UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Università degli Studi di Padova

Corso di Laurea Magistrale in Ingegneria Informatica

Robotica Autonoma A.A. 2016/2017

Tesina Robotica

Professore:
E. Pagello
PhD:
S. Michieletto
M. Carraro

Autore:
Iacopo Mandatelli, 1151791

Indice

1 Scopo

Lo scopo di questa tesina è quello di raccogliere un dataset dopo l'opportuna configuraione di una rete di sensori e di misurare la qualità dei risultati forniti da un'algoritmo di pose recognition (riconoscimento delle pose assunte da una persona) applicato al suddetto dataset. Successivamente sono state verificate le combinazioni di parametri che forniscono i risultati migliori ed infine si è cercato di migliorare l'algoritmo di pose recognition inizialmente utilizzato.

2 Configurazione delle macchine e dei sensori

La rete di computer e sensori è composta da quattro macchine [6], in cui una ha il ruolo di Master e le altre tre di Slave, collegate ad un router tramite cavo ethernet; ad ognuna è inoltre collegato un kinect v2 [5] che al suo interno ha una camera RGB ed un sensore di profondità.
I quattro sensori sono stati sospesi su dei cavalletti e sono stati leggermente inclinati verso il basso, per coprire una zona limitata e definita del laboratorio. La vista in pianta del laboratorio è visibile nell'immagine sottostante: il campo visivo dei vari sensori si sovrappone in buona parte dello spazio utile, in modo da consentire il tracking continuo delle persone all'interno dell'area considerata.
Lato software su ogni macchina è stata eseguita l'installazione di ROS, delle librerie CUDA, della libreria OpenPTrack [3], dei driver per i sensori Kinect v2 e successivamente è stata fatta la sincronizzazione tra le macchine usando il protocollo NTP.
Sono poi stati configurati tutti i sensori, tramite l'uso di una scacchiera di riferimento (spostata manualmente all'interno dello spazio di lavoro), sono state calcolate le trasformate tra una camera e l'altra, fino ad ottenere un albero delle trasformazioni tra tutti i sensori; infine è stata posta la scacchiera sul pavimento, visibile ad almeno due camere, in modo da definire il sistema di riferimento del mondo per la rete di sensori (i relativi file di configurazione sono stati salvati su tutte le macchine).
Per ultimo sono state osservate le point cloud dei vari Kinect ottenute dopo la configurazione e sono state finemente allineate a mano, in modo da ottenere poi un tracking delle persone il più preciso possibile (dall'esperienza di colleghi è emerso che questa fase è cruciale per ottenere poi buoni risultati con il tracking e il riconoscimento delle pose delle persone nella scena).

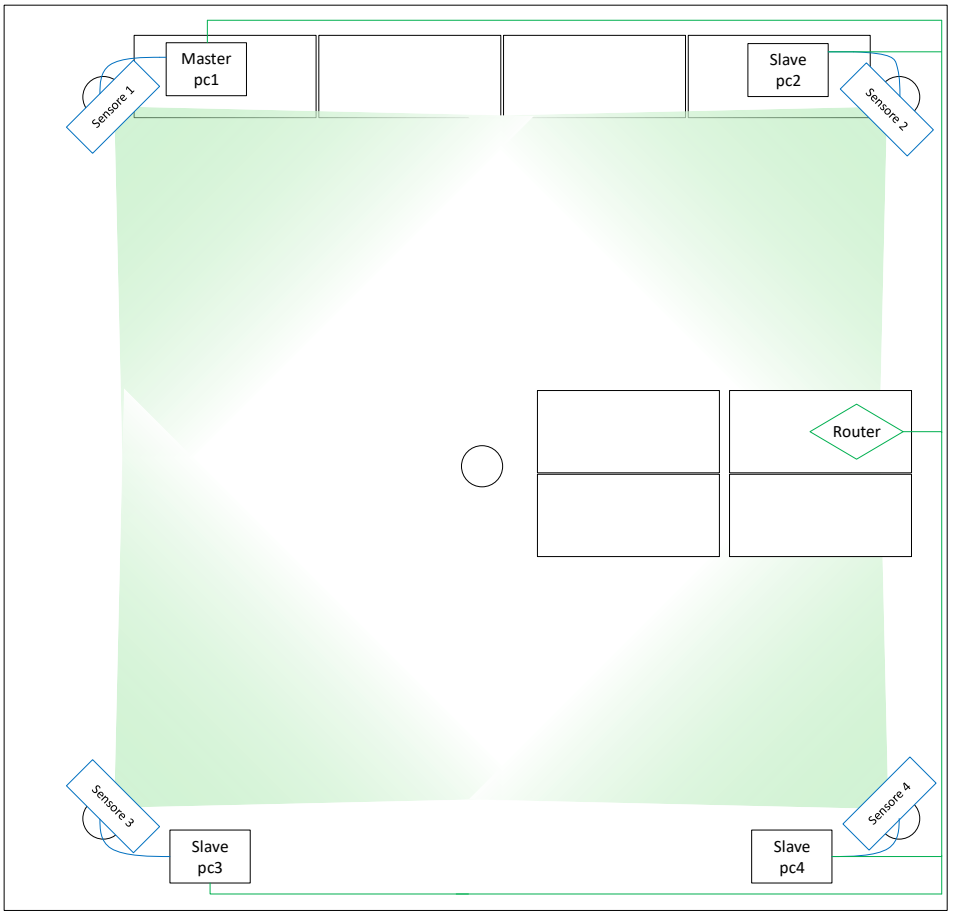


Figura 1: Schema del laboratorio e della disposizione di macchine e sensori.

3 Acquisizione del dataset

Il dataset consiste nella registrazione in formato *rosbag* di vari messaggi ROS pubblicati sui rispettivi topic (vedi sotto), ed in particolare il messaggio che rappresenta le pose è di tipo *opt_msgs/PoseRecognitionArray* pubblicato sul topic */recognizer/poses*: il messaggio esprime per ogni scheletro la miglior posa riconosciuta ed il relativo score.
Nel dataset sono state assunte alternativamente circa 8 pose (delle 4 disponibili nella galleria di pose) per una durata di circa un minuto.

3.1 Registrazione sequenza

Per registrare il dataset, sul Master è stato lanciato il nodo ROS `tracking_node` della libreria OpenPTrack, mentre su tutte le macchine con un sensore collegato (compreso il master quindi) è stato lanciato il nodo ROS `detection_node`.
Il dataset, comprendente la sequenza di pose, consiste in un file *record_01.bag* registrato tramite il comando:

```
rosbag record -b0 -O ~/record_01.bag /tf /detector/skeletons
/kinect_01/depth_lowres/camera_info
/kinect_01/depth_lowres/image/compressedDepth
/kinect_01/rgb_lowres/camera_info /kinect_01/rgb_lowres/image/compressed
/recognizer/markers /recognizer/poses /tracker/skeleton_tracks
/tracker/standard_skeleton_tracks
```

Il file così ottenuto ha una dimensione di circa 2 Gb e contiene dai 1500 ai 1800 messaggi per ciascun topic (variano in base alla frequenza di pubblicazione).

3.2 Descrizione pose e scheletri

Ogni posa (locata nella cartella /open_ptrack/body_pose_recognition/gallery_poses/data) è composta da una o più immagini rappresentanti lo scheletro corrispondente alla posa, e da un file txt con le componenti x, y e z di ogni giunto dello scheletro. In Figura 2 un esempio di posa con il braccio destro alzato.

Figura 2: Esempio di configuazione dello scheletro per la posa braccio destro alzato.

Lo scheletro, composto da 14 *SkeletonJoints* che vanno a formare altrettanti *SkeletonLinks*, è definito come:

```
std::vector<std::pair<SkeletonJoints, SkeletonJoints>>
SkeletonLinks::links =
{
  std::pair<SkeletonJoints, SkeletonJoints> (HEAD, NECK),
  std::pair<SkeletonJoints, SkeletonJoints> (NECK, RSHOULDER),
  std::pair<SkeletonJoints, SkeletonJoints> (RSHOULDER, RELBOW),
  std::pair<SkeletonJoints, SkeletonJoints> (RELBOW, RWRIST),
  std::pair<SkeletonJoints, SkeletonJoints> (NECK, LSHOULDER),
  std::pair<SkeletonJoints, SkeletonJoints> (LSHOULDER, LELBOW),
  std::pair<SkeletonJoints, SkeletonJoints> (LELBOW, LWRIST),
  std::pair<SkeletonJoints, SkeletonJoints> (CHEST, RHIP),
  std::pair<SkeletonJoints, SkeletonJoints> (RHIP, RKNEE),
  std::pair<SkeletonJoints, SkeletonJoints> (RKNEE, RANKLE),
  std::pair<SkeletonJoints, SkeletonJoints> (CHEST, LHIP),
  std::pair<SkeletonJoints, SkeletonJoints> (LHIP, LKNEE),
  std::pair<SkeletonJoints, SkeletonJoints> (LKNEE, LANKLE),
  std::pair<SkeletonJoints, SkeletonJoints> (NECK, CHEST)
}
```

3.3 Creazione ground truth

La ground truth per la sequenza di pose è stata creata esaminando i messaggi rappresentanti l'immagine registrata dalle camere, di tipo
/kinect_01/rgb_lowres/image/compressed.
Tale operazione è stata resa possibile tramite l'utilizzo dello strumento *rqt_graph*.

La ground truth per una sequenza di pose è rappresentata da un file in formato *csv*, in cui il primo elemento è l'istante di inizio della posa, il secondo elemento è l'istante di fine e il terzo è l'id della posa $(3.5; 8.2; posa3)$; le righe sono ordinate temporalmente e soddisfano la condizione $t_{fine}(posa_i) \leq t_{inizio}(posa_j), i < j$, come nell'esempio:

```
3.3;7.0;arms_mid
8.6;12.5;arms_up
14.0;18.1;right_arm_up
19.1;22.7;left_arm_pointing
24.7;29.4;arms_mid
```

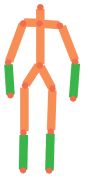
4 Analisi delle performances

Con "analisi delle performances" di un algoritmo di pose recognition si intende la misurazione del grado di precisione con cui il sistema riesce ad individuare le pose assunte da una persona all'interno di una sequenza temporale.
L'insieme delle pose valide è definito a priori, ed è formato da quelle che sono state registrate e sono quindi presenti in libreria (nel nostro caso sono 4), oltre alla posa "unknown", utilizzata quando non viene riconosciuta alcuna posa nota.

4.1 Algoritmo di pose recognition

Per identificare la posa assunta da una persona, l'algoritmo di pose recognition della libreria OpenPTrack [3] confronta lo scheletro della persona nella scena (messaggio ROS di tipo */tracker/standard_skeleton_tracks*) con tutti gli scheletri rappresentanti le varie pose presenti nella galleria.

In particolare al fine del riconoscimento vengono considerati solamente i due avambracci e le due gambe, come riportato nella figura a fianco: l'algoritmo calcola la distanza euclidea tra i links dello scheletro della persona e quelli dello scheletro di riferimento nella libreria delle pose, ottenendo quindi un *similarity_score* (che può essere il migliore, il peggiore o la media dei giunti considerati) per ciascuna posa. Tanto più la posa coincide con una nota, tanto più lo score sarà basso, fino ad arrivare a zero nel caso di coincidenza.



L'algoritmo quindi restituisce come posa identificata quella che ha il *similarity_score* più basso delle altre, a condizione che sia inferiore di un certo *threshold*, altrimenti viene restituita la posa di default "unknown".

4.2 Matrice di confusione e metriche

Per calcolare le metriche che misurano il grado di accuratezza dell'algoritmo viene innanzitutto compilata la *confusion matrix* (o matrice di confusione), ossia una tabella in cui le righe rappresentano le pose predette dall'algoritmo mentre le colonne rappresentano le pose attualmente assunte dalla persona. Queste ultime vengono determinate manualmente come visto nella sezione 3.3 e costituiscono la ground truth.
Un esempio di confusion matrix è riportata nella tabella 1: la cella blu rappresenta il numero di volte in cui l'algoritmo riconosce correttamente la posa2 (true positive,

TP) , quelle nere rappresentano il numero di volte in cui l'algoritmo, correttamente, non riconosce la posa (true negative, TN), mentre in verde e rosso sono rappresentati rispettivamente i falsi positivi e i falsi negativi (FP e FN) [2].

		reali				
		posa0	posa1	posa2	posa3	unknown
predette	posa0	314	0	0	0	70
	posa1	0	127	0	0	0
	posa2	0	0	45	0	2
	posa3	1	0	0	271	26
	unknown	93	61	159	27	589

Tabella 1: Esempio di TP, TN, FP e FN per la posa2.

Le metriche considerate per misurare il grado di precisione dell'algoritmo sono:

- $Precision \triangleq \frac{TP}{TP+FP}$
- $Accuracy \triangleq \frac{TP+TN}{TP+TN+FP+FN}$
- $Recall \triangleq \frac{TP}{TP+FN}$

4.3 Algoritmo per il calcolo delle metriche

Il programma calcola le tre metriche sulla base della confusion matrix compilata per una particolare sequenza registrata.
Innanzitutto viene letto da un file di configurazione la posizione ed il nome del file in formato rosbag che raccoglie vari messaggi ROS (ed in particolare la sequenza di pose restituite dall'algoritmo di pose recognition) e del file in formato csv che rappresenta la ground truth (sezione 3.3).
Vengono quindi letti sequenzialmente i messaggi ros che rappresentano le pose, di tipo */opt_msgs/PoseRecognitionArray*, e l'istante temporale corrispondente; ad ogni posa p_i corrisponde un istante temporale t_i: il programma va quindi a cercare nella ground truth l'intervallo temporale che contiene l'istante t_i e recupera la reale posa r_i corrispondente. Tale posa reale r_i fornirà l'indice di colonna nella confusion matrix mentre l'indice di riga sarà determinato dalla posa stimata p_i; procedendo progressivamente $\forall\ i \in |\#posemessages|$ viene compilata tutta la matrice di confusione.
A questo punto non rimane che calcolare accuracy, precision e recall per ciascuna posa (eccetto per la posa "unknown") come mostrato nella sezione 4.2, per poi determinare la media di ogni singola metrica sulle varie pose. Tali valori saranno la base per l'analisi condotta successivamente.

4.4 Qualità dei risultati in base alla combinazioni di parametri

Di default l'algoritmo di pose recognition considera solo gli avambracci per riconoscere la posa assunta dalla persona e come politica di fusione degli score dei singoli giunti (*per_skeleton_score_fusion_policy*) tiene lo score peggiore tra quelli considerati; come threshold sul valore dello score della posa migliore per soddisfare la condizione soglia, il valore di default è 1.5 e nella tabella 2 tale configurazione standard verrà indicata con **1/top/1.5**. E' possibile osservare come sia precision che accuracy hanno valori molto alti, maggiori del 90% già con la configurazione di default dei parametri, mentre il recall ha un valore più basso: questo è dovuto alla presenza di numerosi falsi negativi.
Innanzitutto si è provato a considerare non solo i due avambracci ma anche le due gambe al fine del riconoscimento della posa, mantenendo inalterati gli altri due parametri: tale configurazione, indicata con **1/all/1.5** produce un peggioramento della precision di circa il 3% e del recall del 5% (l'accuracy rimane sostanzialmente invariata). Questa configurazione quindi produce risultati peggiori rispetto a quella standard.
Successivamente è stata provata la regolazione del parametro di fusione degli score dei singoli giunti sul valore 0, ossia l'algoritmo di pose recognition restituisce come score della posa la media dei singoli giunti che la rappresentano, invece di considerare il valore peggiore come da impostazione standard; tale configurazione, indicata con **0/top/1.5**, produce un peggioramento dalla precision di circa il 7% e un miglioramento dell'accuracy del 3%, ma il recall cresce sensibilmente, passando da 0.713 a 0.913 (+28%). Ciò è determinato dalla grande riduzione del numero di falsi negativi per le varie pose.
Infine è stato analizzando l'andamento delle metriche considerate al variare del threshold sul valore della posa migliore: all'aumento della soglia corrisponde un margine più ampio di accettazione, ossia vengono restituite pose che più si discostano da quella di riferimento presente in libreria. Conseguentemente all'aumentare del threshold la precision diminuisce ma accuracy e recall aumentano. Il miglior compromesso sembra essere la scelta del valore 1.8, con un aumento di oltre il 3% dell'accuracy, un lieve peggioramento della precision e un'aumento del 22% del recall rispetto alla configurazione di parametri di default.
Da quanto osservato quindi, singole configurazioni massimizzano alcune metriche, ma dopo l'analisi dei risultati è possibile affermare che il valore del threshold può essere aumentato rispetto alla configurazione standard, andando ad aumentare le performances dell'algoritmo di pose recognition. Nella tabella sottostante sono riportate tutte le combinazioni di parametri esaminate e i valori delle metriche rispetto a tali configurazioni, oltre alla differenza percentuale rispetto alla configurazione

standard scelta dagli sviluppatori della libreria. Gli incrementi percentuali sono più significativi di quanto si sarebbe portati a pensare al primo sguardo, in quanto precision, accuracy e recall ottenute dall'algoritmo di pose recognition sono già molto buone con l'impostazione stardard dei parametri.

configurazione	precision	accuracy	recall	% prec	% acc	% rec
1/top/1.5	0.928	0.940	0.713	-	-	-
1/all/1.5	0.898	0.945	0.674	-3.2	0.5	-5.5
0/top/1.5	0.858	0.966	0.913	-7.5	2.7	28.1
0/all/1.5	0.685	0.908	0.949	-26.3	-3.5	33.0
1/top/1.2	0.910	0.914	0.421	-1.9	-2.8	-40.1
1/top/1.6	0.925	0.963	0.787	-0.35	2.4	10.4
1/top/1.7	0.914	0.964	0.832	-1.5	2.6	16.6
1/top/1.8	0.913	0.970	0.872	-1.6	3.2	22.3
1/top/1.9	0.872	0.963	0.880	-5.9	2.4	23.4
1/top/1.9	0.871	0.962	0.919	-6.1	2.3	28.8
1/all/1.8	0.910	0.968	0.862	-1.9	2.9	20.8
0/top/1.8	0.738	0.936	0.939	-20.5	-0.5	31.7

Tabella 2: Valori delle metriche considerate per ciascuna combinazione di parametri, con differenza percentuale rispetto alla configurazione di default. In **grassetto** sono evidenziate le configurazioni più significative.

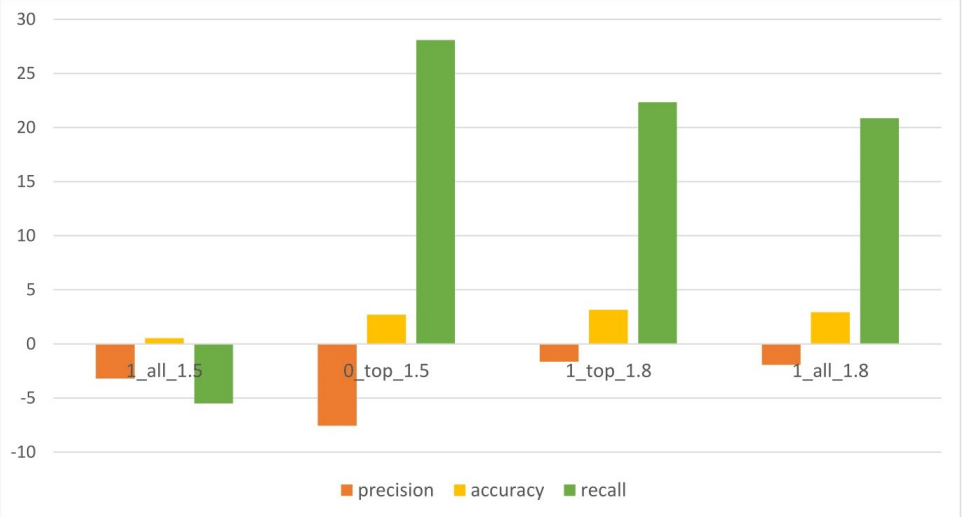


Figura 3: Differenza percentuale delle metriche tra le configurazioni più significative e quella di default.

5 Analisi e Miglioramento dell'algoritmo

La fase finale dell'esperienza è stata quella di cercare di migliorare i risultati ottenuti dall'algoritmo di pose recognition con le impostazione ed il funzionamento di default.

5.1 Rimozione degli outliers

Da un'analisi manuale delle pose prodotte dall'algoritmo, è possibile notare come all'interno di una sequenza temporale in cui il soggetto assume la posa a braccia aperte, *arms_mid*, capita che per un istante temporale la posa stimata sia diversa da quella riconosciuta nel resto della sequenza. Nella sequenza di esempio sottostante, ordinata temporalmente, il valore "0" rappresenta la posa *unknown*, prodotta dall'algoritmo quando non viene riconosciuta alcuna posa nota, mentre il valore "1" rappresenta la posa *arms_mid*.

000000000111111111111111111111101111110000000000

Visionando la sequenza di scheletri e pose in RVIZ, appare chiaro che si tratti di un inconveniente tecnico, ossia lo score della posa sale appena sopra il threshold menzionato precedentemente (sezione 4.1): per una frazione di secondo, figura 4b, viene persa la posa appartente alla sequenza, 4a; ad ulteriore conferma di quanto detto, in figura 4c è riportata l'immagine al corrispondete istante temporale (del frame 4b) catturata dalla camera.

(a) (b) (c)

Figura 4: Pose e immagini prese da RVIZ ed apparteneti alla medesima sequenza di pose.

Per ovviare alla presenza di questi outliers durante l'esecuzione, è stata aggiunta all'algoritmo standard la memorizzazione delle ultime tre pose assunte dallo scheletro: nel caso in cui le tre pose precedenti siano tutte uguali tra loro e la posa corrente predetta sia diversa, essa viene cambiata e la posa restituita sarà uguale

a quelle precedenti. Le pose memorizzate vengono quindi modificate, ossia le due meno recenti vengono scalate di una posizione e la più recente viene aggiornata.
Questa politica opera a valle di tutte le operazioni fatte dall'algoritmo e produce i suoi effetti poco prima della pubblicazione delle pose di ogni scheletro nel messaggio ROS corrispondente.
Con questo metodo è possibile eliminare un singolo outlier, al costo di ritardare in alcuni casi l'inizio di una nuova sequenza di un istante temporale.
La configurazione con l'algoritmo modificato appena presentato è denominata **def_out** nella tabella 3, e riesce a migliorare la precision di oltre lo 1.2% e l'accuracy dell'1.5% rispetto alla configurazione standard, mentre il recall peggiora solo dello 0.5%: osservando la confusion matrix corrispondente si nota come diminuisca il numero di falsi positivi.

5.2 Combinazione degli scores precedenti

Il secondo approccio provato per tentare di migliorare le performances dell'algoritmo può essere considerata una versione "soft" della versione appena presentata: invece di cambiare in modo "hard" la posa corrente sulla base delle tre precedenti, lo score finale di ciascuna posa ad un istante temporale è la combinazione lineare dello score calcolato allo stesso istante e di quello calcolato nell'istante temporale precedente.
In pseudocodice:

```
for each skeleton in the scene:
  for each pose in the gallery:
    *calculate scores for the considered skeleton links
    obtaining scores[t]*
    final_scores[t] = (1 - weight)*scores[t] + weight*scores[t-1]
    scores[t-1] = scores[t]

  *calculate the best pose to return for each skeleton
  on the base of final_scores[t]*
```

Tale criterio per il calcolo degli scores finali viene anche detto multiframe in quanto tiene in considerazione più frame temporali, in quest caso due, per il riconoscimento della posa di ciascuno scheletro. A differenza della politica presentata prima, questo metodo opera a monte di alcune operazioni fatte dall'algoritmo, come ad esempio l'influenza sull'esecuzione prodotta dallo *skeleton_score_fusion_policy*.
L'algoritmo modificato appena descritto è stato testato con diversi valori per la variabile *weight*, da 0.1 a 0.5, tutti riportati nella tabella 3: in particolare la configu-

razione migliore, con $weight = 0.2$, migliora l'accuracy dell'1.6% e il recall dell'1.3% rispetto alla configuazione standard, mentre la precision peggiora dello 0.2%.

5.3 Ampliamento dei links considerati nel riconoscimento della posa

Come riportato nella sezione 4.1, l'algoritmo di pose recognition considera solamente i due avambracci e le due gambe di uno scheletro al fine di riconoscere la posa assunta da una persona nella scena.
Una naturale evoluzione sistema di identificazione è tenere in considerazione anche altri *skeleton links*, come ad esempio le due braccia (anatomicamente identificate come la porzione dell'arto superiore compresa fra la spalla e il gomito).
La figura 5 mostra i link aggiuntivi considerati:

(a) Versione standard dell'algoritmo. (b) Versione modificata dell'algoritmo.

Figura 5: In verde sono riportati i link considerati per il calcolo della posa

Non sono state considerate le porzioni superiori delle gambe o altri links dello scheletro in quanto sono poco caratterizzati: tutte le pose condividono la postura eretta e le gambe in posizione all'incirca parallele; a conferma di ciò, com'è evidenziato nella tabella 2, non c'è alcuna configurazione di parametri in cui l'inclusione delle gambe porta ad un sostanziale miglioramento delle performances.
Nella tabella 3 la configurazione che considera interamente gli arti superiori è riportata con **def_imp** e produce un miglioramento del recall di oltre il 18% e dell'accuracy di quasi il 3% a fronte di un peggioramento della precision di circa il 4%.
E' stato scelto infine di combinare tutti e tre i miglioramenti apportati all'algoritmo standard finora presentati (**def_best**), con risultati significativi: il recall migliora del 18% e l'accuracy del 3%, mentre la precision peggiora di circa l'1.5%.

La tabella ed il grafico seguenti riassumono le varie configurazioni modificate dell'algoritmo standard testate, sono state inoltre evidenziate le più interessanti dal punti di vista delle performances.

configurazione	precision	accuracy	recall	% prec	% acc	% rec
1/top/1.5	0.928	0.940	0.713	-	-	-
def_out	0.939	0.954	0.709	1.2	1.5	-0.5
def_0.1	0.924	0.954	0.721	-0.4	1.5	-1.1
def_0.2	0.926	0.955	0.722	-0.2	1.6	1.3
def_0.3	0.923	0.954	0.721	-0.5	1.5	1.1
def_0.5	0.919	0.954	0.721	-0.9	1.5	1.1
def_imp	0.891	0.964	0.847	-3.9	2.5	18.8
def_best	0.911	0.967	0.845	-1.8	2.8	18.5

Tabella 3: Valori delle metriche considerate per ciascun raffinamento dell'algoritmo, con differenza percentuale rispetto alla configurazione di default. In **grassetto** sono evidenziate le configurazioni più significative.

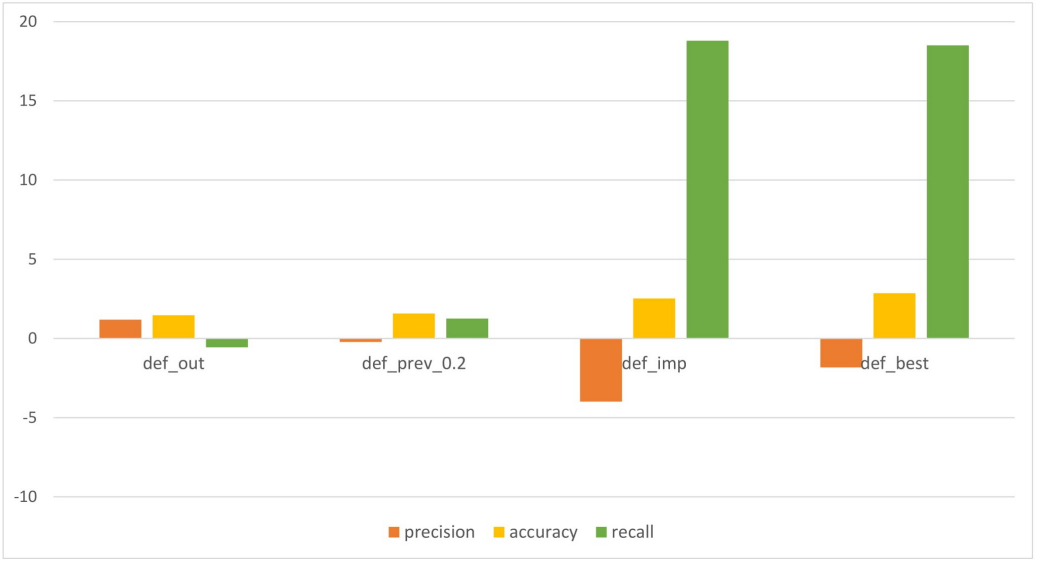


Figura 6: Differenza percentuale delle metriche tra le configurazioni più significative e quella di default.

6 Problemi riscontrati

Ci sono stati dei problemi tecnici a registrare delle nuove pose di riferimento, come ad esempio la scarsa illuminazione del laboratorio, la presenza di altre persone nella scena o la non perfetta calibrazione dei sensori; si è quindi deciso di mantenere le pose presenti di default nella libreria.
Un altro importante problema incontrato è la presenza di un errore nel codice della libreria OpenPTrack che ne precludeva il corretto funzionamento. In particolare quando il parametro per la scelta della politica di fusione degli score dei singoli giunti è posto a 0, ossia viene fatta la media degli score dei singoli giunti per ottenere lo score della posa i-esima, non viene predetta alcuna posa e i valori degli score delle pose hanno valore nan. Dopo molto debugging e reverse engineering è stato trovato e corretto il problema, che risiede nel valore di ritorno errato della lamba function che calcola la media degli score delle pose. Il bug è stato segnalato nell'apposita sezione della repository della libreria [4].
Sono emerse delle difficoltà ad eseguire isolatamente il nodo di pose recognition dal resto della libreria OpenPTrack, ed a compilare i messaggi ROS della libreria; entrambe le difficoltà sono state colmate con l'attenta modifica di vari file CMake e di header.
Non per ultimo ci si è scontrati con la complessità di leggere i messaggi ROS registrati in file di tipo .bag all'interno di un nodo ROS, ed in particolare con la presenza di messaggi innestati e/o combinazione di messaggi appartenenti alla libreria standard ed a librerie esterne (situazione accentuata dalla parziale mancanza di documentazione delle rispettive API c++).

7 Conclusioni

Premesso che sono state assunte pose molto simili a quelle della libreria e che queste ultime non presentavano complicazioni particolari (per esempio pose in cui la persona è seduta o distesa a terra), il funzionamento dell'algoritmo e le considerazioni fatte finora sono largamente dipendenti dalla scelta delle pose standard che l'algoritmo può riconoscere e da quelle effettivamente assunte dalle persone nella sequenza registrata.
Dai risultati ottenuti nelle varie simulazioni effettuate (tabelle 2 e 3), è possibile notare come sia molto difficile aumentare la precision rispetto all'algoritmo in configurazione standard; di fatto l'unica variante che vi riesce è quella ottenuta aggiungendo la rimozione degli outliers, sezione 5.1, che alza la precision di poco più dell'1%: tale risultato apparentemente modesto è motivato dalla già buona precision ottenuta dalla configurazione di default, oltre che alla presenza di numerosi

falsi positivi per alcune pose. Viceversa possiamo notare come sia più semplice incrementare accuracy e recall: quest'ultima in particolare ha un valore decisamente più basso di precision ed accuracy e quindi sono possibili margini di miglioramento più significativi.
In generale quindi risulta difficile incrementare tutte e tre le metriche considerate e ci si trova invece di fronte ad un tradeoff tra precision e accuracy-recall: in base alle necessità sono più appropriate alcune tecniche e/o configurazioni di parametri rispetto ad altre.
Sempre guardando alle due tabelle è possibile osservare come gli affinamenti applicati all'algoritmo, di diversa natura, diano risultati positivi ed incoraggianti; non meno importante però è un'adeguata scelta dei parametri d'esecuzione dell'algoritmo in base al particolare tipo di dataset con cui si lavora, che possono produrre cambiamenti molto significativi.
Tutto il codice scritto per il calcolo delle metriche presentato nella sezione 4.3, oltre alle modifiche apportate all'algoritmo di pose_recognition viste nella sezione 5, è accessibile nella repository https://bitbucket.org/Mandamondo/tesinarobotica [1].

Riferimenti bibliografici

[1] https://bitbucket.org/mandamondo/tesinarobotica.

[2] https://en.wikipedia.org/wiki/type_i_and_type_ii_errors.

[3] https://github.com/openptrack/open_ptrack_v2.

[4] https://github.com/openptrack/open_ptrack_v2/issues/15.

[5] https://support.xbox.com/en-us/xbox-on-windows/accessories/kinect-for-windows-v2-setup.

[6] http://www.dell.com/it/aziende/p/xps-8930-desktop/pd?ref=pd_oc.