione della distanza di ogni punto della scena dalla telecamera che acquisisce l'immagine. E' facile intuire come queste ultime siano estremamente rilevanti nelle applicazioni di robotica collaborativa: l'informazione sulla posizione degli oggetti e delle persone dal robot impiegato è fondamentale affinchè ci sia una collaborazione sicura e efficiente. La segmentazione di immagini 3D quindi conferisce al robot la percezione di cosa c'è nell'ambiente intorno a lui, e dove.

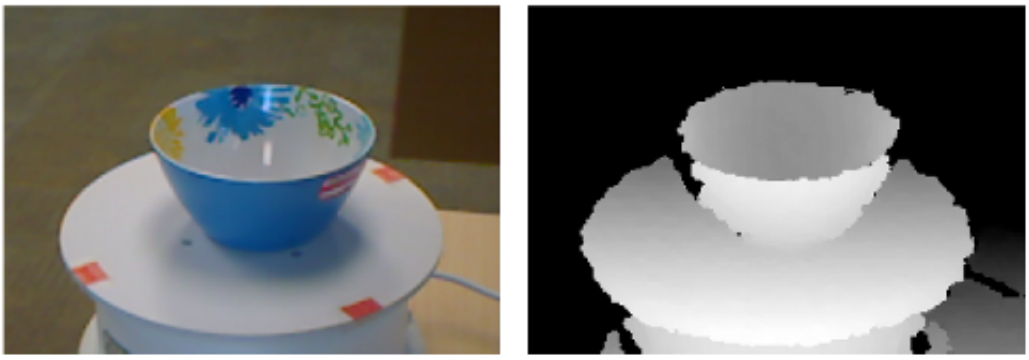


Figura 6: Immagine RGB-D: a sinistra l'immagine RGB, a destra l'immagine *depth*

1.3 Obiettivo

Sono state progettate architetture per fare la segmentazione semantica di immagini 3D [8], [9], [10], [11] dove le classi di interesse possono essere oggetti ed elementi strutturali di una scena (pavimento, muri etc.), ma non sono presenti dataset dedicati per addestrare tali modelli a risolvere il task di human parsing, dove si cerca di segmentare solo rispetto alle varie parti del corpo. Per risolvere il problema si sfrutta ancora la segmentazione 2D. Come proposto in [12], si proiettano segmentazioni 2D provenienti da più telecamere nello spazio 3D, e si ricostruisce la scena. Da qui nasce l'ispirazione per questo lavoro di tesi, ovvero la ricerca, analisi, comparazione e valutazione dei principali tool di annotazione 3D open source disponibili. In questo modo sarebbe possibile confrontare la segmentazione della pointcloud ottenuta tramite l'approccio *multi-view* sviluppato in laboratorio [2], con la sua *ground-truth*. Oltre a questo primo obiettivo, un tool intuitivo e versatile consentirebbe di allenare la rete neurale del robot manipolatore su scenari

[2]Gli esperimenti sono stati condotti nel laboratorio IAS-LAB dell'Università di Padova

specifici: tenere conto, nella fase di addestramento, di situazioni specifiche con oggetti e occlusioni, consentirebbe di aumentare notevolmente le prestazioni del sistema. Infine, creare un ampio dataset 3D permetterebbe di allenare una rete neurale da zero per la segmentazione delle parti del corpo direttamente in 3D.
Il resto della tesi è così organizzato:

- Capitolo 2: viene revisionato lo stato dell'arte su segmentazione e human parsing sia in dati 2D che 3D.
- Capitolo 3: vengono descritti i processi di acquisizione ed elaborazione delle point-cloud ottenute dalla rete di telecamere presente in laboratorio.
- Capitolo 4: vengono analizzati e comparati i vari tool di annotazione 3D.

2 Stato dell'arte

E' molta la ricerca fatta per risolvere il task di human parsing. Con l'avvento del *deep learning*, la quasi totalità dei metodi proposti sfrutta le reti neurali profonde. L'utilizzo di reti neurali richiede, affinchè queste possano essere addestrare, la disponibilità di grandi dataset manualmente annotati. In letteratura sono presenti diversi dataset ([7], [1]) contenenti immagini 2D di scene con persone segmentate rispetto alle varie parti del corpo:

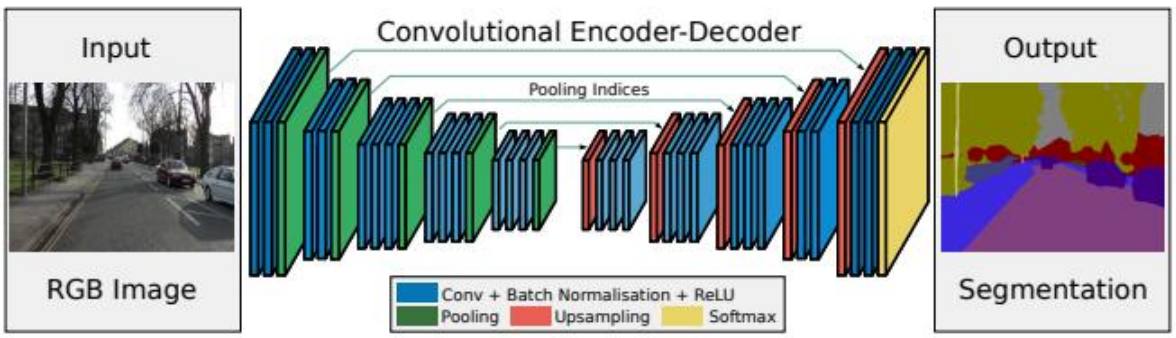
- LIP (*Look Into Person*) [1]: il più vario e più grande, contiene oltre 50000 immagini annotate.
- PASCAL-Person-Part: ne contiene oltre 3000.
- MHP (*Multi Human Parsing*) v2.0 [7]: contiene oltre 25000 immagini con almeno due persone per immagine

Ne consegue, quindi, che si siano sviluppate diverse architetture per fare human parsing su immagini 2D: alcune verranno descritte nel paragrafo seguente. Lo stesso sviluppo non si è riscontrato nell'ambito 3D a causa della mancanza di dataset di pointclouds annotate. Come verrà descritto nel paragrafo dedicato (2.3), si è tentato di risolvere il problema creando dataset sintetici, ma le prestazioni non sono ancora ottimali.

2.1 Modello encoder-decoder

Diverse soluzioni [3], [2], [4] al problema si basano sul modello encoder-decoder[13] (Figura 6), proposto da Long et al. [14] con l'introduzione delle FCN (*Fully Convolutional Networks*). Nella prima fase di *encoding* l'immagine di input subisce un

Figura 7: Modello encoder-decoder delle FCN



processo di *down-sampling*, cioè di graduale riduzione di risoluzione. Vengono generate delle feature maps grazie ai vari livelli di convoluzione, alle quali segue un'operazione di pooling. Le feature maps più esterne hanno risoluzione più alta e conservano le informazioni riguardo i bordi e la posizione degli oggetti. Le features maps

più interne avranno invece una risoluzione più bassa, ma conterranno features ad alto livello che permetteranno di fare predizioni molto accurate sulle classi di interesse. Alla fine del processo di subsampling quindi otteniamo delle *heatmap* (Figura 7) per ogni oggetto, nelle quali ad ogni pixel è associata la probabilità di appartenere a quell'oggetto. Siccome vogliamo ottenere la segmentazione pixel per pixel, è necessario riportare l'immagine alla sua risoluzione originale. Nella seconda parte del modello, quella di *decoding*, l'immagine subisce quindi un processo di *up-sampling*. Una delle tecniche più utilizzate per fare ciò è la *deconvoluzione* (Vedi figura 8): vengono aggiunti dei pixel di *padding*, e poi viene applicato un filtro come in una convoluzione classica. Se applicassimo solo questa operazione, la qualità dell'output sarebbe scadente, perchè

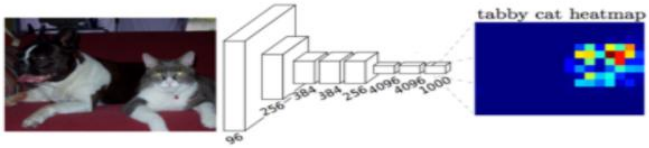
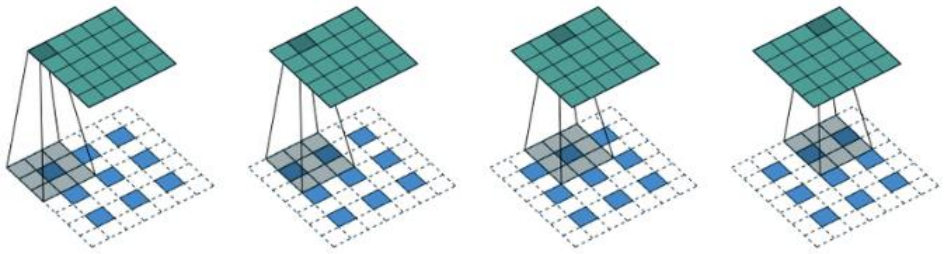


Figura 8: heatmap generata dallo stadio di encoding

Figura 9: Operazione di deconvoluzione

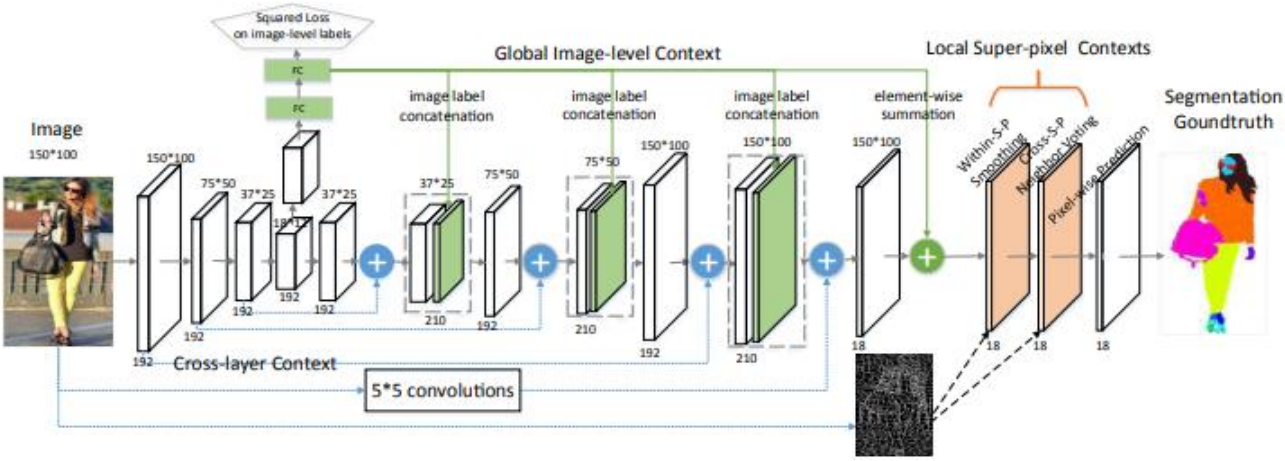


l'immagine di input al processo di up-sampling è a risoluzione molto bassa: il contenuto semantico (le informazioni su cosa c'è nell'immagine) è alto, a discapito delle informazioni spaziali (informazioni sui bordi e dove si trovano gli oggetti). Per questo motivo, si riutilizzano le feature maps molto dettagliate ottenute nella fase di encoding e si combinano con le ricostruzioni graduali dell'immagine. Questo processo è chiamato *skip connections*, e permette di combinare le informazioni semantiche e spaziali (recuperate dai livelli di pooling presenti nella fase di encoding, vedi Figura 6) per ottenere una segmentazione accurata.

2.1.1 Co-CNN

La Co-CNN [3] *Contextualized CNN* è un'architettura basata sul seguente modello (Vedi Figura 9). Nella prima fase di encoding, tramite il down-sampling, vengono generate le feature maps per 4 risoluzioni diverse. Quelle più esterne contengono informazioni spaziali dettagliate, mentre quelle più interne si concentrano sulla struttura globale dell'immagine. Nei livelli FC, viene generato un vettore C-dimensionale, dove C è il numero delle classi di interesse: alla posizione i-esima è contenuta la probabilità che la classe C_i sia presente nell'immagine. Questo vettore sarà utile per la correzione degli errori di

Figura 10: Architettura della Co-CNN



classificazione alla fine del processo. Durante il processo di up-sampling, combiniamo le ricostruzioni dell'immagine con le rispettive feature maps dei primi layer della stessa risoluzione (linea blu tratteggiata). Per ogni livello di up-sampling, inoltre, vengono generate C mappe di probabilità. Nella mappa di probabilità C_i, ad ogni pixel è associata la sua probabilità di appartenenza alla classe C_i. Queste vengono poi concatenate alle feature maps. Il procedimento si ripete per 4 volte, fino a raggiungere la risoluzione originale. L'output a questo livello è composto da C mappe di confidenza, ciascuna riferita ad una classe di appartenenza. Ad ogni mappa, viene sommata pixel per pixel la probabilità che quella determinata classe sia presente nell'immagine. In questo modo, è possibile correggere errori che la rete commette per la mancanza di un contesto globale dell'immagine (Figura 10).

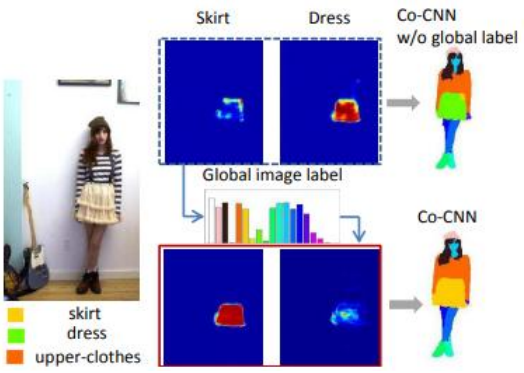
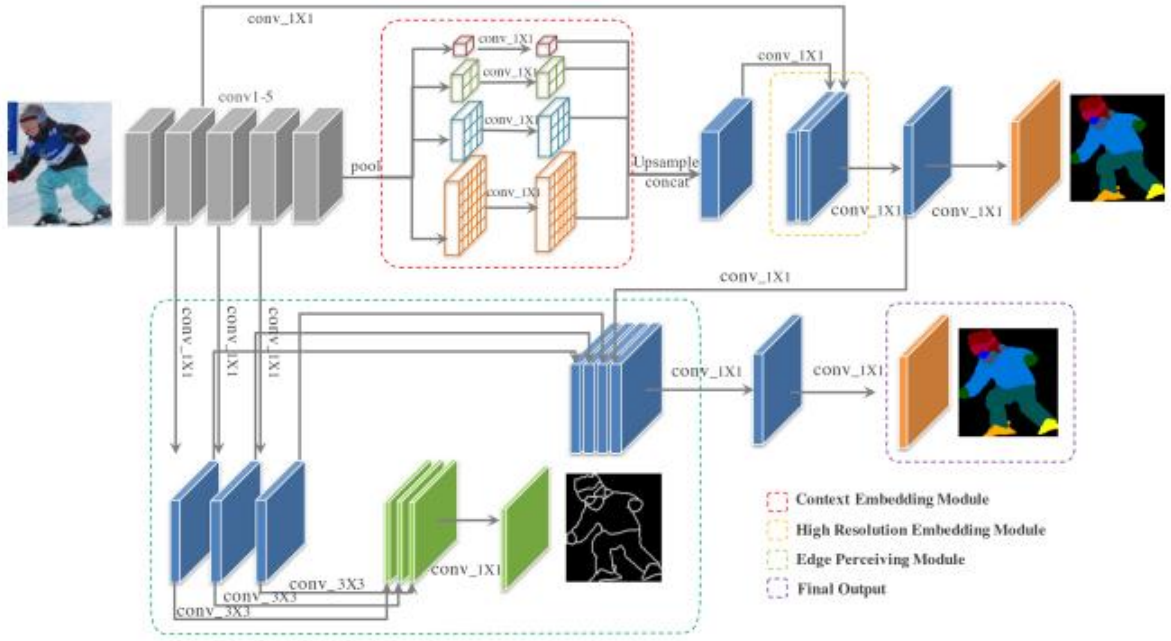


Figura 11: Correzione dell'etichettatura tramite le probabilità globali

2.1.2 SCHP

SCHP [2] (*Self Correction for Human Parsing*) è ottenuta dal miglioramento della CE2P [4], architettura che sfrutta il modello encoder-decoder visto in precedenza. La novità è la presenza di un modulo che impara a riconoscere i contorni molto dettagliati dell'immagine, garantendo una segmentazione accurata. E' composta da 3 moduli

Figura 12: Architettura C2EP



principali:

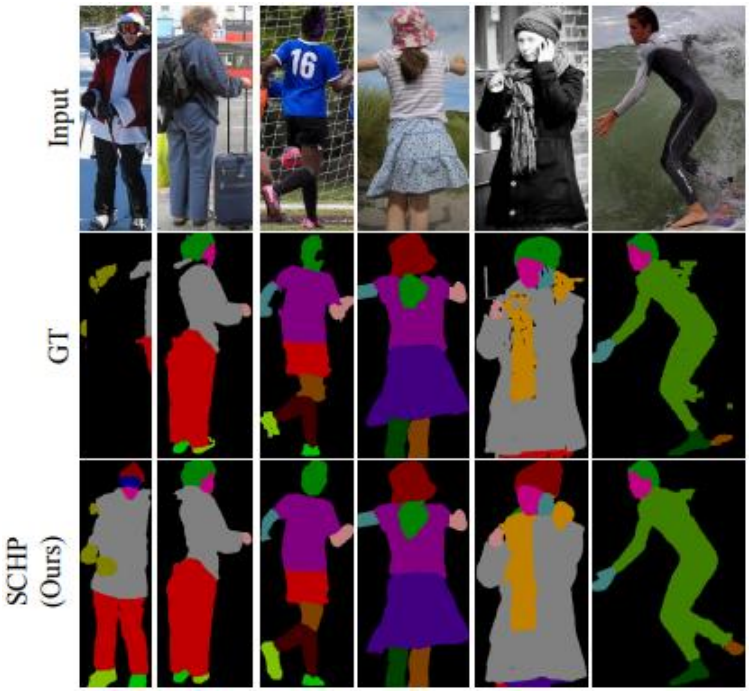
- **Context Embedding Module**: Si occupa di acquisire le informazioni di contesto globali, come la postura e l'orientazione del corpo. Queste, ad esempio, permettono alla rete di riuscire a distinguere un piede destro e sinistro, nonostante siano molto simili nell'apparenza.

- **High-resolution Embedding Module**: Permette di combinare le feature maps molto dettagliate ottenute nella fase di encoding, con le feature maps ad alto contenuto semantico ma di bassa risoluzione. L'output quindi sarà una combinazione di informazioni semantiche e spaziali ad alta risoluzione.

- **Edge Perceiving Module**: Modulo che impara la rappresentazione dei contorni, in modo da rifinire la predizione della segmentazione.

La novità dell'SCHP è quella di introdurre un meccanismo di auto correzione, che permette di rimuovere i pixel rumorosi presenti nelle ground truths del training set in

Figura 13: Risultati di segmentazione di SCHP rispetto alle ground truths



modo da fornire maschere di segmentazione più complete e precise. Spesso, questo meccanismo consente di ottenere risultati migliori delle ground truths. Per questi motivi, ancora oggi SCHP è imbattuta a livello di prestazioni, ed è considerata una delle architetture stato dell'arte per risolvere il task di human parsing.

2.2 Mask R-CNN

In seguito alla pubblicazione della Mask R-CNN [15], sono stati sviluppati diversi metodi [6], [5], [1] basati su questa architettura. La Mask R-CNN è in grado, a partire da un'immagine, di effettuare una segmentazione pixel per pixel degli oggetti di interesse (Figura 13). E' composta da due stadi principali:

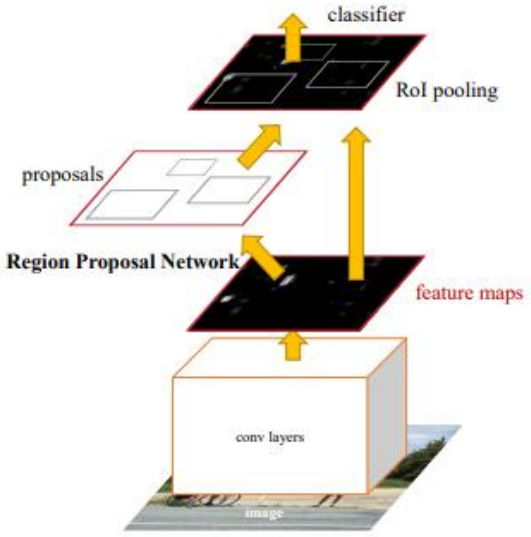


Figura 14: Scheletro di una Mask R-CNN

- RPN (Region Proposal Network): è una CNN che prende in input l'immagine, e restituisce in output un insieme di *proposte di oggetti*, con il relativo score di *objectness*. Vengono quindi proposti i candidati per la segmentazione.
- Fast R-CNN[16]: una rete convoluzionale che, dati in input l'immagine originale e un insieme di proposte di oggetti, restituisce la loro classe di appartenenza e la relativa bounding-box, oltre alla segmentazione pixel per pixel dell'oggetto.

2.2.1 Parsing R-CNN

La Parsing R-CNN [6] è un'architettura che utilizza il modello appena proposto. Viene aggiunto, però, un modulo che consente di segmentare le RoI secondo le varie parti del corpo. E' stata proposta anche una sua rivisitazione, la Renovating Parsing R-CNN [5]. Grazie ai moduli RPN, la gestione delle istanze multiple è semplificata, e i risultati di segmentazione di scene molto affollate sono eccellenti. Questi modelli, infatti, costituiscono lo stato dell'arte per quanto riguarda il *multi* human parsing (Vedi Figura 14).

2.3 3D

Le architetture presentate fino ad ora sono progettate per lavorare solo su immagini 2D. Tuttavia, le tecnologie che operano nel mondo reale in ambito guida autonoma e collaborazione uomo-robot, hanno la necessità di processare dati 3D per percepire la loro distanza dagli oggetti e le persone che le circondano. Il problema di segmentazione di pointclouds suscita quindi molto interesse nella ricerca [8], [9], [10], [11], e ha subito una svolta in seguito alla pubblicazione di PointNet [9]. Questa architettura risolve il problema proposto, ed insieme alle sue rivisitazioni [10] [11]vengono considerate lo stato

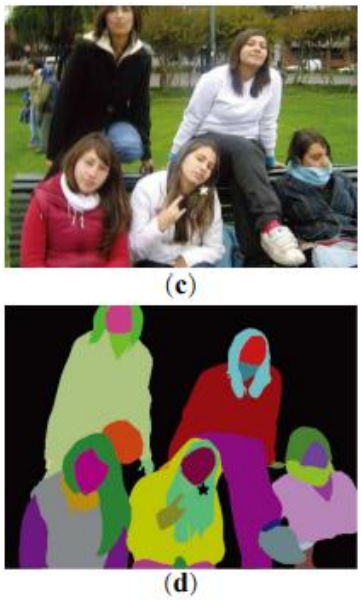


Figura 15: Risultato della segmentazione tramite la Parsing R-CNN

dell'arte per la segmentazione 3D. Le reti che hanno la capacità risolvere il task di human parsing direttamente in 3D quindi ci sono, ma mancano i dati con cui addestrarle. Dataset di scene reali annotate sono ancora assenti, mentre sono stati sviluppati dataset sintetici. *SURREAL* [17] ne è un esempio, e contiene oltre 6 milioni di frame segmentati rispetto alle parti del corpo. I modelli deep learning addestrati su dati sintetici, tuttavia, non riescono a ottenere le stesse prestazioni degli stessi modelli addestrati su dataset reali. Questo fenomeno, denominato *domain gap*, accade perchè è ancora difficile riprodurre artificialmente l'enorme variabilità presente nei dati acquisiti nel mondo reale. Altri approcci al task di human parsing in dati 3D, sfruttano la segmentazione

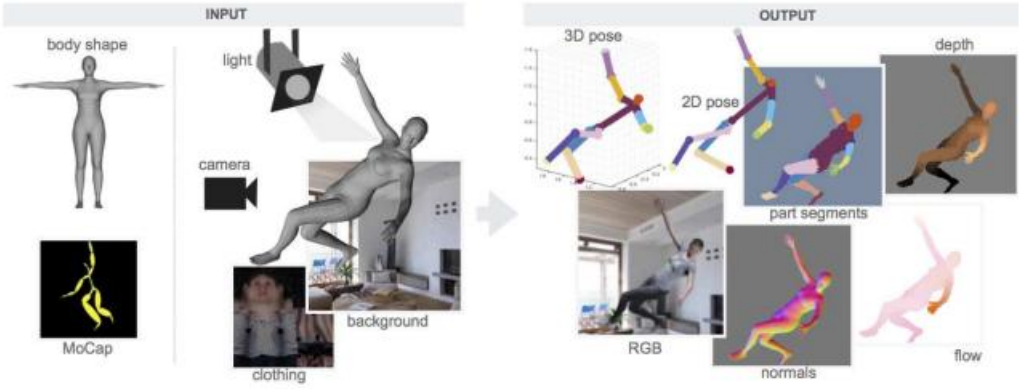


Figura 16: Costruzione sintetica di una scena 3D segmentata

2D. Come proposto in [12] (Figura 16), per mezzo di una rete di telecamere, vengono acquisiti i frame RGB-D da più punti di vista. Quindi, ogni scena 2D viene segmentata rispetto alle varie parti del corpo, per poi essere proiettata nello spazio 3D in un sistema di riferimento comune. Concatenando successivamente le scene provenienti da tutte le telecamere, otteniamo una pointcloud segmentata.

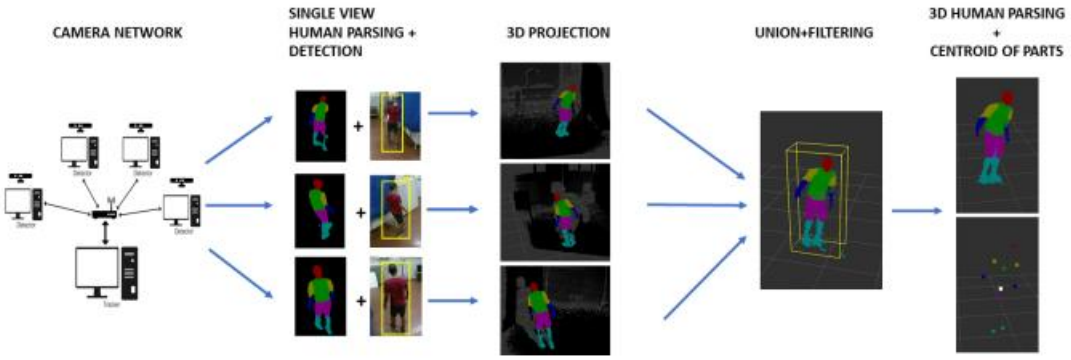


Figura 17: Approccio *multi-view* del problema di human parsing in 3D [12]

3 Esperimenti

In questa sezione verranno descritti gli step necessari per l'acquisizione e l'elaborazione dei dati effettuati prima di testare i vari tool.

Figura 18: Rete di telecamere del laboratorio *IAS-LAB*

3.1 Setup sperimentale

I dati sono stati acquisiti nel laboratorio *IAS-LAB* dell'Università di Padova, dove è presente una rete di 4 telecamere RGB-D *Microsoft Kinect One* (Figura 17). Le telecamere RGB-D (*RGB-Depth*) riescono a percepire l'informazione sulla profondità tramite un sensore a infrarossi. Ad ogni pixel, quindi, è associata la sua distanza dalla telecamera. L'utilizzo di 4 telecamere consente una ricostruzione 3D a 360 gradi della scena, riducendo al minimo la presenza di punti ciechi. Le telecamere sono disposte ai vertici di un rettangolo di dimensioni 3x5 metri (Figura 17). Sono presenti inoltre 4 computer, collegati rispettivamente ad una telecamera ciascuno: 3 di questi sono collegati ad un

computer principale, dove vengono raccolti i dati registrati dalla rete di telecamere. La comunicazione con la rete avviene tramite ROS (*Robot Operating System*), il quale è installato su ogni computer. Due dei componenti principali di ROS, utili a comprendere le fasi di acquisizione ed elaborazione dati, sono due:

- **Nodi**: sono dei programmi eseguibili che utilizzano ROS per comunicare fra loro
- **Topics**: canali di comunicazione che permettono lo scambio di messaggi fra i vari nodi in esecuzione. Hanno un nome univoco e il tipo dei messaggi deve essere specificato. I nodi che generano dati (*publishers*), *pubblicano* su un determinato topic. I nodi che sono interessati a ricevere dati (*subscribers*), si *sottoscrivono*. Ci possono essere multipli publishers e subscribers per lo stesso topic.

3.2 Acquisizione dei dati

Dal computer principale è possibile connettersi tramite la rete locale agli altri 3. Da ogni computer eseguiamo i nodi ROS che pubblicano i dati registrati dalle varie telecamere nei rispettivi 4 topics. Dal computer principale, quindi, possiamo iscriverci ad ognuno di questi topic e salvare le pointcloud provenienti da ogni telecamera in 4 *bag* distinte. Bag è il formato dell'archivio in cui ROS salva i messaggi (in questo caso le pointcloud) provenienti dai vari topic. Inoltre, dobbiamo salvarci le informazioni che permettono di trasformare le pointcloud in un sistema di riferimento comune. Siccome le telecamere non si muovono, queste trasformazioni sono fisse, pertanto possono essere salvate su un semplice file di testo.
Nella mia esperienza ho acquisito le pointcloud per 15 secondi, con un framerate di 30 immagini al secondo: in ogni bag sono state ottenute circa 430 immagini. Alcune (il 5% circa) sono andate perse per i limiti di connessione della rete locale.

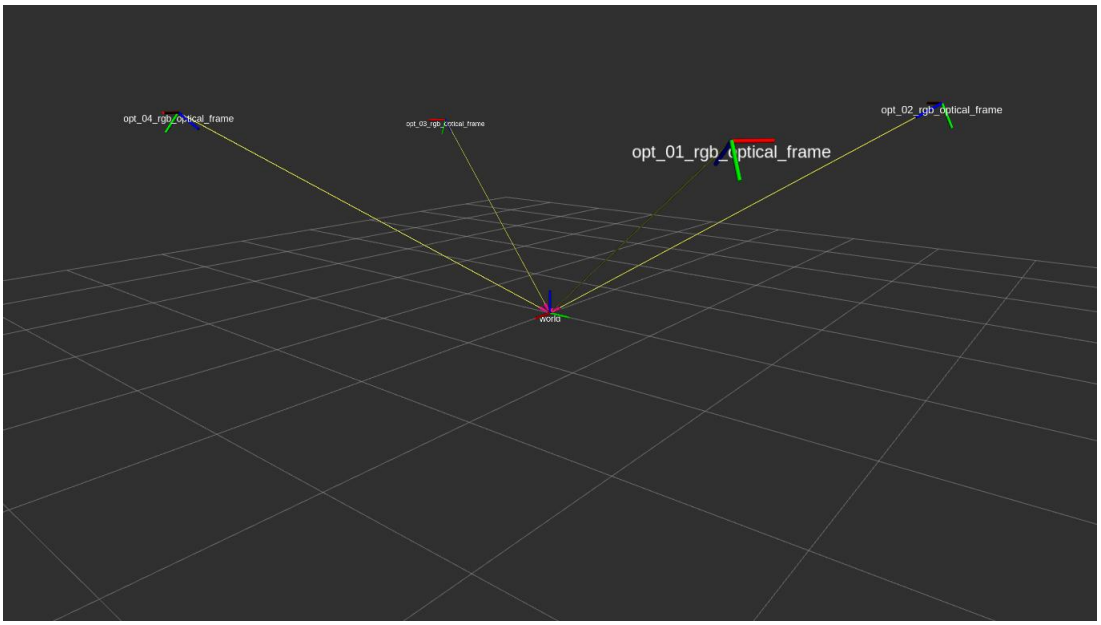
3.3 Preparazione dei dati

Una volta acquisiti i dati, questi devono essere processati per ottenere delle pointcloud che rappresentino l'intera scena. Gli step sono essenzialmente due:

- **Trasformazione**: Le pointcloud acquisite dalle varie telecamere contengono l'informazione di profondità relativa alla telecamera stessa. Per questo motivo, è necessario trasformarle in un sistema di riferimento comune prima di essere fuse per produrre una pointcloud complessiva (Figura 18).
- **Sincronizzazione**: Le telecamere sono temporalmente sincronizzate, ma è comunque presente un ritardo. Quindi non è sufficiente fondere le pointcloud secondo l'ordine di registrazione, ma è necessario effettuare una selezione dei frame in base al loro tempo di acquisizione (*timestamp*). L'istante di acquisizione è memorizzato insieme ad ogni pointcloud.

Ho creato un nodo ROS appositamente per svolgere queste due operazioni. Il nodo si iscrive a 4 topic da cui riceve le pointcloud registrate in laboratorio, e pubblica su un

Figura 19: Posizione dei sistemi di riferimento di ogni telecamera rispetto al riferimento *world* globale.



topic la ricostruzione ottenuta dalla fusione dei 4 punti di vista differenti. E' importante notare che le bag salvate in laboratorio sono riproducibili anche offline, in modo da simulare l'acquisizione come se fosse in tempo reale. Il nodo è costituito da tre funzioni principali:

- **pointcloudsCallback**: Viene chiamata ogni volta che vengono pubblicate le pointcloud sui topic a cui il nodo è iscritto. Tramite un algoritmo di sincronizzazione della libreria ROS, le pointcloud che hanno un timestamp sufficientemente simile vengono automaticamente passate come parametri di questa funzione.

- **transformPointcloud**: Una volta ottenuti i candidati per la ricostruzione della scena, questi vengono trasformati uno ad uno tramite questa funzione, che ha bisogno di ricevere solamente la matrice di trasformazione per passare dal sistema di riferimento della telecamera a quello comune.

- **mergePointclouds**: Riceve in input le pointclouds trasformate, per fonderle e restituire in output una singola pointcloud che le informazioni provenienti da tutte le telecamere.

Ogni ricostruzione viene poi pubblicata su un topic. Queste verranno quindi lette e salvate in una bag. [3]

[3] A causa di errori di calibrazioni nella telecamera numero 1, solo 3 telecamere sono state considerate per la creazione delle pointcloud finali

Figura 20: Flusso di esecuzione del nodo ROS che trasforma, sincronizza e fonde le pointclouds

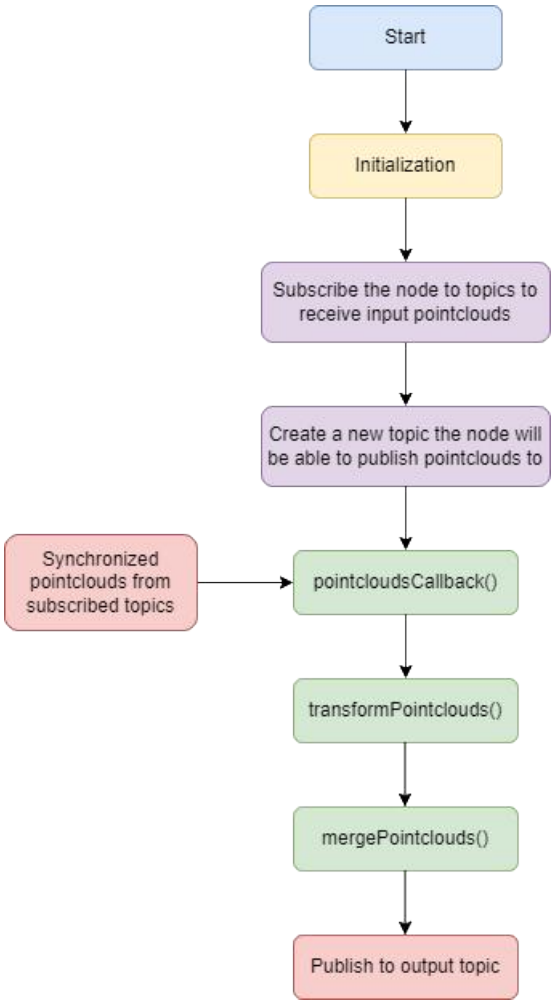
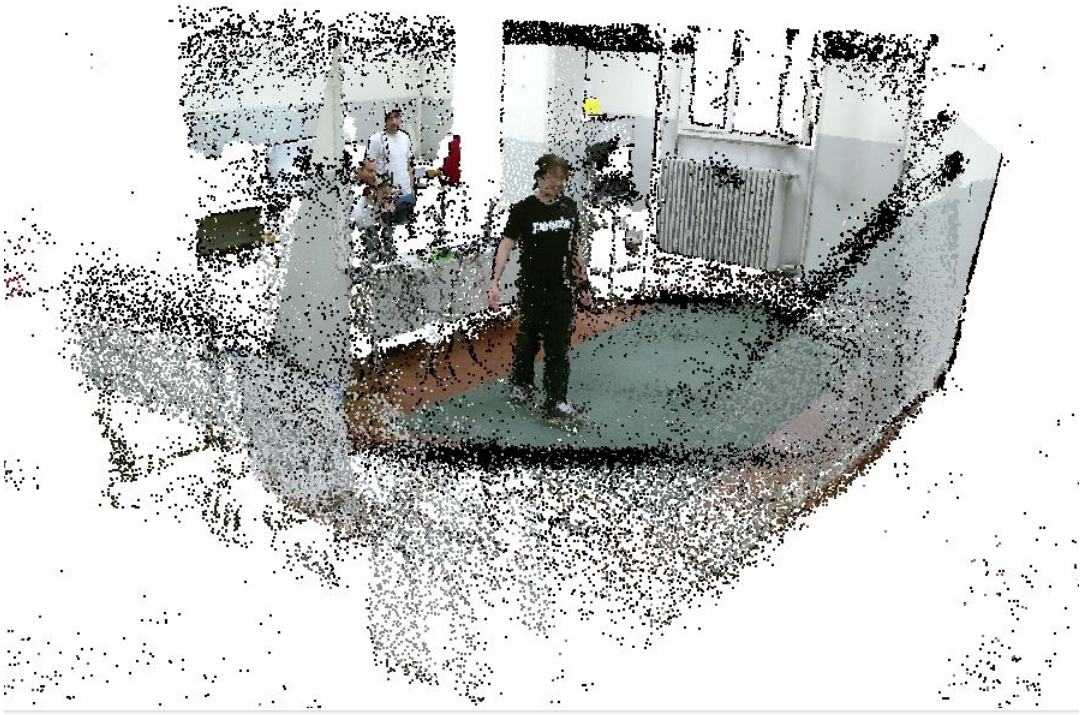


Figura 21: Risultato della ricostruzione della scena fondendo i dati provenienti dalle diverse telecamere



3.4 Risultato

Le pointcloud risultanti sono ordinate e coese. La ricostruzione è quindi di ottima qualità (Figura 20), anche se è presente del rumore, probabilmente dovuto a dei ritardi di sincronizzazione delle telecamere.

4 Analisi dei tool

In questa sezione verranno analizzati e comparati 3 tool di annotazione di pointcloud: Semantic Segmentation Editor [4], LabelCloud [18] e SUSTechPoints [19]. Ogni tool verrà testato su 5 immagini scelte fra tutti i frame acquisiti in laboratorio, sufficienti per visualizzare come si comportano i vari tool in base alle diverse posture e posizioni del soggetto nella scena. I criteri della valutazione saranno:

- Tempo medio di annotazione
- Precisione della segmentazione
- Usabilità
- Presenza di Bug

Le classi con cui verranno segmentate le pointcloud sono 14: testa, busto e differenziati fra destra e sinistra spalle, avambracci, mani, cosce, stinchi e piedi.

4.1 Semantic Segmentation Editor

Semantic Segmentation Editor è un tool basato su un'applicazione web che permette di fare la segmentazione su dati 2D e 3D. E' nato e sviluppato nel contesto della guida autonoma, ma come vedremo si adatta bene anche al task di human parsing. Consente di fare una segmentazione pixel per pixel. Per segmentare un oggetto, è sufficiente selezionare un insieme di pixel a mano libera, e assegnargli una classe.

4.1.1 Interfaccia Utente

Sulla sinistra viene mostrato l'elenco delle classi con cui vogliamo segmentare la pointcloud, con il relativo colore. Il tool rende disponibile un file setting.json in cui è possibile definire il proprio insieme di classi. Vicino al nome di ogni classe è presente il numero di punti della pointcloud assegnati a quella determinata classe.
In alto, abbiamo 4 sezioni:

- Selection Tool: è la modalità con cui selezioniamo un insieme di punti della pointcloud; possiamo scegliere fra mano libera, rettangolo o cerchio.
- Selection Mode: possiamo specificare se l'insieme di punti che selezioniamo va aggiunto o rimosso. Se vogliamo creare un nuovo oggetto, useremo inizialmente l'impostazione "+". Per rimuovere i punti indesiderati, useremo l'impostazione "-".
- View Interaction: Se la casella *Auto Focus* è spuntata, ogni volta che selezioniamo un insieme di punti, viene fatto uno zoom su tale insieme: utile per rifinire un oggetto selezionato. Se la casella *Auto Filter* è spuntata, quando selezioniamo un

[4] https://github.com/Hitachi-Automotive-And-Industry-Lab/semantic-segmentation-editor

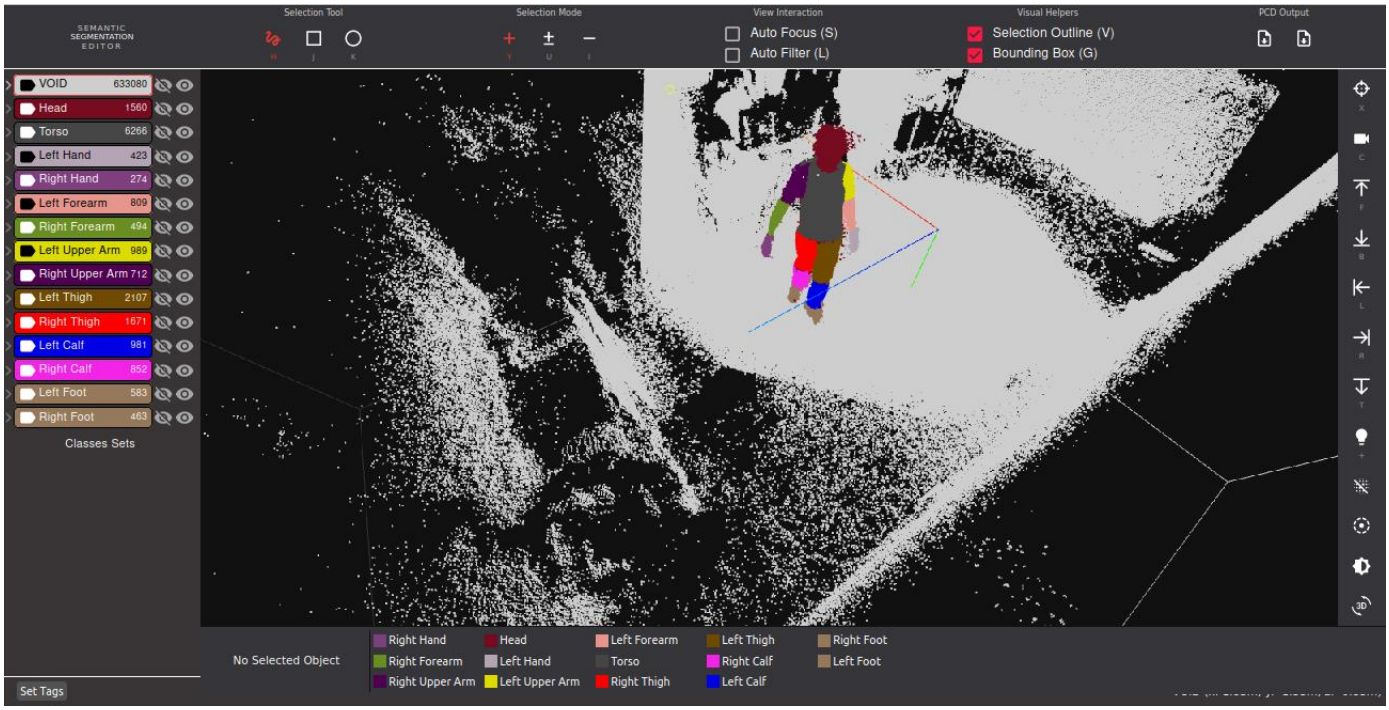


Figura 22: Interfaccia utente di Semantic Segmentation Editor

insieme di punti, tutti gli altri vengono automaticamente resi invisibili, consentendo una segmentazione più pulita e precisa.

- Visual Helpers: poco rilevante, quando la casella Bounding Box è spuntata, viene mostrata una bounding box attorno all'oggetto selezionato.

A destra in alto sono presenti delle funzionalità per osservare da più punti di vista l'oggetto che si sta segmentando. Le funzionalità a destra in basso consentono di modificare la visualizzazione della pointcloud. In ordine dall'alto verso il basso: On/Off dell'RGB della pointcloud, , impostazione della dimensione dei punti, sensibilità e infine modifica del sistema di riferimento. In fondo, vengono mostrati tutti gli oggetti definiti con la relativa classe.

4.1.2 Analisi e risultati

Il tool consente di segmentare le varie parti del corpo in maniera precisa e veloce. L'interfaccia utente fornisce tutti gli strumenti necessari per velocizzare e facilitare l'annotazione. Il tempo medio di annotazione sulle 5 immagini è di 5:30 minuti, quindi ottimo considerando la complessità del task e il numero di classi usate. Il programma è fluido e non si è mai interrotto bruscamente. Tuttavia, non è esente da bug: durante le 5 annotazioni, è capitato 2 volte che alla selezione di una parte del corpo, tutti i punti della pointcloud venissero resi invisibili: è probabilmente un bug legato all'Auto Focus.

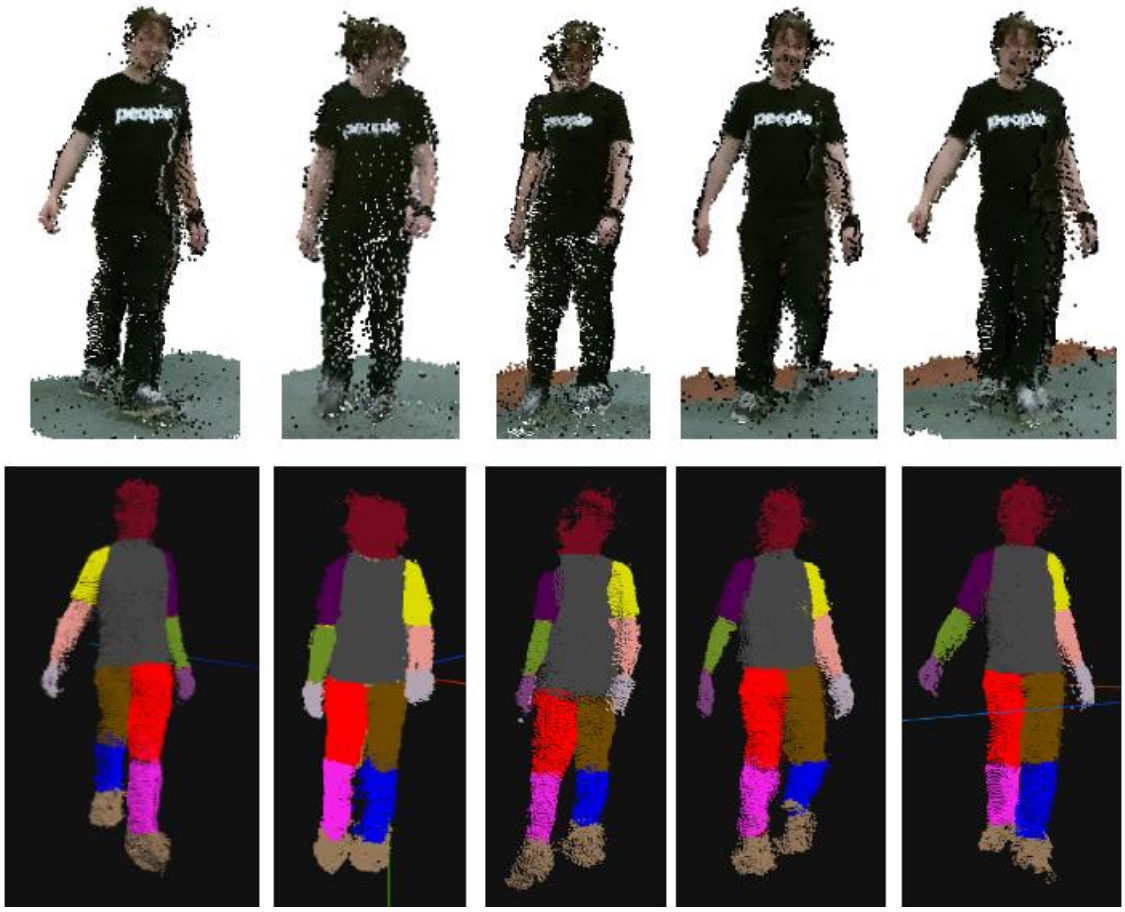


Figura 23: Risultato della segmentazione tramite Semantic Segmentation Editor rispetto alle pointcloud originali

Con la stessa frequenza si è presentato un bug legato alla selezione dei punti: questa ha smesso improvvisamente di funzionare, impedendo l'annotazione di qualunque punto. In realtà è necessario un semplice refresh della pagina per risolvere questi bug: l'annotazione fatta non va persa.
Possiamo vedere dai risultati che queste sono sufficientemente precise e coerenti con la forma delle parti del corpo (Figura 21).

4.2 LabelCloud

LabelCloud è un tool leggero scritto in *python* che permette l'annotazione di pointcloud tramite 3D bounding boxes. Non fa la segmentazione, quindi nel corso del lavoro di tesi è stata sviluppata un'estensione scritta in C++ che assegna ogni punto della pointcloud alla rispettiva classe in base alle posizioni delle bounding boxes.

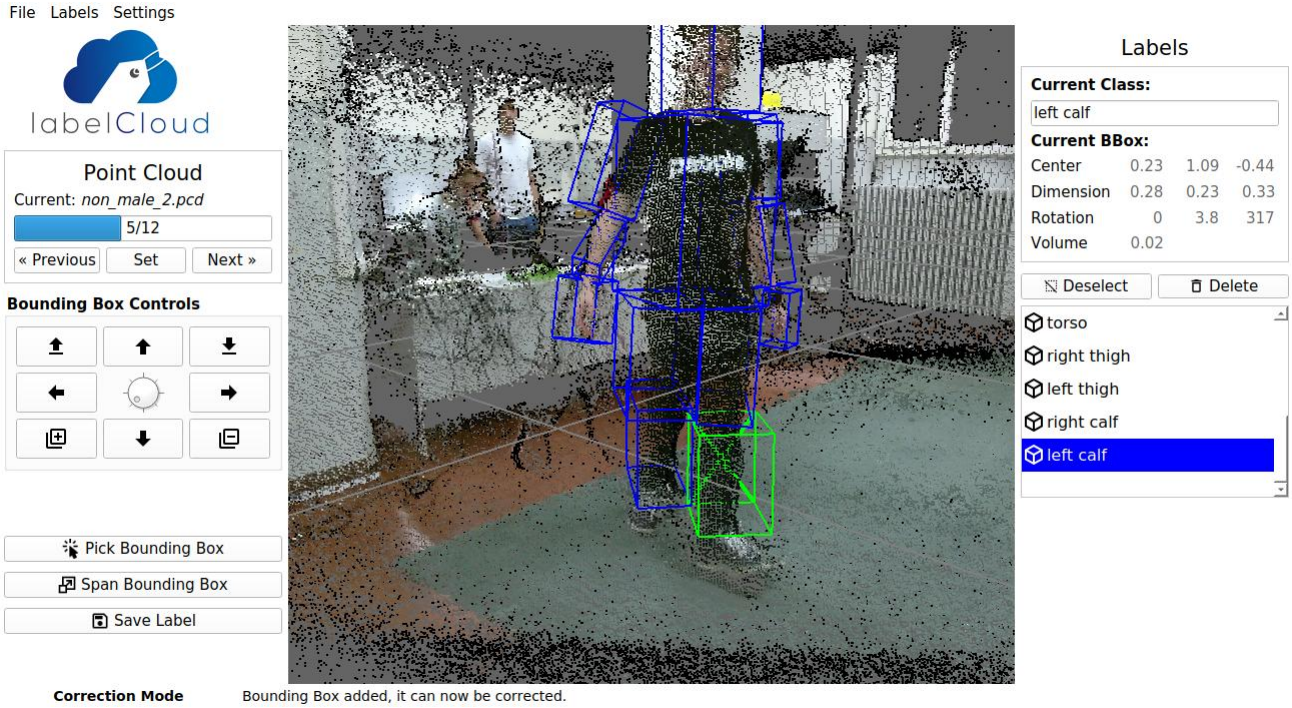


Figura 24: Interfaccia utente di LabelCloud

4.2.1 Interfaccia utente

L'interfaccia utente è molto intuitiva (come riportato in Figura 21). Sulla sinistra, dall'alto verso il basso, abbiamo: il numero di pointcloud da annotare nella cartella di lavoro attuale e un insieme di controlli per ruotare, traslare e ingrandire/rimpicciolire le bounding boxes. Sulla destra, invece, vengono mostrate le coordinate della bounding boxes che stiamo modificando, e più in basso tutte le bounding boxes già create con il relativo label.

4.2.2 Analisi

Il funzionamento del tool è molto semplice: possiamo creare una bounding box e spostarla e modificarla a nostro piacimento. Poi le assegnamo un label. Nel mio caso, in ogni immagine ho creato 14 bounding boxes diverse per ogni parte del corpo. I comandi per interagire con le bounding boxes sono intuitivi, e l'apprendimento è immediato. Il tool è fluido è non ha presentato alcun bug durante il periodo di utilizzo. L'output della annotazione è un file *json* contenente tutte le bounding boxes, descritte dal rispettivo nome e le coordinate degli 8 vertici che le compongono.

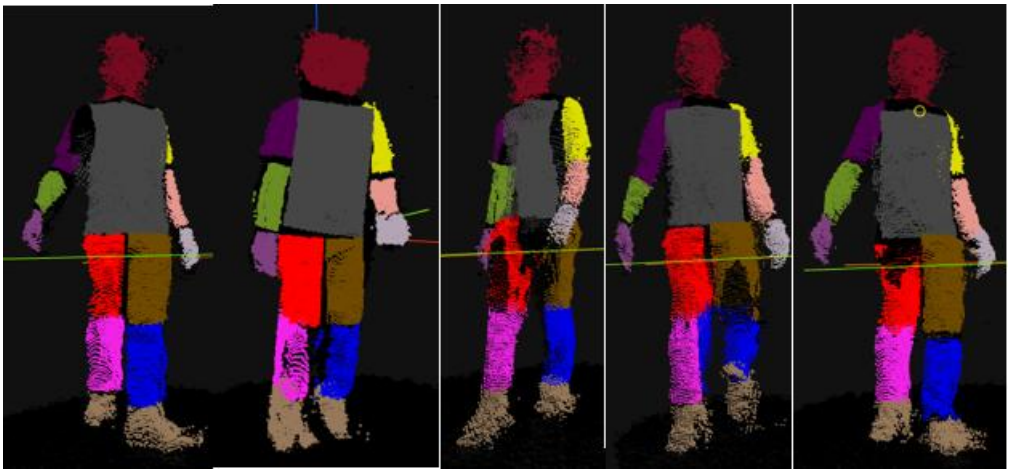


Figura 25: Output della segmentazione tramite l'estensione C++ sviluppata

4.2.3 Algoritmo di segmentazione

Il tool non fa la segmentazione delle pointcloud, ma si limita a creare bounding boxes. Per adattarlo ai nostri scopi ho scritto un programma C++ che fa le seguenti operazioni:

- Legge la pointcloud da segmentare e il rispettivo file json contenente le coordinate delle bounding boxes con le rispettive classi;
- Assegna ogni punto alla classe della bounding box in cui è contenuto, colorandolo.
- Restituisce in output la pointcloud segmentata rispetto alle varie parti del corpo.

I colori assegnati alle varie classi, mostrati in Figura 22, sono gli stessi del primo tool analizzato, in modo da favorire un confronto visivo dei risultati.

4.2.4 Risultati

Il tempo di annotazione medio sulle 5 immagini è di circa 15 minuti. Un comando che velocizzerebbe di molto l'annotazione sarebbe la copia delle bounding boxes: soprattutto per il task di human parsing, data l'ovvia simmetria del problema. Tempo decisamente troppo alto, soprattutto se messo in relazione ai risultati ottenuti. Nonostante l'algoritmo di segmentazione sia preciso, l'annotazione tramite bounding boxes porta a molti più errori rispetto ad una segmentazione pixel per pixel.

4.3 SUSTechPoints

SUSTechPoints è un tool di annotazione *web based* che consente di effettuare la segmentazione di pointcloud per mezzo di 3D bounding boxes. E' nato come piattaforma di annotazione per guida autonoma, ma si adatta discretamente bene al task di human parsing. Tuttavia, alcune funzioni interessanti che dovrebbero ridurre il carico di lavoro, non funzionano come previsto quando è necessario fare una segmentazione per parti.

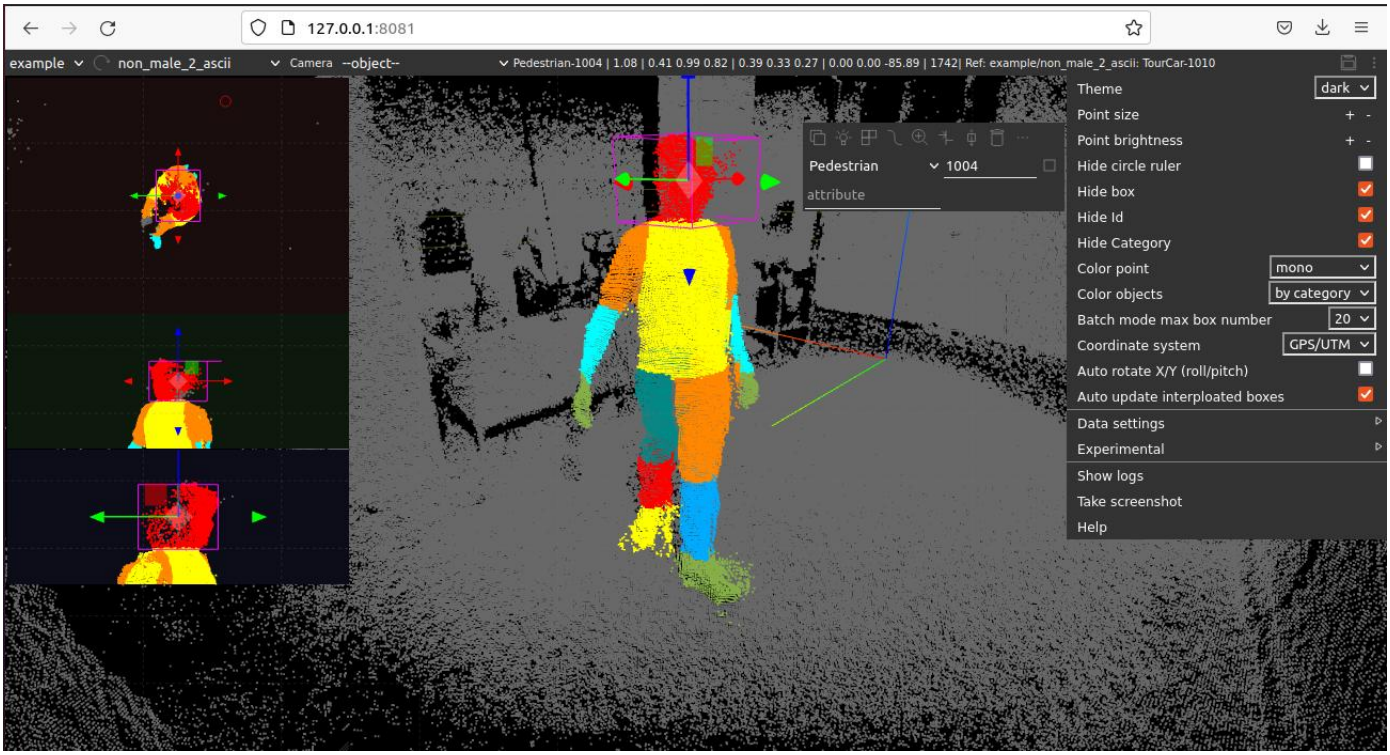


Figura 26: Interfaccia utente di SUSTechPoints

4.3.1 Interfaccia Utente

L'interfaccia utente è leggermente più complessa dei tool precedenti, ma più completa. Oltre alla visione standard della pointcloud, sulla sinistra ci sono 3 viste della bounding box su cui stiamo lavorando da tre prospettive diverse. Queste ci permettono direttamente di traslare, ingrandire o rimpicciolire e ruotare la bounding box da un punto di vista a 360 gradi. Sulla destra è presente un pannello di impostazioni che ci consente di cambiare dimensioni dei punti e la loro luminosità, oppure mostrare/nascondere le bounding boxes e le classi di appartenenza.

4.3.2 Analisi

Per annotare una parte del corpo è sufficiente creare una bounding box e posizionarla e modificarla in base alle proprie esigenze. La vista a 360 gradi velocizza notevolmente questo processo. Essendo un tool sviluppato per la guida autonoma, sono presenti delle funzionalità che facilitano l'annotazione di ampie scene all'aperto. Infatti è presente un'opzione di creazione della bounding box che deduce automaticamente l'orientazione dell'oggetto che stiamo annotando (si pensi ad una macchina ad esempio). Questo però risulta inutile nel momento in cui dobbiamo annotare parti del corpo collegate fra loro, perchè chiaramente l'algoritmo non riconosce dove finisce la testa ed inizia il busto, ad

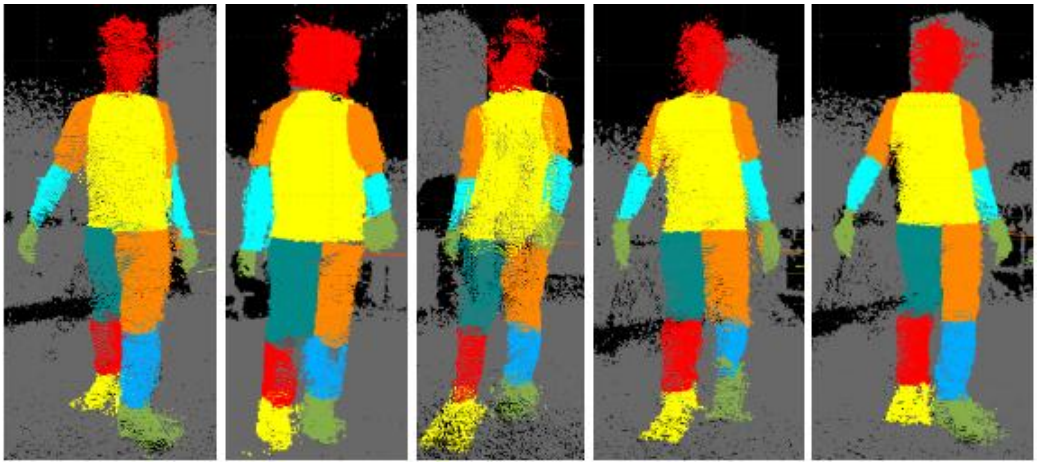


Figura 27: Risultato della segmentazione tramite SUSTechPoints

esempio. Inoltre, SUSTechPoints si propone come tool semi-automatico: nell'ambito guida autonoma, spesso si annotano sequenze di frame strettamente collegate fra loro spazialmente e temporalmente. Il tool quindi mette a disposizione una funzione che, una volta annotato un oggetto, questo venga automaticamente annotato nei frame successivi, riducendo notevolmente il carico di lavoro. Avendo annotato pointcloud temporalmente scollegate fra loro, questa funzione non è stata utilizzata. Un'altro problema di questo tool è l'impossibilità di creare un insieme di classi personalizzato: l'annotazione infatti è stata effettuata su un insieme di classi adatto alla guida autonoma (*Car*, *Bus*, *Pedastrian*...). Per risolverlo è sufficiente modificare il file .json fornito in output con tutte le classi assegnate.

4.3.3 Risultati

Una volta presa dimestichezza con il tool, il tempo di annotazione medio per immagine è di circa 10 minuti. Le segmentazioni risultanti sono tutto sommato precise e coerenti con le immagini originali, considerando che il tool consente l'annotazione solo tramite bounding boxes: le aree in cui è più penalizzato sono i piedi, a causa degli inevitabili punti del pavimento compresi nelle bounding boxes.

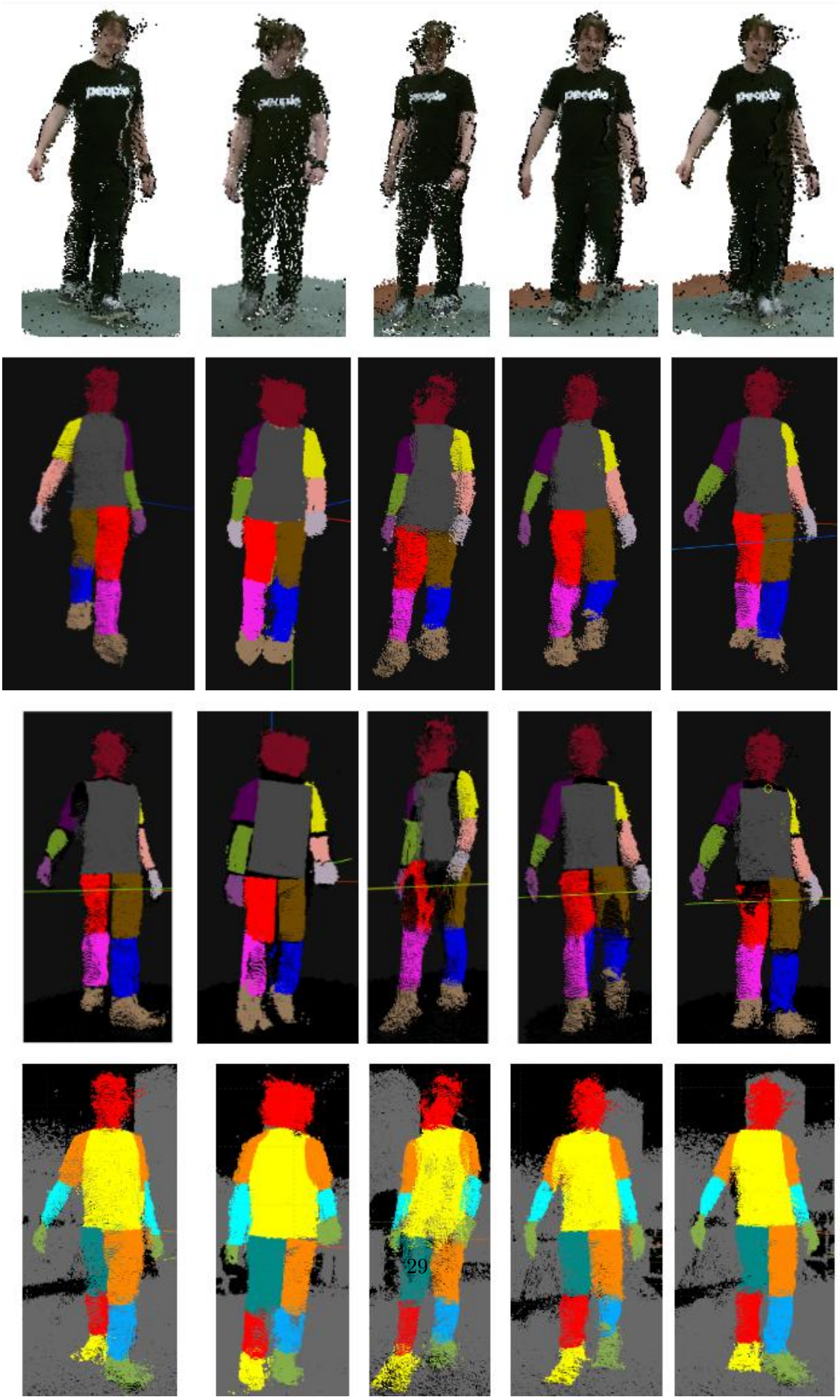
4.4 Comparazione

Vengono riportate nella tabella sottostante alcune caratteristiche e valutazioni dei tool analizzati.

	Tipologia	**Tempo di annotazione medio**	**Insieme di classi personalizzato**	**Presenza di bug**
Semantic Segmentation Editor	**Pixel-wise**	**5:30 minuti**	**Sì**	Sì
LabelCloud	3D Bounding Boxes	15 minuti	**Sì**	**No**
SUSTechPoints	3D Bounding Boxes	10 minuti	No	**No**

Un altro parametro per confrontare i vari tool è chiaramente la qualità della segmentazione in output (Figura 27). Non c'è una metodologia rigorosa per misurarla, ma possiamo analizzare visivamente i risultati prodotti. Semantic Segmentation Editor e SUSTechPoints, nonostante la diversa tipologia, producono entrambi un ottimo risultato rispetto a quello prodotto da LabelCloud. Questo è dovuto al fatto che LabelCloud ci permette solo di posizionare le bounding boxes, senza vedere il risultato della segmentazione in tempo reale e quindi correggere la loro posizione: solo dopo aver eseguito l'estensione sviluppata all'interno della tesi possiamo renderci conto degli errori commessi e questo si nota dalla presenza molto frequente di insiemi di punti non segmentati, in particolare nelle giunzioni fra le varie parti del corpo. Semantic Segmentation Editor, grazie alla possibilità di rifinire parti delicate come i piedi e punti di giuntura di più parti del corpo, ha la meglio su SUSTechPoints, che soffre delle limitazioni imposte dalle bounding boxes. Si noti infatti come i tool a bounding boxes forniscano risultati più scadenti quando gli arti sono più vicini al corpo (immagini 2 e 3), rispetto ai casi in cui le parti del corpo sono spazialmente più distanti fra loro (immagini 1, 3, 4). Un altro punto a favore di Semantic Segmentation Editor è la robustezza alla presenza di occlusioni: i tool basati su 3D bounding boxes richiederebbero infatti una quantità di lavoro eccessivamente maggiore per gestire la presenza di oggetti indesiderati. Bisognerebbe infatti creare delle bounding boxes per aggirare gli ostacoli, ma il tempo di annotazione crescerebbe a dismisura perdendo a priori la competizione con Semantic Segmentation Editor. LabelCloud e SUSTechPoints nascono per essere utilizzati in ambito guida autonoma e, nonostante si addattino discretamente bene al task di human parsing, Semantic Segmentation Editor vince su tutti i punti di vista: il rapporto qualità della della segmentazione su tempo impiegato è decisamente migliore rispetto agli altri tool.

Figura 28: Confronto dei risultati di segmentazione fra i vari tool e le pointcloud originali. I risultati sono nello stesso ordine con cui i tool sono stati analizzati.



5 Conclusioni

In questo lavoro di tesi si è voluto revisionare lo stato dell'arte sulla segmentazione e human parsing, evidenziando le differenze e problematiche che si presentano quando si passa ad affrontare il problema da dati 2D a dati 3D. In particolare, si è fatto notare come la mancanza di dataset 3D annotati rallenti notevolmente lo sviluppo di tecnologie che operano nel mondo reale ed in collaborazione con esseri umani. Alla luce di questi fatti, sono stati analizzati 3 dei principali tool di annotazione 3D open source presenti in letteratura. Ogni tool è stato testato utilizzando un piccolo dataset da 5 pointclouds acquisite in laboratorio tramite un sistema di telecamere che ha permesso un'intera ricostruzione 3D della scena. Dall'analisi delle pointclouds annotate risultanti, si può concludere che sono ottime in relazione alla qualità della segmentazione e il tempo di annotazione speso. Sebbene i tool a bounding boxes si adattino discretamente bene al task di human parsing, Semantic Segmentation Editor consente di effettuare un'annotazione più precisa in un tempo del 50-66% inferiore. E' quindi un ottimo candidato per la creazione di un dataset specializzato, in quanto la segmentazione *pixel-wise* permette di annotare agevolmente anche scene in cui sono presenti oggetti e ostacoli. L'aggiunta di funzionalità e la risoluzione dei bug riscontrati ridurrebbe ulteriormente il carico di lavoro necessario. Quindi, oltre all'acquisizione di un dataset specifico, uno dei futuri sviluppi a partire da questo lavoro di tesi può essere l'estensione ed ulteriore sviluppo dei tool analizzati

Riferimenti bibliografici

[1] Jianshu Li, Jian Zhao, Yunchao Wei, Congyan Lang, Yidong Li, Terence Sim, Shuicheng Yan, and Jiashi Feng. Multiple-human parsing in the wild. *arXiv preprint arXiv:1705.07206*, 2017.

[2] Peike Li, Yunqiu Xu, Yunchao Wei, and Yi Yang. Self-correction for human parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[3] Xiaodan Liang, Chunyan Xu, Xiaohui Shen, Jianchao Yang, Si Liu, Jinhui Tang, Liang Lin, and Shuicheng Yan. Human parsing with contextualized convolutional neural network. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[4] Tao Ruan, Ting Liu, Zilong Huang, Yunchao Wei, Shikui Wei, and Yao Zhao. Devil in the details: Towards accurate single and multiple human parsing. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4814–4821, 2019.

[5] Lu Yang, Qing Song, Zhihui Wang, Mengjie Hu, Chun Liu, Xueshi Xin, Wenhe Jia, and Songcen Xu. Renovating parsing r-cnn for accurate multiple human parsing. In *European Conference on Computer Vision*, pages 421–437. Springer, 2020.

[6] Lu Yang, Qing Song, Zhihui Wang, and Ming Jiang. Parsing r-cnn for instance-level human analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 364–373, 2019.

[7] Ke Gong, Xiaodan Liang, Dongyu Zhang, Xiaohui Shen, and Liang Lin. Look into person: Self-supervised structure-sensitive learning and a new benchmark for human parsing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 932–940, 2017.

[8] Qiangui Huang, Weiyue Wang, and Ulrich Neumann. Recurrent slice networks for 3d segmentation of point clouds. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2626–2635, 2018.

[9] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[10] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[11] Zongyi Wang and Qikun Wang. Object segmentation based on improved pointnet. In *International Conference on Signal Processing and Communication Technology (SPCT 2021)*, volume 12178, pages 336–341. SPIE, 2022.

[12] Matteo Terreran, Leonardo Barcellona, Daniele Evangelista, and Stefano Ghidoni. Multi-view human parsing for human-robot collaboration. In *2021 20th International Conference on Advanced Robotics (ICAR)*, pages 905–912. IEEE, 2021.

[13] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017.

[14] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[15] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[16] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[17] Gul Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 109–117, 2017.

[18] Christoph Sager, Patrick Zschech, and Niklas Kühl. labelcloud: A lightweight domain-independent labeling tool for 3d object detection in point clouds. *arXiv preprint arXiv:2103.04970*, 2021.

[19] E Li, Shuaijun Wang, Chengyang Li, Dachuan Li, Xiangbin Wu, and Qi Hao. Sustech points: A portable 3d point cloud interactive annotation platform system. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1108–1115. IEEE, 2020.