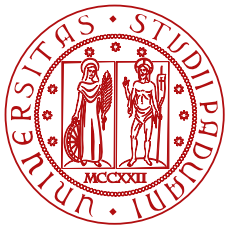


Università degli Studi di Padova

Laurea Magistrale in Ingegneria Meccatronica

# Progettazione di un Framework Multimodale per la Valutazione del Benessere degli Operatori


Studente

**Paride Gnesotto**

**Matricola: 2061971**

Relatore

**Prof. Monica Reggiani**

**Università degli Studi di Padova**

Correlatore

**Dott. Mattia Guidolin**

**Università degli Studi di Padova**



Anno Accademico
2023/2024

**Sommario**

L'emergere dell'Industria 5.0 ha spostato il paradigma industriale dall'essere orientato alle prestazioni a una produzione di beni più antropocentrica, sostenibile ed economicamente resiliente. Uno dei principali obiettivi tecnologici di questo nuovo paradigma industriale è lo sviluppo degli Human Digital Twin (HDT), estensione del concetto di Digital Twin (DT) dell'Industria 4.0: una versione completamente digitale di un sistema o di un componente, in grado di evolversi e di comportarsi allo stesso modo del proprio gemello fisico, e da cui è possibile ricavare dei feedback con cui generare strategie di controllo ed ottenere una migliore integrazione del sistema/componente in un sistema più ampio (o un sistema di sistemi). In questo modo, l'HDT permetterebbe di migliorare il modo in cui gli operatori umani e le macchine integrano i reciproci punti di forza, consentendo una forma di produzione industriale tecnologicamente avanzata e contemporaneamente attenta alle esigenze umane, definita come Human-centric Smart Manufacturing (HSM).

L'obiettivo di questa tesi è la progettazione e lo sviluppo di un framework in grado di trattare dati provenienti da fonti eterogenee e di rappresentarli adeguatamente per la futura implementazione di un HDT volto alla valutazione del benessere umano nell'industria manifatturiera. Viene inoltre eseguita una analisi delle prestazioni del framework attraverso un esperimento in ambiente di laboratorio, simulando un'operazione di assemblaggio in un arco di tempo medio-lungo, e mostrando come la raccolta di dati eterogenei possa funzionare come base per monitorare e fornire un feedback in tempo reale sull'ergonomia e lo stress dell'operatore.

# Indice

# Lista delle Figure

# Lista delle Tabelle



# Lista degli Algoritmi

# Lista degli Acronimi

**AI** Artificial Intelligence

**API** Application Programming Interface

**AR** Augmented Reality

**BLE** Bluetooth Low Energy

**BPE** Body Pose Estimation

**CPS** Cyber-Physical System

**DDS** Data Distribution Service

**DT** Digital Twin

**ECG** Electrocardiography

**EDA** Electro-Dermal Activity

**EEG** Electroencephalography

**EMG** Electromyography

**EOL** End Of Life

**FoI** Field of Illumination

**GATT** Generic Attribute Profile

**GPA** Generalized Procrustes Analysis

**HCS** Human-Cyber System

**HCPS** Human-Cyber-Physical System

**HDT** Human-Digital Twin

**Hi-ROS** Human Interaction in ROS

**HPS** Human-Physical System

**HRC** Human-Robot Collaboration

**HRV** Heart Rate Variability

**HSM** Human-centric Smart Manufacturing

**IMU** Inertial Measurement Unit

**IoT** Internet of Things

**IR** Infrared

**MoCap** Motion Capture

**NFOV** Narrow Field of View

**PMD** Polar Measurement Data

**QoS** Quality of Service

**RGB** Red Green and Blue

**RGB-D** Red Green Blue and Depth

**rcl** ROS client library

**ROS** Robot Operating System

**sdr** sistema di riferimento

**SDK** Software Development Kit

**SIG** Special Interest Group

**STL** Standard Template Library

**SVD** Singular Value Decomposition

**TF** Transform

**tf2** ROS2 transform library

**ToF** Time of Flight

**UUID** Universally Unique Identifier

**WEM** Workforce Ergonomics and Management

**WFOV** Wide Field of View

**XML** eXstensible Markup Language

**YAML** Yet Another Markup Language

# 1 Introduzione

## 1.1 Industria 4.0 ed Industria 5.0

Gli ultimi anni hanno visto la definizione e sviluppo di un nuovo paradigma industriale noto con il nome di Industria 5.0. Esso può essere considerato come la naturale evoluzione del concetto di Industria 4.0, che a sua volta ha rappresentato il principale modello di produzione industriale dell'ultimo decennio [1], [2].

Con Industria 4.0, termine derivante dal tedesco *Industrie 4.0* e presentato alla Fiera di Hannover del 2011, si intende quella serie di modifiche in ambito manifatturiero (e più in generale produttivo) che hanno visto il proprio sviluppo iniziale in Germania nei primi anni 2010 [3]. Questo paradigma pone l'attenzione sull'utilizzo delle più avanzate tecnologie informatiche disponibili, allo scopo di ottimizzare la produttività su tutti i livelli grazie alle potenzialità di calcolo e comunicazione che esse offrono. In tale ottica l'obiettivo dell'Industria 4.0 consiste nel realizzare sistemi produttivi più "intelligenti" (spesso indicati con il termine *smart*), formati da componenti e sottosistemi fortemente interconnessi che possono scambiare in tempo reale grandi quantitativi di dati. Ciò rende possibile la generazione di un gemello digitale (Digital Twin, DT) del sistema o dell'ambiente, grazie al quale si rende possibile un livello di conoscenza, controllo e personalizzazione della produzione industriale senza precedenti [4]. Per questa sua natura spiccatamente tecno-centrica, orientata alla produttività e caratterizzata dalla raccolta ed analisi di grandi quantità di informazioni, tra le parole chiave tipicamente associate all'Industria 4.0 figurano ad esempio: Internet of Things (IoT), Big Data, Data Analytics, AI, Smart Manufacturing, Cyber-Physical System (CPS), Augmented Reality (AR) [3].

Durante il periodo di pieno sviluppo di questo paradigma produttivo, complici i risultati positivi in termini di incremento della produttività, altre tematiche di ricerca si sono aggiunte a quelle nominate in precedenza. Tra esse si trovano ad esempio la sostenibilità della produzione industriale, il ruolo degli operatori umani all'interno di fabbriche sempre più automatizzate e le conseguenze che ciò comporta per la società. L'importanza di questi temi è cresciuta velocemente negli ultimi dieci anni, portandoli ad essere al centro dell'attenzione su molteplici livelli, dalla singola persona fino alle entità statali e sovrastatali [2]. I motivi

dietro a ciò sono molteplici, e spaziano dalla crescente pressione che gli effetti sempre più tangibili del cambiamento climatico applicano sulla popolazione, a spinte di tipo economico e geopolitico come nel caso della ridefinizione dei rapporti commerciali ed industriali con altri Paesi. Infine, la pandemia di COVID-19 può essere interpretata come un evento spartiacque che ha portato a discutere diffusamente di temi come il benessere della persona ed i suoi effetti a lungo termine sull'economia, gli effetti che la società dei consumi ha sull'ambiente, ed il fatto che nel mondo globalizzato le catene di produzione e distribuzione sono di lunghezza tale da essere altamente suscettibili ad eventi di simile portata, con ripercussioni di lunga durata per l'economia di diversi continenti [5]. Per questo, invece di essere proposti come semplici appendici della già matura Industria 4.0, tali argomenti si sono sviluppati come tematiche centrali della sua diretta evoluzione, la cosiddetta Industria 5.0, come si può vedere in Fig. 1.1. Da questo punto di vista la quarta rivoluzione industriale può essere vista come un "abilitatore tecnologico" che ha posto le basi per la realizzazione dei nuovi obiettivi in ambito industriale e sociale che si presuppone verranno perseguiti negli anni a venire. L'importanza di tali obiettivi ha portato inoltre alla definizione della quinta rivoluzione industriale in tempi molto più brevi rispetto alla durata delle precedenti, evitando di attendere che l'Industria 4.0 terminasse il proprio ciclo di vita e con esso la spinta tecnologica che la caratterizza [1].

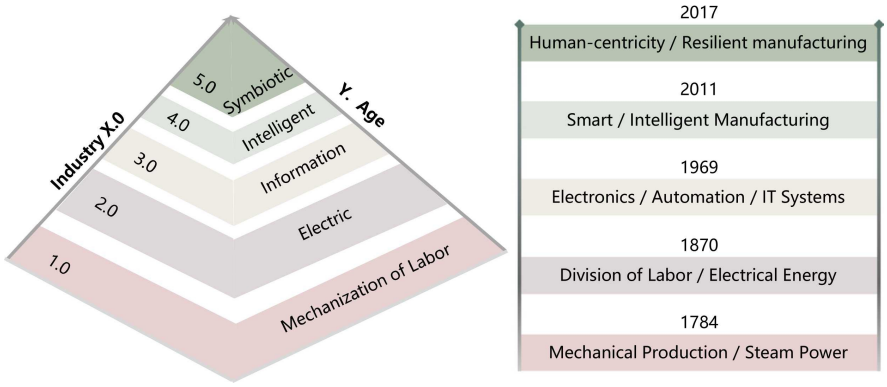



Figura 1.1: Le rivoluzioni industriali e le loro caratteristiche (Fonte: [2]).

Come è possibile osservare in Fig. 1.2, i tre temi che definiscono l'Industria 5.0 sono Antropocentrismo, Resilienza e Sostenibilità. Il primo di essi ha come obiettivo quello di riportare al centro dei sistemi produttivi l'essere umano, non inteso come un annullamento dell'automazione industriale ma come ristrutturazione dell'industria attorno al concetto che l'elemento umano sia parte

integrante ed ineliminabile dei processi produttivi (e quindi non solo un consumatore finale dei prodotti e servizi). Per questo motivo la cura dell'esperienza umana deve essere considerata ad ogni livello delle catene produttive, specialmente dove la carenza di tale cura può comportare le maggiori problematiche dal punto di vista della salute degli operatori. Uno degli ambiti più interessati da questo obiettivo è quello manifatturiero per una serie di motivi, tra cui:

- La necessità di ricorrere ad operatori umani, nonostante l'alto livello di automazione raggiunto dall'industria, per mansioni che richiedono elevati livelli di specializzazione e/o manualità;
- La diffusione di sistemi di robotica collaborativa, la conseguente ricerca di una migliore integrazione dell'operatore negli spazi di lavoro automatizzati, e la necessità di fiducia in tali sistemi per poterne garantire l'accettazione da parte del lavoratore;
- La presenza di operazioni e movimenti potenzialmente nocivi per la salute su periodi di tempo medio-lunghi, a causa della presenza di posizioni disagevoli o di ripetitività delle azioni richieste durante lo svolgimento della mansione, che vanno ad impattare sull'ergonomia del lavoratore e che possono quindi essere causa di problemi per l'apparato muscoloscheletrico.

## 1.2 Human-centric Smart Manufacturing

Una forma di industria manifatturiera che segue i dettami dell'Industria 5.0 viene tipicamente indicata con il termine Human-centric Smart Manufacturing (HSM). Al concetto dell'Industria 4.0 di Smart Manufacturing, ovvero di una produzione altamente informatizzata e perciò dotata di elevati livelli di efficienza, qualità e flessibilità produttive, la HSM aggiunge infatti il requisito antropocentrico a quelli "sistema-centrici" appena elencati [6]. La realizzazione di una HSM passa per una integrazione dell'essere umano molto più profonda di quella ottenuta durante le precedenti rivoluzioni industriali. Questo viene tradotto con il concetto di Human-Cyber-Physical System (HCPS), ovvero un sistema composto da tre sottosistemi interconnessi: lo Human System (l'operatore), il Cyber System (il sistema di controllo) ed il Physical System (i macchinari). Mentre le prime due forme di industrializzazione hanno portato allo sviluppo


Figura 1.2: I temi centrali dell'Industria 5.0 (Fonte: [1]).

degli Human-Physical System (HPS), dove le componenti meccaniche svolgono parte del lavoro manuale al posto del lavoratore, la terza e quarta rivoluzione industriale hanno visto la nascita delle interfacce Cyber-Physical e Human-Cyber con lo sviluppo dei sistemi di controllo che scaricano parte del carico di lavoro mentale dell'operatore. Per implementare gli HCPS richiesti dalla HSM è perciò necessario rafforzare le interfacce tra i tre sistemi, specialmente quella dello Human-Cyber System: il Cyber System deve quindi essere dotato di una conoscenza sufficientemente approfondita anche dell'umano (e non solo del Physical System) per realizzare una collaborazione uomo-macchina (Human-Robot Collaboration, HRC) fruttuosa sia dal punto di vista della condivisione del lavoro manuale che di quello mentale [7].

## 1.3 Human-Digital Twin

Lo strumento individuato come adatto a questo scopo è lo Human-Digital Twin (HDT), estensione al caso di una persona del concetto di DT proprio dell'Industria 4.0 [8]. I DT sono modelli digitali di sistemi fisici, sufficientemente avanzati da poter essere considerati come dei "gemelli digitali" dei sistemi che rappresentano, tramite i quali è quindi possibile eseguirne il controllo ad alto livello grazie alla conoscenza che essi forniscono. Gli HDT sono pertanto una applicazione di questo concetto ad un essere umano, per rappresentarne lo stato

in modo continuativo nel tempo e permettendo di utilizzare tali informazioni sia nell'esecuzione delle strategie di controllo del sistema che per fornire un feedback alla persona stessa. In quest'ottica, gli HDT sono necessari allo sviluppo della HSM prefissato nell'Industria 5.0 [9].

Nel contesto di applicazioni industriali del concetto di HDT, lo stato del lavoratore che si vuole rappresentare ed utilizzare nella gestione delle strategie di controllo dei sistemi collaborativi di produzione è principalmente quello di salute. Questo non solo permette di perseguire il principio di antropocentrismo dell'Industria 5.0 con l'obiettivo di massimizzare il benessere degli operatori, ma anche di limitare gli effetti economici e sociali su diverse scale temporali che le malattie professionali possono causare [10]. Con l'affermarsi dell'Industria 5.0 il concetto di benessere in ambito lavorativo si è esteso da quello più tradizionale e legato all'aspetto fisico della persona, ad uno che tiene conto anche degli aspetti relativi alla salute mentale. Nell'ambito dell'industria manifatturiera, la salute di tipo fisico è tipicamente associata all'ergonomia dell'operatore, che viene minata dall'assunzione di posizioni disagevoli, dal sollevamento di carichi pesanti e dall'esecuzione di azioni ripetitive ad alta frequenza. Per quanto riguarda il benessere mentale dei lavoratori, invece, sono di interesse gli aspetti cognitivi come l'affaticamento e lo stress, che possono essere causati da operazioni che richiedono elevati livelli di concentrazione e/o manualità, oltre che dal semplice sforzo fisico prolungato.

La valutazione dei sopracitati elementi che contribuiscono allo stato di benessere dei lavoratori, o perlomeno a quello di interesse per gli operatori dell'industria manifatturiera, può avvenire secondo una serie di strumenti e metodi. Oltre all'utilizzo di questionari ed osservazioni dirette da parte di esperti in ambito ergonomico e psicologico, che hanno il difetto di richiedere la presenza di personale formato in merito, negli anni si è sviluppato il ricorso a misure dirette tramite appositi sistemi di misura [11]. Tali sistemi si possono suddividere in due categorie, a seconda del tipo di dati che raccolgono:

- Sistemi che misurano la cinematica e/o dinamica del corpo;
- Sistemi che misurano segnali fisiologici.

### 1.3.1 Misure cinematiche e dinamiche

In questa categoria rientrano i sensori di forza ed il Motion Capture (MoCap), ovverosia sistemi di cattura della posa e movimento del corpo. I primi sono tipi-

camente utilizzati per quantificare le forze esterne al corpo umano come i pesi sollevati (ad esempio tramite l'uso di tappetini sensorizzati), ma difficilmente riescono a rappresentare le reazioni interne che coinvolgono le articolazioni. Il secondo gruppo rappresenta invece quello di maggior interesse per la valutazione del benessere della persona, dato che è possibile associare in modo relativamente diretto i dati provenienti dalla stima della posa di una persona (Body Pose Estimation, BPE) agli indici ergonomici già ampiamente sfruttati dagli esperti del settore: è pertanto considerato necessario nell'implementazione di un HDT in ambito industriale per valutare il benessere del lavoratore.

Esistono diverse tecnologie per il MoCap disponibili in commercio:

- Sistemi ottici basati su *marker* apposti sul corpo della persona: rappresentano solitamente il metodo con le migliori performance in termini di accuratezza, ma sono allo stesso tempo quelli che richiedono il setup più complesso ed i costi maggiori, data la necessità di un posizionamento molto accurato dei marker e del ricorso a molteplici telecamere per far fronte ai problemi di occlusione, cosa che li rende poco adatti all'uso in ambiti di tipo industriale;

- Sistemi ottici *markerless* che fanno uso di telecamere a colori (Red Green and Blue, RGB) o con l'aggiunta di sensori di profondità per migliorarne le performance (le cosiddette telecamere Red Green Blue and Depth, RGB-D): non richiedono l'uso di tute o marker appositi, ma sono i sistemi meno accurati e soffrono anche in questo caso di problemi di occlusione;

- Sistemi *non ottici* che fanno uso di sensori inerziali (Inertial Measurement Unit, IMU) da apporre sul corpo: non presentano problemi di occlusione, e sono molto più portatili dei sistemi ottici marker-based pur avendo buona accuratezza, ma forniscono solamente informazioni relative all'orientazione dei sensori e richiedono pertanto l'implementazione di un modello del corpo umano per poterne estrarre la posa, oltre al fatto di soffrire di problemi di deriva della posa nel tempo dovuta all'integrazione delle misure di accelerazione. Tale fenomeno è tipicamente indicato con il termine "drift".

### 1.3.2 Misure di segnali fisiologici

Questo tipo di misurazioni possono dare informazioni legate sia alla salute fisica, come ad esempio il livello di sforzo, sia allo stato di benessere cognitivo,

come ad esempio stress e concentrazione, anche se non sempre è nota una corrispondenza segnale-stato di salute come quella presente tra posa della persona e fattori di rischio per l'apparato muscoloscheletrico.

I segnali che possono essere misurati sono:

- L'elettrocardiogramma (ECG) o, sua versione semplificata, la variazione della frequenza di battito cardiaco (Heart Rate Variability, HRV): sono un classico indicatore dello stato di attività fisica, ma anche di stati come agitazione, eccitazione, paura;
- L'elettroencefalogramma (EEG): raccoglie diverse frequenze di segnali elettrici in categorie che sono associate tipicamente a diversi stati come concentrazione e riposo o a diverse zone del cervello, ma richiede l'uso di apparecchiatura costosa e tipicamente scomoda da indossare, oltre ad essere influenzata dal rumore elettrico;
- L'elettromiografia (EMG): misura l'attività elettrica dei fasci muscolari per valutarne l'attivazione, ma richiede posizionamenti molto accurati degli elettrodi e nonostante gli sforzi per miniaturizzarne la tecnologia resta comunque poco adatta ad ambiti industriali a causa della sensibilità estrema al rumore ed alla limitata correlazione con lo stato di salute generale della persona;
- L'attività elettrodermica (Electro-Dermal Activity, EDA): è nota con varie nomenclature, ma in generale misura la resistenza o conduttanza della pelle, grandezze chiaramente affette dal livello di sudorazione e pertanto di interesse per rappresentare lo stato di attività fisica, affaticamento o stress della persona.

## 1.4 Obiettivo della tesi

Questo lavoro di tesi si svolge all'interno del contesto relativo alla realizzazione di un HDT per applicazioni industriali, volto alla rappresentazione dello stato di benessere degli operatori dell'industria manifatturiera. Il Dipartimento di Tecnica e Gestione dei Sistemi Industriali è infatti attivo in questo campo di ricerca, avendo già mosso i primi passi verso la realizzazione di tale obiettivo, con lo sviluppo della piattaforma WEM (Workforce Ergonomics and Management) per la valutazione online di molteplici indici ergonomici a partire dalla stima della posa della persona ottenuta con sistemi di MoCap [12], [13].

L'obiettivo della tesi è quindi quello di sviluppare un framework che permetta la raccolta e presentazione di dati utili ad un HDT di futura realizzazione. Il framework che si vuole sviluppare è inoltre definito con l'aggettivo *multimodale*, in quanto la volontà di estendere la valutazione dello stato di salute dell'operatore contemplando anche una stima del benessere cognitivo da parte dell'HDT rende necessaria la raccolta di dati provenienti da sistemi di sensori molto diversi fra loro. Nello sviluppo del framework sono stati tenuti in considerazione i requisiti che l'applicazione in contesti industriali richiede, come ad esempio le performance in termini di tempo di risposta da garantire affinché esso possa essere integrato all'interno del sistema di controllo di una postazione di lavoro collaborativa avanzata. Una tale postazione sarebbe capace di adattarsi ad operatori con necessità differenti e di reagire alle variazioni nel loro stato di salute durante lo svolgimento delle attività, facilitando l'accessibilità ed inclusività dei luoghi di lavoro del futuro e migliorando il livello di benessere psicofisico dei lavoratori.

La tesi è così strutturata: nel Capitolo 2 vengono descritti i requisiti del framework e le scelte progettuali fatte per soddisfarli, nel Capitolo 3 viene discussa l'implementazione del principale blocco che compone il framework e permette di integrare diverse tecnologie di MoCap allineando i dati delle pose stimate da tali metodi. Successivamente, nel Capitolo 4 si tratta la realizzazione del software che permette di raccogliere dati dal sistema di ECG e da quello di EDA. Infine, nel Capitolo 5 si descrivono gli esperimenti svolti per la valutazione del framework e si procede ad analizzarne i risultati, mentre il Capitolo 6 conclude il lavoro con una discussione dell'intero progetto, i risultati ottenuti ed i possibili sviluppi futuri.

# 2 Design del Framework

Il framework che si è sviluppato ha come obiettivo la raccolta dati relativi allo stato di benessere degli operatori nell'industria di prodotto, da utilizzare come punto di partenza per la futura implementazione di un HDT capace di stimare tale stato di benessere e fornire le relative informazioni al lavoratore ed al sistema di controllo della postazione collaborativa avanzata che lo integrerebbe. Pertanto, sono stati identificati i seguenti requisiti:

- Integrazione di sensori di tipologia differente per rappresentare diverse grandezze fisiologiche;
- Performance compatibili con l'utilizzo online per poter garantire una valutazione continuativa dello stato di salute;
- Utilizzo di sensoristica disponibile commercialmente e per la quale siano forniti SDK ed API che permettono di gestirne la comunicazione.

Dal primo requisito, unito alle intenzioni di sviluppo futuro del framework, è stato deciso di ricorrere a ROS (Robot Operating System) in quanto pensato proprio per la realizzazione di sistemi distribuiti che integrano sensori di vario tipo [14]. Considerato il secondo requisito, invece, è stato scelto di implementare il codice sfruttando il linguaggio di programmazione C++, che permette ottime performance dal punto di vista del tempo di calcolo e la gestione della memoria, come spiegato più nel dettaglio in sez. 2.2. Infine, la scelta della sensoristica da implementare è ricaduta sulla combinazione di due tipi di MoCap, inerziale e markerless, visto che la stima della posa è considerata di fondamentale importanza per la valutazione del benessere dato il suo utilizzo nelle valutazioni ergonomiche. Per quanto riguarda le misure di segnali fisiologici si è invece scelto di implementare la misurazione ECG e quella EDA. La prima scelta è dovuta al fatto che la conoscenza del tracciato elettrocardiografico permette di ottenere una informazione molto più dettagliata sullo stato di benessere e salute di una persona rispetto al solo battito cardiaco, mentre per quanto riguarda la seconda si è considerato che conoscere il tasso di sudorazione potrebbe essere una alternativa molto più semplice da realizzare in ambienti industriali rispetto alle misure EMG, che sono per propria natura estremamente sensibili alle

interferenze elettromagnetiche oltre che rappresentare informazioni di difficile lettura ed estremamente localizzate sul corpo del soggetto.

## 2.1 Robot Operating System

Con Robot Operating System (ROS) si intende un ecosistema multipiattaforma ed open-source (distribuito sotto licenza Apache 2.0) di librerie e strumenti software orientato allo sviluppo di applicazioni per sistemi robotici [14]. Grazie alla propria natura di software libero, ROS ha sviluppato negli anni una folta comunità di utenti ed una vasta gamma di pacchetti e librerie che lo hanno reso lo standard *de facto* per lo sviluppo di applicazioni nel campo della robotica, oltre a generare un interesse crescente anche per l'utilizzo in ambito industriale [15]. Questo ha portato allo sviluppo della seconda generazione di ROS, detta ROS 2, sviluppata con l'intenzione di superare i limiti della precedente in ambito di sicurezza, affidabilità su canali non ideali, utilizzo in sistemi embedded e sistemi distribuiti multi-robot [16].

È stato scelto di utilizzare ROS 2 per la realizzazione del progetto perché esso presenta numerosi vantaggi e migliorie rispetto alla prima generazione (come spiegato in 2.1.5), ma soprattutto in quanto ROS 1 si trova ufficialmente nella fase finale del proprio ciclo di vita (End Of Life, EOL) e si è preferito sviluppare il framework utilizzando il sistema che permettesse il supporto più longevo possibile.

### 2.1.1 Nodi e Messaggi

Gli elementi fondamentali in ROS sono detti *nodi*: essi sono pensati per svolgere un singolo obiettivo modulare, e realizzano il calcolo distribuito lavorando in parallelo ad altri nodi con cui scambiano *messaggi*. Tali messaggi possono essere di tipo standard, forniti dalle librerie di ROS, oppure definiti dall'utente tramite *interfacce* personalizzate che passano per la creazione di un file testuale di tipo `.msg` in cui si utilizza una struttura del tipo:

```
nome_campo  tipo_dato
```

Un messaggio può contenere molteplici campi, a loro volta di tipo standard o definito in altre interfacce create dall'utente. Questa struttura dati è concettualmente analoga ad una `struct` nel linguaggio C++, consentendo di raggruppare variabili ed informazioni di tipo diverso all'interno di uno stesso oggetto. La dif-

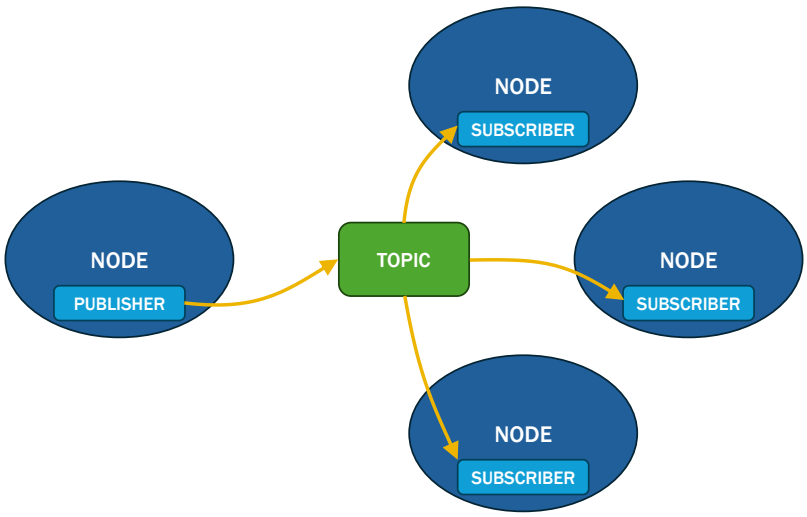



Figura 2.1: Rappresentazione dei topic.

ferenza principale sta nel fatto che i messaggi ROS sono pensati per implementare la comunicazione all'interno di sistemi distribuiti e per questo motivo sono definiti in modo tale da essere indipendenti dal linguaggio di programmazione usato e dal sistema operativo.

### 2.1.2 Topic, Service e Action

I nodi possono comunicare in tre modi diversi. Con i *topic* è possibile definire una comunicazione di tipo uno-a-molti in cui un nodo detto *publisher* può inviare messaggi di un tipo specifico a tutti i nodi che hanno effettuato una sottoscrizione al topic, motivo per cui vengono detti nodi *subscriber* (Fig. 2.1).

Il secondo metodo è quello dei *service*, che fa uso di una comunicazione di tipo *server-client* in cui un nodo client invia delle *richieste* al nodo server, il quale successivamente invia un messaggio di *risposta* al nodo che ha fatto richiesta (Fig. 2.2). I messaggi scambiati tra client e server sono definiti tramite un'interfaccia creata a partire da un file di tipo `.srv`, avente una struttura che richiama quella vista per i messaggi semplici:

```
campi_richiesta     tipi_richiesta
---
campi_risposta      tipi_risposta
```

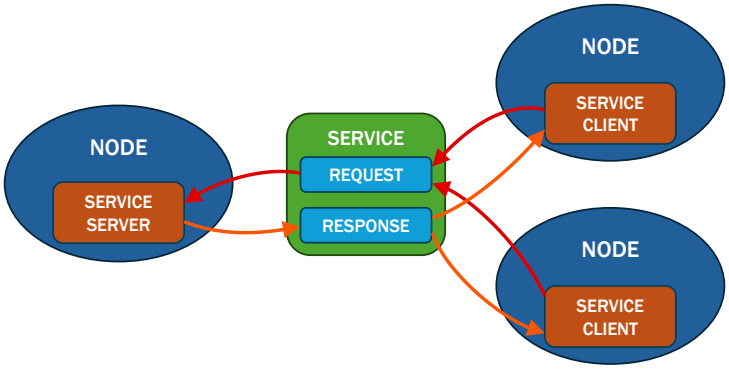



Figura 2.2: Rappresentazione dei service.

Infine si hanno le *action*, che sono una forma di comunicazione costruita a partire dalle due precedentemente descritte (Fig. 2.3). Le action possono essere considerate come dei service in cui il nodo server può fornire del feedback continuo sullo stato del calcolo al nodo client tramite un topic apposito, ed il nodo client può interrompere l'esecuzione dell'azione. L'interfaccia di una action è definita in un file di tipo `.action` strutturato come segue:

```
campi_richiesta     tipi_richiesta
---
campi_risultato     tipi_risultato
---
campi_feedback      tipi_feedback
```

### 2.1.3 Workspace e Package

ROS è pensato per la creazione di applicazioni di qualsiasi tipo, pertanto la modularità è di fondamentale importanza per la gestione dei progetti. Innanzitutto è possibile definire dei *workspace*, directory appositamente strutturate e pensate per contenere al proprio interno tutte e sole le funzionalità relative ad un determinato progetto.

La tipica struttura di un workspace è la seguente:

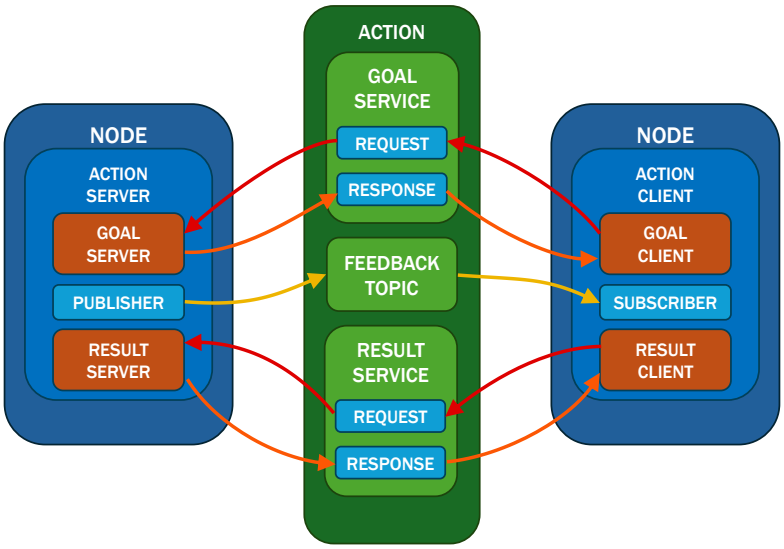



Figura 2.3: Rappresentazione delle action.

```
./ (root del workspace)
|-- build/
|-- install/
|-- log/
|-- src/
```

La directory `src/` contiene il codice sorgente che l'utente vuole compilare ed eseguire, mentre le directory `build/`, `install/`, e `log/` sono generate dal compilatore e contengono rispettivamente i file intermedi necessari, l'installazione degli eseguibili, e file di log relativi al processo di compilazione.

Tali workspace funzionano con un sistema di *overlay* che permette di non disturbare l'installazione base di ROS con codice pensato per delle specifiche applicazioni: è pertanto necessario eseguire un `source` dell'installazione di ROS per poterne utilizzare le funzionalità base (come ad esempio la compilazione del codice, tramite il programma `colcon`[1]), ed in seguito anche un source dell'*environment* del workspace che si vuole utilizzare una volta che quest'ulti-

[1]Per ulteriori informazioni in merito al funzionamento di `colcon`, è possibile visitare: https://docs.ros.org/en/humble/Tutorials/Beginner-Client-Libraries/Colcon-Tutorial.html.

mo è stato compilato. Di seguito un esempio di sourcing dell'installazione e del workspace in ambiente Linux:

```
# Source of the ROS 2 installation
source /opt/ros/<version>/setup.bash
# Source of the workspace environment, must be performed in the root
    of the workspace
source install/local_setup.bash
```

Le funzionalità del workspace precedentemente accennate sono definite tramite i *package*, directory che vengono poste all'interno di `src/`. Essi possono contenere codice che definisce nodi, interfacce customizzate e/o il codice che gestisce il lancio dei programmi. I package scritti in C++ vengono compilati tramite CMake, ed hanno la seguente struttura:

```
package_name/
|-- CMakeLists.txt
|-- include/package_name/
|-- launch/
|-- msg/
|-- package.xml
|-- src/
|-- srv/
```

I file `CMakeLists.txt` e `package.xml` sono file di configurazione, il primo contiene informazioni relative alla compilazione del pacchetto mentre il secondo i metadati che lo descrivono. Nella directory `include/` vengono inseriti gli header delle librerie scritte dall'utente, in `launch/` i file di avvio, in `msg/` e `srv/` le interfacce user-defined ed in `src/` il codice sorgente del pacchetto.

### 2.1.4 Launch file

ROS permette di definire dei *launch file* che contengono le informazioni necessarie all'inizializzazione di molteplici nodi, anche appartenenti a package diversi, indicando per ciascuno eventuali parametri. Questa funzionalità risulta estremamente importante per poter avviare una intera rete di nodi ROS con un singolo comando, semplificando molto la gestione di sistemi complessi. I launch file possono essere scritti in Python, XML o YAML, sebbene il primo linguaggio sia quello più utilizzato a questo scopo, in quanto permette di scrivere in modo agevole i launch file relativi ad infrastrutture di nodi ROS molto complesse e

dinamiche grazie alle maggiori astrazione e libertà fornite dalla sua natura di linguaggio di programmazione ad alto livello.

### 2.1.5 Vantaggi di ROS 2

Come accennato all'inizio di questo capitolo, ROS 2 è stato sviluppato per migliorare alcuni aspetti della prima generazione nell'ottica dello sviluppo di software distribuito. Le principali differenze sono:

- Il ricorso al middleware standard DDS come protocollo di rete;
- Il passaggio ad una architettura distribuita che elimina una situazione di Single Point Of Failure nella gestione della rete di cui i processi fanno parte;
- L'utilizzo di una libreria-base scritta in C (ROS client library, rcl), alla quale si interfacciano le librerie in C++ (rclcpp) e Python (rclpy), garantendo quindi le stesse funzionalità indipendentemente dal linguaggio scelto;
- La compatibilità con i più diffusi sistemi operativi, non solo Linux;
- La possibilità di gestire i parametri di qualità richiesti per la comunicazione (Quality of Service, QoS);
- La possibilità di gestire processi in parallelo in modo migliore, grazie ai cambiamenti in termini di gestione dell'esecuzione e di risposta alle callback;
- Migliori performance e compatibilità per applicazioni real-time.

Sebbene la motivazione principale per la scelta della seconda generazione di ROS per l'implementazione del framework sia quella della fine del supporto per ROS 1, queste migliorie rappresentano degli ulteriori punti a favore di ROS 2 e vanno a rinforzare ulteriormente la decisione.

## 2.2 Scelta del linguaggio di programmazione

I pacchetti ROS possono essere scritti sia in C++ che in Python. Nonostante la libreria di base su cui si appoggiano tutte le funzionalità di ROS sia scritta in C, garantendo quindi performance equiparabili per il codice di entrambi i linguaggi, c'è da considerare che i pacchetti che sono stati scritti come parte del lavoro di tesi possono eseguire anche altri calcoli oltre alle chiamate che gestiscono la

comunicazione ed i nodi ROS. Per questo motivo, assieme ai requisiti richiesti dall'applicazione industriale a cui è orientato il framework, è stato necessario scegliere il linguaggio di programmazione che garantisse le migliori prestazioni.

Zehra *et al.* [17] riassumono le principali differenze tra i due linguaggi di programmazione in questione, mostrando come la scelta migliore dal punto di vista delle prestazioni sia C++ sia per quanto riguarda il tempo di calcolo che per l'uso della memoria. Questo è dovuto a vari fattori, tra cui:

- *Tipologia di linguaggio*: C++ è un linguaggio compilato, mentre Python è interpretato. Ciò significa che nel primo caso il codice può essere ottimizzato dal compilatore per ottenere degli eseguibili più efficienti, invece nel caso di Python è necessario attendere uno step aggiuntivo da parte dell'interprete prima di poter ottenere il codice oggetto;

- *Tipizzazione dei dati*: C++ è un linguaggio staticamente tipizzato, in cui quindi il tipo dei dati è definito in tempo di compilazione ed è fisso durante l'esecuzione, permettendo una gestione più ottimizzata della memoria. Python è invece un linguaggio dinamicamente tipizzato, in cui il tipo di un dato può essere definito durante l'esecuzione: ciò significa che sono necessari controlli di compatibilità dei dati da parte dell'interprete per ogni operazione;

- *Gestione della memoria*: Python sfrutta una gestione automatica della memoria, mentre C++ richiede all'utente la gestione della stessa tramite apposite funzioni. Per questo motivo Python fa uso di un *garbage collector* che controlla in run-time quali aree di memoria è possibile liberare, operazione che ovviamente richiede del tempo;

- *Paradigmi di programmazione*: Le librerie standard di C++ (raccolte nella Standard Template Library, STL) ricorrono alla programmazione generica per ottenere un codice efficiente ed altamente modulare grazie all'uso dei template, spostando buona parte dei calcoli in tempo di compilazione.

Per i sopracitati motivi è stato scelto di scrivere il codice sorgente in C++. Tuttavia, date le maggiori potenzialità di Python rispetto alle alternative basate su linguaggi di markup, cosa che permette di definire reti di nodi ROS più avanzate, è stato scelto di utilizzare quest'ultimo per la scrittura dei launch file. Tale scelta non influenza le performance in quanto i launch file sono eseguiti solo una volta durante la fase di inizializzazione della rete di nodi ROS.

## 2.3 Sensoristica

Il lavoro di questa tesi ha l'obiettivo di raccogliere dati per una valutazione del benessere cognitivo oltre che fisico della persona impegnata in attività industriali. Per questo motivo sono stati scelti sensori che ricadono nelle due categorie di *stima della posa* e *segnali fisiologici*.

### 2.3.1 Stima della posa

(a) Microsoft Azure Kinect (Fonte: [18]). (b) Xsens MVN Awinda (Fonte: [19]).

Figura 2.4: I sistemi di sensori utilizzati per il MoCap in questo progetto di tesi.

La stima della posa della persona è una misura necessaria per la valutazione del benessere da un punto di vista più tradizionalmente ergonomico. Come discusso in precedenza, esistono diverse tecnologie per svolgere tale compito, ciascuna con i propri pro e contro. Date le caratteristiche complementari dei sistemi di MoCap markerless basati sull'utilizzo di telecamere RGB-D e di quelli che invece fanno uso di un set di sensori inerziali, è stato deciso di utilizzare entrambe le tecnologie nel framework con l'intenzione di superarne le reci-

proche limitazioni. Tali limitazioni, si ricorda, sono la possibilità di occlusioni ed una minore accuratezza per i sistemi markerless e l'accumulo di errori di posizionamento assoluto per i sistemi inerziali.

Per quanto riguarda il MoCap markerless, sono state utilizzate le videocamere RGB-D Azure Kinect (Microsoft Corporation, Redmond - Washington, USA, Fig. 2.4a), già disponibili presso i laboratori del Dipartimento e per le quali è fornito un driver ROS dalla Microsoft Corporation[2]. Il driver in questione permette di configurare il funzionamento della videocamera tramite i launch file di ROS e di pubblicare tramite appositi messaggi vari flussi di dati in output, tra cui le immagini RGB, i dati di profondità, le point cloud e, di fondamentale importanza, i dati provenienti dall'algoritmo di body tracking. Questi ultimi vengono ricevuti sotto forma di messaggi del tipo `visualization_msgs::MarkerArray`, contenenti in sequenza le posizioni dei landmark che descrivono la posa della persona. Un apposito pacchetto ROS[3] volto alla rappresentazione dei dati di MoCap in un formato adeguato a visualizzare e gestire le pose per la valutazione degli indici ergonomici è stato sviluppato in passato dal gruppo di ricerca del Dipartimento, esso verrà descritto più nel dettaglio nel Cap. 3. La Microsoft Azure Kinect presenta le seguenti specifiche tecniche:

- 1 sensore di profondità ToF (Time of Flight) IR da 1 Mpixel, ovvero in grado di misurare la profondità della scena utilizzando un emettitore infrarosso e misurando il tempo necessario al segnale di luminoso per ritornare al sensore. Essa è capace di operare in modalità NFOV unbinned[4] con risoluzione di $640 \times 576$ pixel a 30 fps in un range 0.5 m–3.86 m, in modalità NFOV 2x2 binned ad una risoluzione di $320 \times 288$ pixel a 30 fps in un range 0.5 m–5.46 m, in modalità WFOV unbinned ad una risoluzione di $1024 \times 1024$ pixel a 15 fps in un range 0.5 m–2.21 m ed in modalità WFOV 2x2 binned ad una risoluzione di $512 \times 512$ pixel a 30 fps in un range 0.5 m–2.88 m. Il FoI nelle modalità NFOV è di $75° \times 65°$, mentre per le modalità WFOV è di $120° \times 120°$.

- 1 sensore RGB da 12 Mpixel capace di produrre video in formato 16:9 ad

[2]Disponibile presso: https://github.com/microsoft/Azure_Kinect_ROS_Driver.
[3]Disponibile presso: https://github.com/HiROS-unipd/skeleton_msgs.
[4]Con *2x2 binning* si intende la tecnica di combinazione del segnale proveniente da 4 pixel adiacenti a formare un singolo pixel equivalente, utilizzata nelle telecamere digitali per aumentarne la sensibilità alla luce e ridurre il rumore a discapito della risoluzione assoluta.

una risoluzione di $3840 \times 2160$ pixel fino a 30 fps e video in formato 4:3 ad una risoluzione di $4096 \times 3072$ pixel fino a 15 fps.

- 1 sensore IMU che include sia un accelerometro che un giroscopio campionati ad una frequenza di 1.6 kHz (ma pubblicati ad una frequenza di 208 Hz).
- 7 microfoni a formare un array circolare.

Il MoCap inerziale è invece ottenuto tramite tute MVN Awinda (Xsens, Enschede - Paesi Bassi), che utilizzano 17 IMU indossate come visibile in Fig. 2.4b per fornire una stima della posa della persona. Anche in questo caso l'azienda ha reso disponibile un'SDK per la lettura dei dati prodotti dal sistema[5], come pure sono stati precedentemente sviluppati dei pacchetti di driver che permettono di esporre in real-time i dati provenienti dalla tuta in appositi topic e convertirli in messaggi ROS adeguatamente strutturati. Le specifiche tecniche della tuta MVN Awinda sono:

- 17 IMU per il corpo + 1 prop sensor per il MoCap di un oggetto. Accuratezza della posizione angolare in condizioni statiche di ±0.5° per gli angoli di rollio e beccheggio di ±1° per l'angolo di imbardata, accuratezza in condizioni dinamiche di ±0.75° per rollio e beccheggio, ±1.5° per l'imbardata. Fondo scala di ±2000 °/s per il giroscopio, ±160 $m/s^2$ per l'accelerometro, ±1.9 G per il magnetometro. Durata della batteria di 6 h di uso continuativo.
- 1 stazione wireless per la ricarica e la comunicazione wireless con le IMU. Garantisce un range di 20 m in ambienti interni e di 50 m all'esterno. La frequenza di aggiornamento dei dati dipende dal numero di sensori connessi ed è pari a 120 Hz fino a 5 sensori, 100 Hz fino a 9 sensori e di 60 Hz fino a 20 sensori.
- Licenza per il software MVN Analyze, che permette di utilizzare il prodotto, calibrare la tuta sulle dimensioni dell'operatore e gestire lo stream e raccolta dei dati del MoCap e di tutti i sensori delle IMU.
- Tute e fasce con velcro per indossare i sensori.

[5]Disponibile presso: https://www.movella.com/support/software-documentation.


Figura 2.5: Sensore Polar H10 per la misurazione dell'ECG (Fonte: [20]).

### 2.3.2 Segnali fisiologici

Per poter svolgere una valutazione dello stato cognitivo dell'operatore sono necessari dati in grado di rappresentare gli effetti fisici di stati mentali come l'affaticamento e lo stress. Una tipica soluzione è quella di eseguire una misura del battito cardiaco o del tracciato elettrocardiografico della persona, per la quale si è scelto il sensore di frequenza cardiaca Polar H10 (Polar Electro Oy, Kempele - Finlandia, Fig. 2.5). L'azienda fornisce strumenti per lo sviluppo di applicativi presso il sito https://www.polar.com/en/developers/. Le specifiche tecniche sono le seguenti:

- Sensore ECG con risoluzione espressa in µV e frequenza di campionamento di 130 Hz, possibilità di esprimere il battito cardiaco ed intervalli RR con tempo di campionamento di 1 s.
- accelerometro con frequenza di campionamento di 25 Hz, 50 Hz, 100 Hz e 200 Hz.
- Durata batteria 400 h.
- Fascia toracica con elettrodi per la misura del battito cardiaco.

Un'altra misurazione che è possibile considerare per una valutazione dello stress è la sudorazione, per cui è stato scelto di utilizzare il sensore di attività elettro-dermica (EDA) biosignalsplux (Plus Biosignals, Lisbona - Portogal-

(a) Hub a 4 canali (Fonte: [21]).

(b) Sensore EDA per biosignalsplux (Fonte: [22]).

Figura 2.6: Sistema Plux biosignalsplux utilizzato per la misura dell'EDA.

lo, Fig. 2.6). Le API per lo sviluppo di software sono disponibili presso il sito https://support.pluxbiosignals.com/knowledge-base/official-plux-application-programming-interfaces-apis/. Le specifiche tecniche del sistema sono:

- 1 hub a 4 canali biosignalsplux, 4 canali dati e 1 canale per il riferimento comune. Risoluzione a 8 bit 16 bit per canale. Frequenza di campionamento fino a 3 kHz con quattro canali attivi, 4 kHz con tre canali in uso, 5 kHz con due canali e 8 kHz con un solo canale attivo. Comunicazione Bluetooth con range di 10 m. Batteria ricaricabile da 700 mA h che permette 10 h di streaming o 24 h di registrazione nella memoria interna opzionale da 16 GB.
- 4 sensori EDA che misurano la conduttanza della pelle in un range di 0 µSv–20 µSv, banda passante di 0–3 Hz
- 1 sensore ECG per la misura del tracciato con range di ±1.47 mV e banda passante di 25–100 Hz.

Oltre che per la presenza di documentazione e strumenti di sviluppo per la lettura dei dati direttamente dai dispositivi, sono stati scelti i sensori Polar H10 e Plux biosignalsplux per la realizzazione delle misure ECG ed EDA anche per il livello di miniaturizzazione raggiunto senza sacrificare le performance. L'analisi dei dati che si possono ottenere da questi sistemi ed il loro significato

nella rappresentazione dello stato di salute della persona richiedono il parere di esperti nel campo della medicina. L'obiettivo di questo progetto in merito a tali sensori è stato pertanto quello di valutare la fattibilità di una implementazione degli stessi in un sistema capace di leggere e ripubblicare in tempo reale i dati relativi alle rispettive misure anche in ambienti come gli impianti industriali.

# 3 Stima multimodale della posa

Per poter sviluppare un sistema in grado di fornire ad un HDT tutte le informazioni utili alla valutazione e rappresentazione dello stato di salute di un lavoratore è sicuramente necessario includere la stima della posa, dato l'utilizzo che se ne può fare nelle valutazioni di carattere ergonomico. Esse sono infatti fondamentali per conoscere lo stato dell'apparato muscoloscheletrico, che rappresenta dal punto di vista statistico una delle principali criticità per quanto riguarda l'incidenza delle malattie professionali.

Il primo passo nell'implementazione del framework è perciò quello di allineare i dati provenienti dalle due forme di stima della posa della persona, il MoCap inerziale e quello markerless, per poter correggere le problematiche di drift che affliggono la prima tecnologia utilizzando le informazioni fornite dalla seconda. Per questo motivo come parte del lavoro di tesi è stato sviluppato il pacchetto ROS chiamato `skeleton_aligner`. Tale pacchetto definisce un nodo di classe `Aligner` sviluppato allo scopo di realizzare l'allineamento richiesto. Per poterne spiegare il funzionamento è necessario prima di tutto trattare lo strumento matematico alla base: le matrici di trasformazione in coordinate omogenee.

## 3.1 Matrici di Trasformazione

La rappresentazione in coordinate omogenee di un punto dello spazio avente coordinate cartesiane $(x, y, z)$ è:

$$P = (x\ y\ z\ 1)^\top \tag{3.1}$$

Tale sistema di coordinate permette di rappresentare una qualsiasi trasformazione affine[1] nello spazio tridimensionale come un semplice prodotto matriciale del tipo:

$$P' = \mathbf{T}P \tag{3.2}$$

[1]Una composizione di una trasformazione lineare e di una traslazione, che in generale rispetta il parallelismo tra rette.

Dove $P$ e $P'$ rappresentano rispettivamente le coordinate omogenee del punto di partenza e quelle del punto di arrivo della trasformazione, mentre $\mathbf{T}$ è la matrice $4 \times 4$ che descrive la trasformazione. Nel caso di una rototraslazione, la matrice assume la seguente forma:

$$\mathbf{T} = \left[\begin{array}{ccc|c} & \mathbf{R_O} & & \begin{matrix} x_0 \\ y_0 \\ z_0 \end{matrix} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{3.3}$$

Dove $\mathbf{R_O}$ è la matrice ortogonale speciale di dimensione $3 \times 3$ che rappresenta la rotazione attorno ad un asse passante per l'origine del sdr, mentre $\mathbf{v} = (x_0\ y_0\ z_0)^\top$ è il vettore che descrive la traslazione.

Tale formula, oltre a rappresentare una isometria diretta[2] nello spazio che manda il punto $P$ nel punto $P'$, può anche essere interpretata come il cambio di sistema di riferimento[3] che esprime lo stesso punto $P$ da un sdr $\mathcal{R}$ all'sdr $\mathcal{R}'$ che si ottiene applicando ad $\mathcal{R}$ la trasformazione inversa $\mathbf{T}^{-1}$, dove:

$$\mathbf{T^{-1}} = \begin{bmatrix} \mathbf{R_O}^\top & -\mathbf{R_O}^\top \mathbf{v} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{3.4}$$

Ovvero, le coordinate degli assi e dell'origine di $\mathcal{R}'$ (espresse nel primo sdr) si ottengono a partire da quelle di $\mathcal{R}$ semplicemente ruotandone gli assi secondo la matrice $\mathbf{R_O}^\top$ e traslandone l'origine di $-\mathbf{R_O}^\top \mathbf{v}$.

Questa doppia interpretazione permette di ottenere la trasformata che lega gli sdr in cui sono espressi i dati dei due sistemi di MoCap, andando semplicemente a calcolare tramite l'Analisi di Procuste [23] (descritta in sez. 3.4) la trasformata che allinea le pose ottenute dai due metodi. Le due interpretazioni sono matematicamente equivalenti, ma per l'applicazione del metodo di Procuste è più intuitivo ragionare come se si stesse cercando di allineare due oggetti dalla stessa forma con diversi posizionamenti ed orientazioni nello spazio, mentre l'obiettivo reale è quello di risalire alla matrice di trasformazione che indica quanto l'sdr che subisce il drift si sposta relativamente a quello che resta fisso, dato che le pose stimate dalle due tipologie di MoCap rappresentano la stessa

[2] Un movimento rigido.

[3] Si ricorda che un sdr nello spazio è definito da una base di tre vettori ortonormali ed un punto di origine.

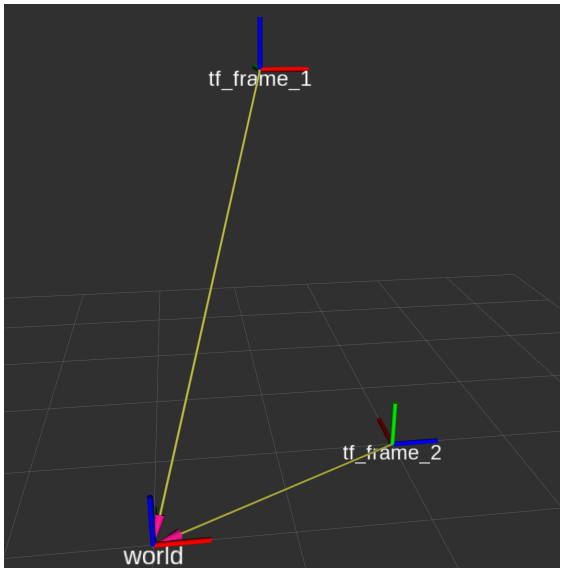



Figura 3.1: Esempio di due sdr definiti rispetto al sistema di riferimento globale tramite TF (indicate dalle frecce in giallo).

persona. Questo secondo punto di vista permette infatti di avere una comprensione più immediata di come funzionano i tool forniti da ROS per visualizzare i dati in diversi sdr.

## 3.2 Trasformate in ROS

Tra le librerie standard di ROS figura la *transform library* (nota come `tf2`), pensata per rappresentare e trattare le matrici di trasformazione in coordinate omogenee (vedi Fig. 3.1). Esse sono infatti ampiamente utilizzate in robotica vista la necessità di passare dalle coordinate dell'end effector espresse nello spazio cartesiano a quelle di posizionamento degli attuatori espresse nello spazio dei giunti, operazione che richiede la ricostruzione della catena cinematica combinando diversi cambi di sistema di riferimento. Tramite tali matrici è possibile realizzare una qualsiasi sequenza di cambi di sistema di riferimento tramite una semplice concatenazione di moltiplicazioni matriciali, e rappresentano quindi un utilissimo strumento per i calcoli di cinematica diretta ed inversa. Adottando quindi la stessa nomenclatura usata in `tf2`, ci si riferirà spesso alle matrici di trasformazione con il termine TF.

## 3.3 Hi-ROS

Il progetto di tesi si appoggia inoltre ad una serie di pacchetti ROS definiti all'interno del progetto Hi-ROS[4] (Human Interaction in ROS), che permettono la gestione della stima della posa di una o più persone sfruttando una interfaccia definita appositamente [24]. Tale interfaccia fa parte del pacchetto `skeleton_msgs` e definisce i seguenti messaggi:

- `KinematicState`, rappresentazione cinematica di un punto ottenuta combinando tre tipi di messaggi standard forniti nel pacchetto `geometry_msgs`. Consiste in un campo di tipo `Pose` che rappresenta la posizione e/o orientazione (tramite quaternione) del punto nello spazio, un campo di tipo `Twist` che rappresenta le velocità lineari ed angolari, ed un campo di tipo `Accel` che rappresenta le accelerazioni;

- `Marker`, rappresenta uno specifico punto di interesse nella posa della persona, tipicamente una articolazione. Contiene un campo `KinematicState` che ne indica la posizione, un campo `float64` per indicare l'intervallo di confidenza con cui è noto tale punto, un campo `int32` per rappresentarlo con un indice numerico univoco, ed un campo `string` per associarvi un nome;

- `Link`, usato per indicare il collegamento rigido tra due `Marker`. Oltre ad essere composto dagli stessi campi contenuti in `Marker`, dove in questo caso il `KinematicState` è usato per definire l'orientazione del link, presenta anche due `int32` che rappresentano gli indici dei marker che ne definiscono gli estremi;

- `Box`, definisce un parallelepipedo rettangolo nello spazio. Consta di un campo `KinematicState` per indicarne il centro e tre campi `float64` che contengono le informazioni sulle dimensioni di tale box;

- `Skeleton`, rappresentazione della posa, intera o parziale, di una persona in un dato istante di tempo. Contiene un campo `Box` che indica la *bounding box*[5] dello scheletro, un vettore di marker ed un vettore dei link relativi a tali

[4] Disponibile presso: https://github.com/HiROS-unipd.
[5] Ovvero la "scatola" di volume minimo che lo racchiude.

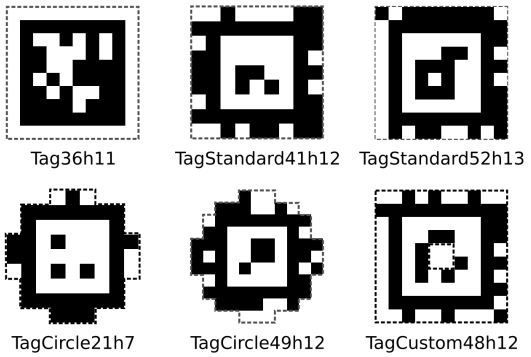


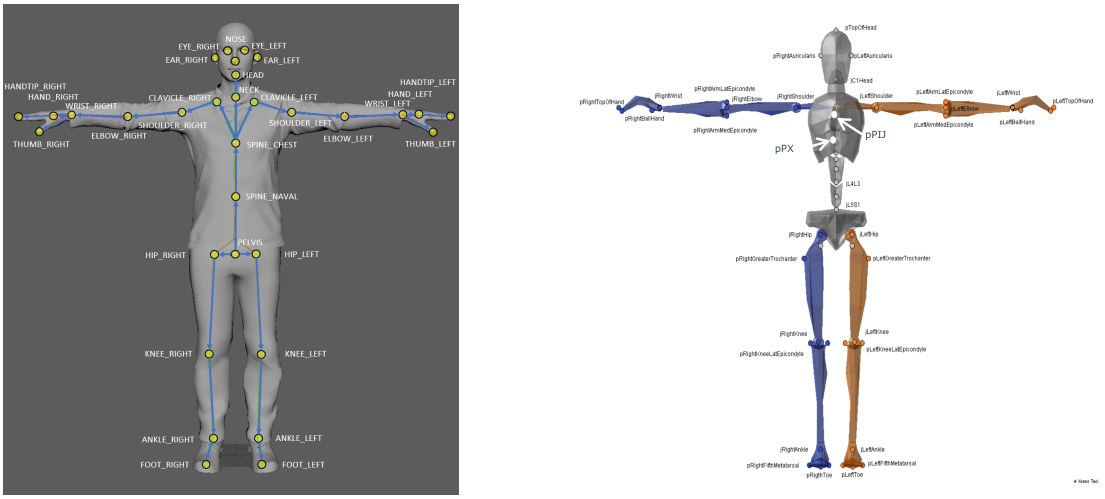

Figura 3.2: Vari esempi di formati di AprilTag (Fonte: [25]).

marker. Oltre a questo, sono anche incluse le informazioni relative all'id dello scheletro (`int32`), al sistema di riferimento a cui si riferisce la posa (`string`), al timestamp dello scheletro (campo di tipo `time`), al numero massimo di marker e link, e all'intervallo di confidenza con cui è nota la posa;

- `SkeletonGroup`, usato per contenere un gruppo di scheletri rappresentati nello stesso istante di tempo e rispetto ad un sistema di riferimento comune. Consta di un `Header` contenente tali informazioni, e di un vettore di `Skeleton`.

Per questo motivo, si farà riferimento alle pose ottenute dai sistemi di MoCap utilizzando il termine "scheletri".

Nel caso in esame, le TF sono utilizzate per rappresentare la relazione esistente tra gli sdr rispetto cui sono espressi gli scheletri ottenuti dai due sistemi di MoCap. Tramite un AprilTag [28] (alcuni esempi sono visibili in Fig. 3.2) viene definito un sdr globale che viene indicato con il nome "world". Gli scheletri ottenuti dalla telecamera Kinect (Fig. 3.3a mostra come sono definiti) sono espressi normalmente in un sdr posizionato alla base della stessa (indicato come "markerless"), ma tramite il processo di calibrazione viene calcolata la TF statica che lega tale sistema a "world". Pertanto, gli scheletri Kinect sono rappresentabili direttamente nel sdr globale, essendo noto a ROS il legame tra i due. Per quanto riguarda invece gli scheletri del MoCap inerziale (definiti come visibile in Fig. 3.3b), essi sono rappresentati in un sdr differente, indicato con il nome "inertial": tale sdr può essere posizionato dall'utente in modo da coincidere inizialmente con "world" tramite una procedura di allineamento manuale, ma




(a) Kinect (Fonte: [26]). (b) Xsens (Fonte: [27]).

Figura 3.3: Definizione dei landmark utilizzati dai sistemi di MoCap Azure Kinect e Xsens MVN.

a causa del fenomeno di drift subisce un graduale ed evidente allontanamento rispetto al sistema globale.

Generalizzando quanto ottenuto con i processi di calibrazione per la Kinect e di allineamento manuale per i sensori Xsens, è possibile considerare la presenza di una TF che permette di esprimere i punti dall'sdr "markerless" ad un generico sistema "markerless root" (rispetto a cui vogliamo esprimere tutti i dati del MoCap Kinect), ed una TF che fa lo stesso con i sdr "inertial" ed "inertial root" (se si decide di esprimere gli scheletri Xsens in un sdr diverso da quello originale). Ne segue che per esprimere nel sistema globale la posa dell'sdr "inertial root" che subisce il fenomeno di drift è necessario chiudere la catena di trasformazioni che manda nel sdr "world" i punti espressi nel sistema di riferimento "inertial root", applicando perciò la seguente operazione:

$$P' = \mathbf{T}^{\mathcal{M}}_{\mathcal{M}_{root}} \mathbf{T}^{\mathcal{I}}_{\mathcal{M}} \mathbf{T}^{\mathcal{I}_{root}}_{\mathcal{I}} P \tag{3.5}$$

In questo caso specifico la prima e la terza matrice possono essere considerate come note[6], essendo rispettivamente quella ottenuta in fase di calibrazione del MoCap markerless ($\mathcal{M}_{root}$ è in questo caso l'sdr "world") e l'identità, se si

[6]Come si vedrà in seguito, esse vanno comunque calcolate, ma vengono ricavate direttamente tramite le funzionalità fornite da ROS.

considera che $\mathcal{I}_{root}$ coincide con $\mathcal{I}$ in cui sono espressi gli scheletri ottenuti dal MoCap inerziale. L'unica incognita è pertanto la matrice $\mathbf{T}^{\mathcal{I}}_{\mathcal{M}}$, che rappresenta la TF tra il sistema di riferimento "inertial" e quello "markerless". È possibile ottenere tale matrice calcolando la trasformazione che permette di sovrapporre, ad ogni istante di tempo, i keypoint che definiscono la posa dello scheletro ottenuto dal MoCap inerziale a quelle dello scheletro del MoCap markerless.

## 3.4 Analisi di Procuste

L'Analisi di Procuste [23] è un metodo dell'analisi statistica tipicamente applicato alla sovrapposizione delle figure geometriche, ma che può essere in generale utilizzato per estrarre una media rappresentativa a partire da degli insiemi di dati, come ad esempio quelli ottenuti a partire da questionari [29]. Nel nostro caso ci è di interesse per la sua tipica applicazione nella sovrapposizione di oggetti aventi la stessa forma[7] ma diverse posizioni ed orientazioni nello spazio, ciascuno dei quali è definito come un insieme di landmark che lo descrive adeguatamente. Ogni oggetto viene quindi rappresentato da una matrice $3 \times n$, dove le colonne corrispondono alle coordinate cartesiane degli $n$ landmark che lo descrivono. Nella sua formulazione più generica il metodo in questione è conosciuto come "Analisi Generalizzata di Procuste" (Generalized Procrustes Analysis, GPA), dove l'aggettivo sta ad indicare il fatto che essa permette di sovrapporre più di due figure geometriche. L'algoritmo della GPA è il seguente:

1. **Scegliere** uno degli oggetti come forma media approssimata.
2. **Allineare** gli oggetti alla forma media approssimata.
3. **Calcolare** la nuova forma media approssimata.
4. **Confrontare** la nuova forma media con la precedente: se la differenza tra le due è superiore ad una soglia prefissata allora scegliere la nuova media come forma di riferimento e ripetere dal passo 2, altrimenti si è trovata la media cercata.

Il problema a cui si applica è detto "Problema Ortogonale di Procuste", indicando il fatto che per realizzare l'allineamento si vanno a calcolare matrici

[7] Ovvero le informazioni geometriche che rimangono una volta rimossi gli effetti di traslazione, rotazione e cambio di scala da un oggetto.

di rotazione e/o riflessione (e perciò ortogonali). I passaggi di cui è composto il punto 2 sono pertanto:

(a) **Calcolare** il centroide di ciascun oggetto.

(b) **Traslare** ogni oggetto sull'origine dell'sdr sottraendo ad ogni punto di ciascuno il proprio centroide.

(c) **Normalizzare** la dimensione di ogni oggetto.

(d) **Allineare** ogni oggetto alla forma media tramite una operazione di rotazione/riflessione, cercando di minimizzare l'errore quadratico medio nel posizionamento di ciascun punto.

Dovendo allineare solamente due forme, la componente iterativa della GPA viene a mancare: ci interessano quindi solamente i punti 1 e 2 dell'algoritmo. Inoltre, nel caso degli scheletri ottenuti dal MoCap, essi possono essere considerati già in scala (rappresentano la stessa persona), quindi il passaggio di normalizzazione delle dimensioni diventa anch'esso superfluo. Infine, il problema ortogonale in questione è ulteriormente vincolato dal fatto che la matrice ortogonale che allinea le due figure al punto (d) deve rappresentare una rotazione e non una riflessione, pertanto è necessario uno step aggiuntivo a valle del calcolo della matrice di "rotazione" per verificare e correggere l'eventualità che essa rappresenti invece una riflessione.

## 3.5 Calcolo della matrice di trasformazione

Il metodo di Procuste viene applicato come descritto nelle seguenti sottosezioni.

### 3.5.1 Calcolo del centroide e traslazione sull'origine

La struttura della generica matrice $\mathbf{K}$ contenente gli $n$ landmark (o keypoint) di uno scheletro espresso nel sdr $\mathcal{F}$ è la seguente:

$$\mathbf{K}^{\mathcal{F}} = \begin{bmatrix} x_0 & x_1 & & x_{n-1} \\ y_0 & y_1 & \dots & y_{n-1} \\ z_0 & z_1 & & z_{n-1} \end{bmatrix} \in \mathbb{R}^{3\times n} \tag{3.6}$$

Per ciascuno scheletro, il centroide è ottenuto come punto medio dei landmark dati, pertanto la matrice delle coordinate traslate sull'origine è:

$$\mathbf{K}_c^{\mathcal{F}} = \mathbf{K}^{\mathcal{F}} - \frac{1}{n}\begin{bmatrix} \sum_{i=0}^{n} x_i & \cdots & \sum_{i=0}^{n} x_i \\ \sum_{i=0}^{n} y_i & \cdots & \sum_{i=0}^{n} y_i \\ \sum_{i=0}^{n} z_i & & \sum_{i=0}^{n} z_i \end{bmatrix} \tag{3.7}$$

Per semplificare la notazione, si indica con $\mathbf{K}_{\mathcal{M}}$ la matrice contenente i punti dello scheletro ottenuto dal MoCap markerless e traslato sull'origine e similmente si utilizza $\mathbf{K}_{\mathcal{I}}$ per quello ottenuto dal MoCap inerziale.

### 3.5.2 Calcolo della matrice di rotazione

Lo step di allineamento si può esprimere come la ricerca della matrice ortogonale $\mathbf{R}$ che soddisfa:

$$\min_{\mathbf{R}} \left\| \mathbf{R}\mathbf{K}_{\mathcal{I}} - \mathbf{K}_{\mathcal{M}} \right\|_F \tag{3.8}$$

Dove $\|\cdot\|_F$ è la norma di Frobenius, definita come:

$$\left\|\mathbf{A}\right\|_F = \sqrt{\sum_{i}^{m}\sum_{j}^{n}\left|a_{ij}\right|^2} = \sqrt{\mathrm{Tr}(\mathbf{A}^\top\mathbf{A})} \tag{3.9}$$

Pertanto, è possibile esprimere $\mathbf{R}$ come:

$$\begin{aligned} \mathbf{R} &= \arg\min_{\mathbf{R}} \left\| \mathbf{R}\mathbf{K}_{\mathcal{I}} - \mathbf{K}_{\mathcal{M}} \right\|_F \overset{(3.9)}{=} \arg\min_{\mathbf{R}} \left( \mathrm{Tr}\left( \left( \mathbf{R}\mathbf{K}_{\mathcal{I}} - \mathbf{K}_{\mathcal{M}} \right)^\top \left( \mathbf{R}\mathbf{K}_{\mathcal{I}} - \mathbf{K}_{\mathcal{M}} \right) \right) \right) = \\ &= \arg\min_{\mathbf{R}} \left( \mathrm{Tr}\left( \mathbf{K}_{\mathcal{I}}^\top\mathbf{R}^\top\mathbf{R}\mathbf{K}_{\mathcal{I}} - \mathbf{K}_{\mathcal{M}}^\top\mathbf{K}_{\mathcal{M}} - \mathbf{K}_{\mathcal{I}}^\top\mathbf{R}^\top\mathbf{K}_{\mathcal{M}} - \mathbf{K}_{\mathcal{M}}^\top\mathbf{R}\mathbf{K}_{\mathcal{I}} \right) \right) \overset{\mathbf{R}\in SO(3)}{=} \\ &= \arg\min_{\mathbf{R}} \left( \mathrm{Tr}\left( \mathbf{K}_{\mathcal{I}}^\top\mathbf{K}_{\mathcal{I}} - \mathbf{K}_{\mathcal{M}}^\top\mathbf{K}_{\mathcal{M}} \right) - \mathrm{Tr}\left( \mathbf{K}_{\mathcal{I}}^\top\mathbf{R}^\top\mathbf{K}_{\mathcal{M}} + \mathbf{K}_{\mathcal{M}}^\top\mathbf{R}\mathbf{K}_{\mathcal{I}} \right) \right) \end{aligned} \tag{3.10}$$

Osservando che $\mathbf{K}_{\mathcal{I}}^\top\mathbf{R}^\top\mathbf{K}_{\mathcal{M}} = \left( \mathbf{K}_{\mathcal{M}}^\top\mathbf{R}\mathbf{K}_{\mathcal{I}} \right)^\top$ e ricordando che per la traccia vale $\mathrm{Tr}(\mathbf{A}) = \mathrm{Tr}(\mathbf{A}^\top)$, è possibile semplificare con:

$$\mathbf{R} = \arg\min_{\mathbf{R}} \left( \mathrm{Tr}\left( \mathbf{K}_{\mathcal{I}}^\top\mathbf{K}_{\mathcal{I}} - \mathbf{K}_{\mathcal{M}}^\top\mathbf{K}_{\mathcal{M}} \right) - 2\mathrm{Tr}\left( \mathbf{K}_{\mathcal{M}}^\top\mathbf{R}\mathbf{K}_{\mathcal{I}} \right) \right) \tag{3.11}$$

Poiché $\mathbf{R}$ non compare al primo addendo, si può riscrivere il problema come:

$$\mathbf{R} = \arg\max_{\mathbf{R}} \left( \mathrm{Tr} \left( \mathbf{K}_{\mathcal{M}}^{\top} \mathbf{R} \mathbf{K}_{\mathcal{I}} \right) \right) \tag{3.12}$$

Definendo la matrice di covarianza $\mathbf{H} = \mathbf{K}_{\mathcal{I}} \mathbf{K}_{\mathcal{M}}^{\top}$ è possibile applicare a quest'ultima la decomposizione ai valori singolari (Singular Value Decomposition, SVD), ottenendo $\mathbf{H} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\top}$, dove $\mathbf{\Sigma}$ è la matrice avente sulla diagonale gli autovalori di $\mathbf{H}$ e $\mathbf{U}, \mathbf{V}$ sono matrici ortogonali. Facendo uso della proprietà di ciclicità della traccia[8] di una matrice e sostituendo la SVD di $\mathbf{H}$ in Eq. (3.12) si ottiene:

$$\mathbf{R} = \arg\max_{\mathbf{R}} \left( \mathrm{Tr} \left( \mathbf{\Sigma} \mathbf{V}^{\top} \mathbf{R} \mathbf{U} \right) \right) \tag{3.13}$$

La matrice $\mathbf{\Sigma}$ è composta da autovalori e quindi non contiene elementi negativi, mentre la matrice $\mathbf{S} = \mathbf{V}^{\top}\mathbf{R}\mathbf{U}$ è il prodotto di tre matrici ortogonali, pertanto è a sua volta ortogonale e questo significa che i suoi autovalori hanno tutti modulo unitario. Ciò significa che per massimizzare la traccia della matrice $\mathbf{\Sigma}\mathbf{S}$ è necessario che $\mathbf{S}$ abbia tutti autovalori unitari, ovvero che sia la matrice identità, da cui:

$$\mathbf{S} = \mathbf{V}^{\top}\mathbf{R}\mathbf{U} = \mathbf{I} \Rightarrow \mathbf{R} = \mathbf{V}\mathbf{U}^{\top} \tag{3.14}$$

È pertanto possibile calcolare la matrice di rotazione che permette di sovrapporre i due scheletri minimizzando l'errore quadratico medio a partire dalle matrici ottenute dalla SVD della matrice di covarianza $\mathbf{H}$. Le funzionalità necessarie a questi calcoli sono fornite dalla libreria `Eigen` [30], che permette di definire matrici di varie dimensioni e di eseguire diverse operazioni su di esse, tra cui la stessa SVD. Poiché gli scheletri provenienti dai due sistemi di MoCap presentano configurazioni diverse, è stato scelto di utilizzare nel calcolo della matrice di rotazione il sottoinsieme dei landmark che meglio corrispondono tra le due topologie. Tali landmark vengono definiti tramite un vettore di indici in un file YAML che viene poi letto dal launch file e passato come parametro al nodo ROS.

[8] $\mathrm{Tr}(\mathbf{ABC}) = \mathrm{Tr}(\mathbf{BCA}) = \mathrm{Tr}(\mathbf{CAB})$

### 3.5.3 Verifica della matrice di rotazione

Resta da verificare che la matrice ortogonale ottenuta al passaggio precedente rappresenti una rotazione e non una riflessione. Per farlo è sufficiente calcolare il determinante di $\mathbf{R}$: per essere una matrice di rotazione deve infatti valere $\det(\mathbf{V}\mathbf{U}^\top) = 1$, mentre nel caso in cui $\det(\mathbf{V}\mathbf{U}^\top) = -1$, essa rappresenterebbe una matrice di riflessione. Qualora ciò dovesse accadere, è possibile risalire alla matrice di rotazione propria cercata nel modo seguente:

$$\mathbf{R} = \mathbf{V} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \mathbf{U}^\top \tag{3.15}$$

L'algoritmo per il calcolo di $\mathbf{R}$ discusso in 3.5.2 e 3.5.3 ed implementato nel pacchetto `skeleton_aligner` è noto come Algoritmo di Kabsch-Umeyama [31].

### 3.5.4 Calcolo della traslazione

Il passaggio successivo consiste nel calcolare la traslazione che, applicata alla matrice che rappresenta lo scheletro inerziale, ne allinea il centroide a quello dello scheletro markerless. Indicando con $C_\mathcal{M}$ il centroide dello scheletro markerless e con $C_\mathcal{I}$ quello dello scheletro inerziale, è possibile calcolare la traslazione passando alle coordinate omogenee. La trasformazione che permette di passare da un punto all'altro è infatti:

$$\begin{bmatrix} C_\mathcal{M} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} C_\mathcal{I} \\ 1 \end{bmatrix} \tag{3.16}$$

Dove $\mathbf{R}$ è la matrice di rotazione calcolata nei punti precedenti e $\mathbf{t}$ indica il vettore $3 \times 1$ che rappresenta la traslazione cercata. Ne segue che:

$$C_\mathcal{M} = \mathbf{R}C_\mathcal{I} + \mathbf{t} \Rightarrow \mathbf{t} = C_\mathcal{M} - \mathbf{R}C_\mathcal{I} \tag{3.17}$$

Similmente a quanto fatto nel caso della rotazione, i landmark utilizzati nel calcolo della componente di traslazione sono definiti nel file YAML di configurazione. Come nel caso precedente, anche per questa operazione è possibile selezionare quali keypoint degli scheletri si vuole andare ad allineare con la traslazione, definendo un apposito vettore di indici che volendo può essere differente da quello utilizzato per la rotazione.

Questo conclude il calcolo della matrice di trasformazione che permette di allineare lo scheletro inerziale a quello markerless minimizzando l'errore quadratico medio di posizionamento dei punti corrispondenti. Essendo un calcolo eseguito ad ogni frame di aggiornamento degli scheletri, esso è soggetto ad effetti di rumore che influenzano la qualità del risultato, per questo motivo è necessario implementare un algoritmo che riduca tali effetti mediante una operazione di media delle TF.

## 3.6 Calcolo della TF media

Bisogna tenere in considerazione che gli scheletri sono oggetti in movimento e che la posizione ed orientazione relative tra i due variano nel tempo a causa del fenomeno di drift a cui è soggetto il MoCap inerziale. Questo comporta che la matrice di trasformazione non sia costante nel tempo e vada ricalcolata continuamente per assicurarsi di conoscere il posizionamento corretto dell'sdr inerziale rispetto a quello globale. A ciò si aggiunge anche il fatto che non necessariamente i due sistemi di MoCap pubblicano aggiornamenti alla stessa frequenza (cosa che infatti non accade) e non è assicurato che gli scheletri pubblicati siano completi, specialmente nel caso del MoCap markerless a causa dei fenomeni di occlusione o di uscita dal campo visivo della videocamera. Infine, la minore accuratezza del MoCap markerless rispetto a quello inerziale si manifesta tipicamente con la presenza di maggiore rumore nella stima delle pose, che a sua volta influenza le TF calcolate con l'algoritmo precedentemente descritto. La composizione di tutti questi effetti comporta la possibilità di variazioni repentine della TF al passare del tempo, che si può osservare dalla presenza di "sfarfallii" nella visualizzazione dello scheletro inerziale riallineato, compromettendo l'accuratezza dell'intero sistema.

Per evitare tale fenomeno, si è deciso di implementare un buffer all'interno del pacchetto `skeleton_aligner`, così da tenere in memoria le informazioni delle TF precedentemente calcolate e poterne ricavare la media pesata nel tempo. Considerato che non tutte le TF sono necessariamente rappresentative del rapporto che lega il sistema di riferimento inerziale a quello globale (ad esempio a causa di uno scheletro markerless affetto da eccessivo rumore di misura), il buffer impiega un metodo di clustering non supervisionato per separarle in gruppi diversi a seconda della posizione ed orientazione che esse rappresentano, così da considerare solamente l'insieme di TF che meglio descrive il legame tra i

due sdr. Il buffer contiene quindi un vettore di elementi di tipo `Cluster`, classe definita per gestire il calcolo della media delle TF.

### 3.6.1 Gestione del Buffer

L'algoritmo che descrive come viene gestito il buffer tramite il suo metodo principale, chiamato `updateClusters`, è descritto in Alg. 1.

**Algoritmo 1** Metodo `updateClusters()`

**Input:** TF calcolata da `Aligner`

```
timeBasedCleanup()                                  // Elimina cluster troppo datati
se ∄ cluster allora                                 // Popolamento dei cluster
    Crea primo cluster
altrimenti
    per ogni cluster esegui
        Calcola distanza della TF dal cluster
    fine
    se ∃ cluster t.c. (distanza < distanza_clustering) allora
        Inserisci TF nel cluster t.c. min(distanza)
    altrimenti
        Crea nuovo cluster
    fine
fine
sortClusters()                                      // Ordina cluster per dimensione
distanceBasedCleanup()                              // Elimina cluster troppo vicini al principale
```

La funzione `timeBasedCleanup` elimina i cluster il cui elemento più recente ha una età (calcolata come la differenza tra il tempo attuale e quello di ricezione della TF) superiore ad una soglia che può essere modificata dall'utente. La `distanza` tra la TF e ciascun cluster è calcolata utilizzando la TF media del cluster, ed è scelta come il massimo fra la distanza euclidea tra i vettori $\mathbf{t}$ e quella angolare tra le matrici di rotazione $\mathbf{R}$, ottenuta a partire dai quaternioni ad esse equivalenti e tenendo conto del percorso più breve tra i due.

La distanza angolare tra due quaternioni $p$ e $q$ si può indicare come il quaternione $q_r$ tale che:

$$q_r p = q \implies q_r = q p^{-1} \tag{3.18}$$

Considerando la notazione scalare/vettore dei quaternioni, per cui $p = (p_0, \mathbf{v})$ dove $p_0 \in \mathbb{R}$ e $\mathbf{v}$ è un vettore in $\mathbb{R}^3$, e ricordando che l'inverso di un quaternione corrisponde al suo coniugato $\overline{p} = (p_0, -\mathbf{v})$, è possibile scrivere:

$$q_r = (q_0, \mathbf{w})(p_0, -\mathbf{v}) = (p_0 q_0 - \mathbf{v} \cdot \mathbf{w}, -p_0 \mathbf{w} - q_0 \mathbf{v} + \mathbf{w} \times \mathbf{v}) = (q_{r0}, \mathbf{v}_r) \tag{3.19}$$

Poiché è possibile esprimere ciascun quaternione unitario nella forma $q = (\cos \frac{\phi}{2}, \sin \frac{\phi}{2} \mathbf{u})$, a cui è associata la rotazione destrorsa di angolo $\phi \in [0, 2\pi]$ attorno all'asse identificato dal versore $\mathbf{u}$, allora $q_r$ identifica una rotazione tale che:

$$\cos \frac{\phi}{2} = q_{r0} \Longrightarrow \phi = 2 \arccos q_{0r} \tag{3.20}$$

Ovviamente bisogna tenere in considerazione il fatto che in realtà ad ogni rotazione nello spazio tridimensionale corrispondono due quaternioni opposti. Per poter calcolare la distanza angolare lungo il percorso più breve è quindi necessario valutare il segno del prodotto scalare $q_{r0}$ tra i quaternioni: se esso risulta essere negativo allora l'angolo ottenuto dalla Eq. (3.20) sarebbe quello superiore a 180 deg e quindi lungo il percorso più lungo. In tal caso è necessario sostituire $q$ con $-q$ nel calcolo di $q_{r0}$. La libreria `Eigen` implementa proprio in questo modo il calcolo della distanza angolare tra due quaternioni.

Nell'implementazione del sistema di allineamento delle pose, le distanze tra TF (euclidea ed angolare) sono normalizzate all'interno del range di valori $(0, 1]$ tramite la definizione di parametri per le massime distanze da considerare: qualsiasi distanza superiore ad esse viene limitata ad 1. La funzione `sortClusters` ordina i cluster in ordine decrescente di dimensioni, in modo tale da avere i cluster disposti in ordine dal più al meno rappresentativo della TF reale: un cluster di dimensioni elevate è infatti popolato da un grande numero di TF vicine tra loro, ed è più probabile che esse rappresentino la vera trasformazione che lega i due sdr, mentre un cluster di piccole dimensioni è verosimilmente popolato da TF spurie ottenute a causa di effetti quali il rumore di misura. Il primo cluster dopo il riordinamento è pertanto quello che viene utilizzato per calcolare la TF media che il pacchetto `skeleton_aligner` deve pubblicare. La funzione `distanceBasedCleanup` calcola la distanza tra le TF medie dei primi due cluster ed elimina il secondo qualora esso raggiunga una distanza troppo vicina a quella del primo. Nel caso di presenza tra i primi posti di due cluster aventi medie quasi uguali, infatti, si potrebbe verificare la presenza di due gruppi di TF molto simili ma non abbastanza da formare un unico gruppo, cosa che può comportare una crescita parallela dei due ed il conseguente alternarsi dei

due nel ruolo di cluster principale. Questo porta ad un effetto di sfarfallio nella visualizzazione dello scheletro allineato, dovuto alla continua alternanza nella pubblicazione delle TF medie dei due cluster. La rimozione dei cluster secondari che raggiungono distanze troppo vicine al primo permette di evitare questo effetto e di far evolvere il cluster principale in modo più organico.

### 3.6.2 Gestione dei Cluster

Come detto in precedenza, il buffer contiene un vettore di cluster, ciascuno dei quali rappresenta un insieme di TF aventi distanza limitata dal resto delle TF che lo popolano. Scopo di ciascun cluster è pertanto quello di tenere aggiornato il valore medio delle TF che esso contiene, ed esporlo al buffer quando quest'ultimo lo richiede per la pubblicazione da parte della classe Aligner. Per calcolare la TF media è stato scelto di ricorrere ad una media pesata nel tempo, in modo che le componenti più datate del cluster influenzassero meno il valore finale della media. Questa operazione presenta tuttavia due problematiche:

1. Gli oggetti che stiamo trattando sono matrici di trasformazione espresse in coordinate omogenee. Fare una media pesata "semplice" di tali dati può comportare la perdita del significato geometrico degli stessi, specialmente per quanto riguarda la componente rotazionale delle matrici.

2. Salvare tutte le TF passate per una implementazione tradizionale della media pesata è di fatto impossibile, sia per motivi di occupazione della memoria che di degradazione delle performance dovuta all'aumento delle operazioni che è necessario eseguire man mano che cresce la quantità di TF da calcolare. Questo è ovviamente in contrasto con i requisiti del progetto, che può vedere il sistema di allineamento in esecuzione per lunghi periodi di tempo.

#### ▹ Realizzazione della media pesata delle TF

Il primo problema è stato risolto utilizzando un algoritmo che calcola la media pesata con un metodo ai minimi quadrati cercando la soluzione del problema algebrico $\mathbf{Ax} = \mathbf{b}$, dove $\mathbf{A}$ e $\mathbf{b}$ sono ottenute a partire dalle componenti rotazionali e traslazionali provenienti da vettori di TF: questa formulazione ricerca $\mathbf{x}$ che minimizza la differenza tra $\mathbf{Ax}$ e $\mathbf{b}$. Nello specifico, al generico passo $k$ si ha che $\mathbf{b}$ è ricavata a partire dal vettore delle $k + 1$ TF che si intende mediare, ed ha la seguente forma:

$$\mathbf{b} = \begin{bmatrix} w_0\mathbf{b}_0 \\ \vdots \\ w_k\mathbf{b}_k \end{bmatrix} \in \mathbb{R}^{12(k+1)\times 1} \tag{3.21}$$

Dove $w_i = \sqrt{w^{k-i}}$ e $\mathbf{b}_i = [R_{00}\ R_{01}\ R_{02}\ t_0 \ldots R_{20}\ R_{21}\ R_{22}\ t_2]^\top$ è la colonna $12\times 1$ ottenuta leggendo le prime tre righe della matrice $T_i$, ovvero tutte le informazioni rotazionali e traslazionali della TF. Il peso $w$ da applicare è definito dall'utente tramite un parametro del nodo `skeleton_aligner` e deve appartenere all'intervallo $(0, 1]$. Come è possibile osservare esso è applicato in modo tale da essere unitario per la matrice di trasformazione più recente e decresce in modo esponenziale per le TF più datate, riducendone l'effetto sulla media. Per $w \to 1$ si va ad aumentare il peso delle TF meno recenti e quindi il filtraggio del rumore, a discapito della velocità di risposta a movimenti più veloci. La matrice $\mathbf{A}$ è invece ottenuta a partire da un vettore di $k+1$ TF identità, ed ha la seguente struttura:

$$\mathbf{A} = \begin{bmatrix} w_0\mathbb{I}_{12} \\ \vdots \\ w_k\mathbb{I}_{12} \end{bmatrix} \in \mathbb{R}^{12(k+1)\times 12} \tag{3.22}$$

Dove $\mathbb{I}_n$ è la matrice identità di dimensione $n$. Costruendo $\mathbf{A}$ in questo modo si cerca la soluzione $\mathbf{x} \in \mathbb{R}^{12\times 1}$ (con la stessa struttura dei $\mathbf{b}_i$) che minimizza la differenza quadratica con il set di TF da mediare, opportunamente pesate[9]. Il problema algebrico è risolto facendo ricorso all'implementazione "Divide and Conquer" dell'SVD fornita dalla libreria `Eigen`, perché utilizzando la scomposizione della matrice $\mathbf{A} = \mathbf{V}\mathbf{\Sigma}\mathbf{U}^\top$ la soluzione del problema risulta più stabile numericamente e la particolare implementazione sfruttata è ottimizzata per matrici di grandi dimensioni. Si procede poi alla ricostruzione della matrice di trasformazione a partire dal vettore colonna $\mathbf{x}$ ed alla riortogonalizzazione della componente rotazionale tramite una nuova applicazione dell'SVD per assicurarsi che la componente $\mathbf{R}$ ottenuta non abbia perso la proprietà di ortogonalità a causa di errori di approssimazione numerica.

[9] L'uso della radice quadrata è dovuto al fatto che, intuitivamente, $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ e quindi ciascun peso viene moltiplicato per se stesso.

### Ottimizzazione del codice

Per quanto riguarda la seconda problematica evidenziata in precedenza, una prima possibile soluzione può essere quella di limitare il numero di elementi che un cluster può contenere. Questo ovviamente andrebbe a limitare l'occupazione massima di memoria ed il numero di operazioni di calcolo da eseguire per ottenere la media pesata. Tuttavia questo metodo presenta lo svantaggio di dover implementare delle funzioni per gestire l'inserimento di un nuovo elemento in un cluster che ha raggiunto la massima capacità, andando ad eliminare l'elemento più datato e richiedendo di far scorrere tutti gli elementi di una posizione prima di poter procedere all'inserimento. Inoltre, la cancellazione degli elementi meno aggiornati fa perdere la parte a lungo termine dell'effetto di memoria della media pesata, sebbene impostando un peso diverso da 1 esso sia tecnicamente di lieve entità nel caso degli elementi più datati (a cui sono associati dei pesi molto bassi). Infine, è comunque richiesto di salvare un sufficiente numero di TF per poter avere una rappresentazione abbastanza accurata della media, e ciò va fatto anche per i cluster secondari che, sebbene inutilizzati dal punto di vista del calcolo della media "reale", devono esporre la propria media per poter informare l'Alg. 1 su dove posizionare ciascuna nuova TF.

Una soluzione alternativa è quella di ricercare una formulazione iterativa della media pesata, in quanto permetterebbe di ridurre ad un solo elemento (la media precedente) la quantità di dati da mantenere in memoria senza tuttavia rinunciare alle informazioni che le precedenti TF hanno apportato al cluster. Questo ovviamente a meno della discesa sotto il limite di rappresentabilità dei numeri in virgola mobile, che però avviene in modo sicuramente più graduale rispetto al troncamento ad un numero arbitrario di TF salvate in memoria precedentemente ipotizzato.

Avendo già definito l'algoritmo che gestisce il calcolo della media pesata di un insieme ordinato di TF senza perdita di significato della componente rotazionale, la ricerca della formula iterativa si riduce a quella per la media

pesata tradizionale di matrici $4 \times 4$, che può essere scritta come:

$$\overline{\mathbf{T}}_k = \frac{\sum_{i=0}^{k} w^{k-i}\mathbf{T}_i}{\sum_{i=0}^{k} w^{k-i}} = \frac{\mathbf{T}_k + \sum_{i=0}^{k-1} w^{k-i}\mathbf{T}_i}{1 + \sum_{i=0}^{k-1} w^{k-i}} = \frac{\mathbf{T}_k}{1 + \sum_{i=0}^{k-1} w^{k-i}} + \frac{w\sum_{i=0}^{k-1} w^{k-1-i}\mathbf{T}_i}{1 + \sum_{i=0}^{k-1} w^{k-i}} =$$
$$= \frac{\mathbf{T}_k}{1 + \sum_{i=0}^{k-1} w^{k-i}} + \frac{w\sum_{i=0}^{k-1} w^{k-1-i}\mathbf{T}_i}{1 + \sum_{i=0}^{k-1} w^{k-i}} \frac{\sum_{i=0}^{k-1} w^{k-1-i}}{\sum_{i=0}^{k-1} w^{k-1-i}} = \frac{\mathbf{T}_k}{1 + \sum_{i=0}^{k-1} w^{k-i}} + \frac{\overline{\mathbf{T}}_{k-1}\sum_{i=0}^{k-1} w^{k-i}}{1 + \sum_{i=0}^{k-1} w^{k-i}} \tag{3.23}$$

La formula iterativa è pertanto:

$$\overline{\mathbf{T}}_k = \frac{\mathbf{T}_k + w_k^*\overline{\mathbf{T}}_{k-1}}{1 + w_k^*}, \quad w_k^* = \sum_{i=0}^{k-1} w^{k-i} = w_{k-1}^* + w^k \tag{3.24}$$

Ovvero la media pesata al passo $k$ è la media tra la più recente TF e la media precedente, pesata con un $w^*$ pari alla somma tra il peso precedente e $w^k$. Al passo 0 la formula viene inizializzata con $\overline{\mathbf{T}}_0 = \mathbf{T}_0$, $w_0^* = 0$. L'algoritmo precedentemente descritto è quindi utilizzato sfruttando questa formulazione ogni volta che un cluster deve calcolare il proprio valore medio aggiornato, permettendo di limitare le dimensioni di $\mathbf{A}$ e $\mathbf{b}$ al calcolo della media tra due soli elementi (ovvero nel caso $k = 1$) e pertanto il numero di operazioni da svolgere. Ciascun cluster inoltre deve tenere in memoria solamente il valore della propria dimensione attuale, del peso cumulativo e della precedente TF media al posto di un intero vettore di TF, riducendo l'occupazione di memoria. Questo ha portato all'eliminazione del problema dei tempi di calcolo crescenti con l'aumento delle dimensioni che si era osservato in una precedente versione del software che non faceva uso di tale step di ottimizzazione.

## 3.7 Pubblicazione della TF

Una volta ottenuta la TF media che allinea i due scheletri, la classe Aligner procede con la pubblicazione tramite le funzionalità rese disponibili dalla libreria tf2. In primo luogo vengono cercati i `root_frame` di ciascuno scheletro, ovvero i sdr in cui sono espressi ("world" per il MoCap markerless e "xsens" per quello inerziale). Questa procedura viene svolta andando a risalire il `root_tree`

di ciascuna TF, e permette di generalizzare l'applicazione del codice in modo che funzioni anche invertendo i due scheletri o usando scheletri provenienti da sistemi MoCap con specifiche diverse. Esso rende inoltre possibile l'utilizzo del sistema di allineamento delle pose per svolgere una calibrazione estrinseca di due sistemi di MoCap markerless, senza la necessità di ricorrere ad AprilTag o altri metodi per la definizione di un sdr globale ma tramite il semplice allineamento dei due scheletri ottenuti.

Una volta ottenuti i `root_frame` di ogni scheletro, viene letta la TF che esplicita la relazione tra ogni scheletro ed il suo root, corrispondenti alla prima ed all'inversa della terza matrice di trasformazione che compaiono nell'Eq. (3.5). Tali TF sono statiche, in quanto per ciascuno scheletro è costante nel tempo la posizione rispetto al proprio root, lo spostamento causato dal drift avviene infatti tra l'sdr root dello scheletro inerziale e quello markerless. Esse vengono ricavate da un oggetto di tipo `tf2_ros::Buffer`, associato ad un `tf2_ros::TransformListener` che è in grado di leggere tutte le TF pubblicate nel sistema ROS. Poiché $\mathbf{T}_{\mathcal{M}}^{\mathcal{I}}$ è la media pesata calcolata dal pacchetto `skeleton_aligner`, è finalmente possibile applicare l'equazione e trovare la TF che indica la posizione del sdr "inertial root" in quello globale. Pubblicando tale TF tramite un `tf2_ros::StaticTransformBroadcaster`, ROS è in grado di mappare correttamente nel sdr "world" i landmark degli scheletri ottenuti dal MoCap inerziale.

Come infatti visibile in Fig. 3.4 il sistema permette di riallineare lo scheletro inerziale a quello markerless e quindi correggere il fenomeno di drift di notevole entità (l'immagine proviene da un esperimento della durata totale di circa 60 min) che si manifesta durante l'uso del MoCap inerziale.

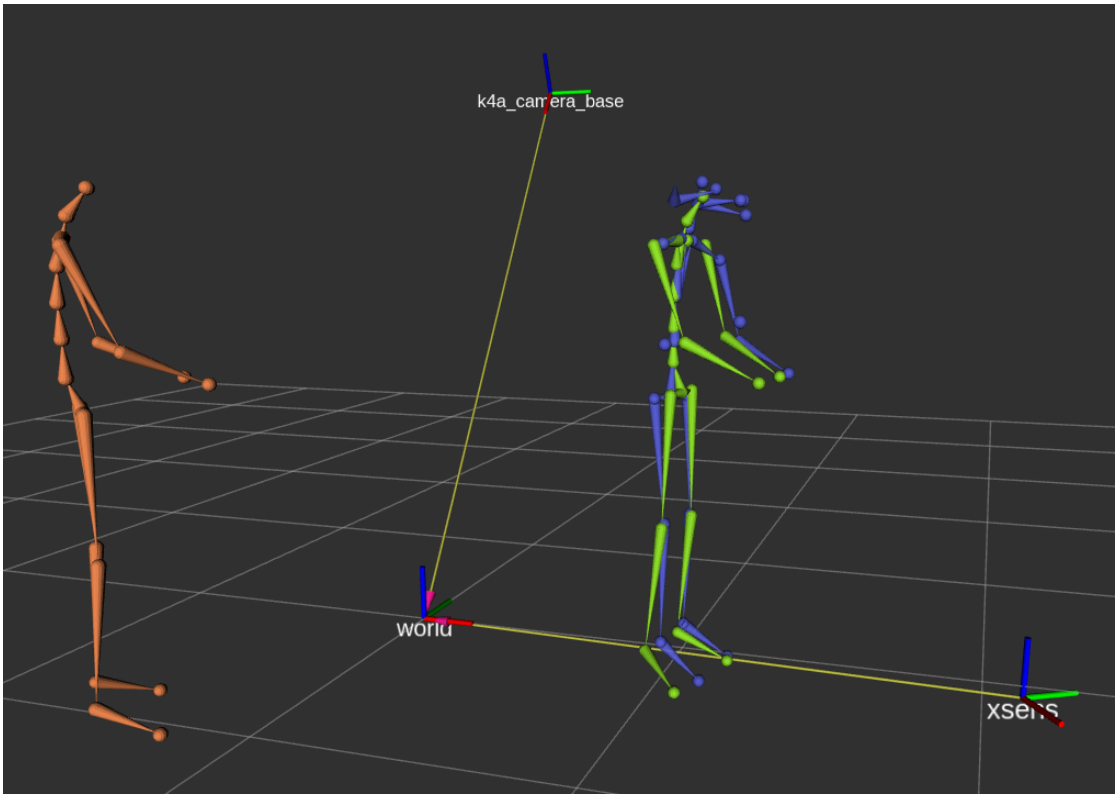



Figura 3.4: Esempio del sistema di allineamento in azione. In viola lo scheletro ottenuto dal MoCap markerless, in arancione quello ottenuto dal MoCap inerziale non allineato ed in verde nella sua versione allineata.

# 4 Raccolta dei segnali fisiologici

Per quanto concerne i dati biometrici, come descritto in precedenza in 2.3.2, la scelta è ricaduta sui sensori di attività elettro-dermica e sull'elettrocardiogramma, utilizzando rispettivamente il sistema biosignalsplux con sensori EDA ed il sensore ECG Polar H10. Questo tipo di misurazioni si presta ad essere utilizzata per una quantità elevata e molto varia di possibili analisi, le quali tuttavia non necessariamente rappresentano informazioni utili per la rappresentazione dello stato di benessere di un lavoratore in ambienti industriali. Poiché la scelta di quante e quali analisi svolgere a partire da tali misure esula dagli obiettivi del progetto, si è deciso di realizzare una integrazione dei sensori scelti che non presenta funzionalità di ulteriore elaborazione dei dati raccolti. Il framework proposto nel lavoro di tesi è infatti orientato alla realizzazione di un sistema in grado di sfruttare queste e potenzialmente altre tipologie di sensori disponibili a livello commerciale per poter raccogliere, ed in futuro elaborare, i dati relativi al benessere fisico e mentale di un operatore in tempo reale ed esporli in tempo reale in modo efficiente ad un sistema capace di produrre una stima online dello stato di salute dell'operatore. Per entrambi i sistemi di sensori ci si è quindi posti l'obiettivo di realizzare nodi ROS in grado di gestire la comunicazione sotto ogni suo aspetto: stabilendo la connessione Bluetooth con il sistema, avviando le operazioni di lettura e pubblicazione dati del sensore, leggendo tali dati ed infine convertendoli attraverso messaggi adeguati allo scopo e ripubblicandoli in appositi topic ROS.

## 4.1 Sensore EDA

Per biosignalsplux vengono forniti degli SDK dall'azienda, compresa una versione scritta in C++[1] e compatibile con i maggiori sistemi operativi (tra cui versioni a 32 bit), dimostrando ampi livelli di compatibilità con lo sviluppo di software su sistemi diversi. Nel caso di sistemi operativi GNU/Linux, viene fornito il file header `plux.h` che definisce le interfacce delle classi e funzioni implementate nella libreria statica `plux.a`. Essa è una libreria precompilata,

[1]Disponibile presso: https://github.com/pluxbiosignals/cpp-samples.

pertanto non è possibile accedere all'implementazione, ma l'header in questione è ben documentato e fornisce funzionalità di comunicazione e raccolta dati per le diverse famiglie di prodotti e sensori vendute dall'azienda. Oltre alla libreria, l'API rilasciata da Plux contiene anche un esempio di implementazione delle funzionalità possibili all'interno di una classe chiamata `SimpleDev`, mostrando come è possibile gestire la connessione, la lettura di informazioni relativi al sistema a cui si è connessi, l'avvio e arresto della lettura dati, la definizione delle callback in risposta agli eventi come l'arrivo di un frame di dati e la gestione delle eccezioni previste dal sistema. La presenza di queste librerie mature e ben documentate rende il sistema sicuramente promettente dal punto di vista dello sviluppo di software più avanzati rispetto al driver realizzato in questo lavoro.

Il driver realizzato ha come parametri di ingresso l'indirizzo fisico del device nel formato `BTHXX:XX:XX:XX:XX:XX`, il nome del topic su cui pubblicare i dati del sensore, un array contenente gli indici delle porte (numerate da 1 a 4) da cui si vogliono leggere i dati, la frequenza di campionamento di ciascuna porta in Hz espressa come un array di numeri in virgola mobile ed infine la risoluzione in bit con cui si vuole procedere alla misura per ciascuna delle porte, anche in questo caso espressa come un array di interi. Il nodo ha tra i propri elementi la classe `PluxSignals`, la quale eredita la classe `Plux::SignalsDev` definita nell'header `plux.h`. La classe `Plux::SignalsDev` descrive il comportamento di un oggetto pensato per la gestione della comunicazione con i dispositivi Plux, e presenta le dichiarazioni di alcune callback pensate per essere modificate mediante overriding[2] dagli sviluppatori che utilizzano l'SDK. Dopo aver inizializzato la comunicazione sulla rete ROS, il nodo procede ad istanziare l'oggetto della classe `PluxSignals`, il quale va a connettersi al dispositivo, ad avviare la sessione di acquisizione dei dati da parte delle porte indicate e ad invocare la funzione `loop` (implementata in `plux.a`) che gestisce i messaggi in arrivo. Tra i metodi che è possibile modificare mediante overriding vi è la funzione di callback `onRawFrame`, a cui vengono passati come parametri dal loop il numero di sequenza che identifica il messaggio ricevuto ed un array di interi contenente la misura dei dati di ciascuna porta, nello stesso ordine con cui le porte sono indicate all'avvio

[2]Con *overriding* si intende il fatto che qualora una classe base ed una classe da essa derivata presentino tra i propri metodi la stessa dichiarazione di funzione, allora la funzione presente nella classe derivata sovrascrive quella della classe base. In questo modo è possibile modificare il comportamento di un metodo a seconda del tipo a cui appartiene l'oggetto che la invoca.

della sessione di registrazione. Dopo aver salvato il dato all'interno della classe `PluxSignlas`, la funzione `onRawFrame` ritorna con un valore booleano `true`, cosa che permette alla funzione `loop` di terminare; a questo punto il nodo richiede il dato, lo converte da vettore di interi (come è trattato nella libreria Plux) ad array di interi (come ROS gestisce dati di tipo intero) in modo tale da poter essere pubblicato nel topic indicato in fase di inizializzazione. All'arresto del nodo viene anche interrotta la sessione di registrazione del sensore e chiusa la connessione.

Il driver è quindi in grado di gestire tutte le fasi necessarie alla comunicazione con il sistema biosignalsplux e di pubblicarne i dati grezzi in tempo reale, anche se è resta da definire una rappresentazione più significativa delle misure di EDA che il sensore permette di raccogliere. Per questo motivo sarà necessario interfacciarsi con esperti del campo della medicina e dell'ingegneria biomedica nei prossimi step evolutivi del framework multimodale, come pure andrà valutato l'utilizzo di strumenti capaci di gestire i dati di questo tipo per garantirne la funzionalità online, come ad esempio metodi di Machine Learning.

## 4.2 Sensore ECG

Per quanto riguarda il sensore Polar H10, sebbene l'azienda renda disponibili vari strumenti sul proprio sito, essi sono pensati per applicazioni su dispositivi mobili[3]. Tuttavia, poiché il sistema aderisce alle specifiche Bluetooth Low Energy (BLE) definite dal Bluetooth SIG, esso permette di comunicare sfruttando i Profili e Servizi definiti sulla base del protocollo GATT che regola la comunicazione tra generici dispositivi BLE. Il protocollo GATT descrive una struttura gerarchica per il salvataggio dei dati: ciascun dato è definito tramite una unità di memorizzazione detta Caratteristica, che contiene il Valore ed un Descrittore opzionale. Un Servizio è composto di una o più caratteristiche, e rappresenta una unità organizzativa dei dati a seconda del loro significato (ad esempio il servizio delle informazioni di dispositivo contiene dati come l'indirizzo fisico, il livello di batteria *etc*.). Infine, un Profilo è definito come un insieme di servizi (alcuni dei quali opzionali) e rappresenta quindi un caso d'uso per il device sulla base dei dati che esso può memorizzare e trasmettere. Ciascuno singolo Profilo, Servizio e Caratteristica è identificato tramite un numero a 128 bit detto

[3]Si veda ad esempio: https://github.com/polarofficial/polar-ble-sdk.

identificativo univoco universale (Universally Unique Identifier, UUID), tramite il quale è quindi possibile interfacciarsi con un dispositivo BLE e comprendere i dati che esso espone. Il Bluetooth SIG ha riservato per la definizione degli attributi dello standard gli UUID nel formato esadecimale `xxxxxxxx-0000-1000-8000-00805F9B34FB`, in modo tale da poter identificare i GATT tramite alias a 16 o 32 bit, ma permette ai costruttori di implementare Profili, Servizi e Caratteristiche custom utilizzando gli UUID esterni a tale "intervallo" (è infatti contiguo solo se si considera un ordine dei byte little-endian). Sebbene non ci sia alcun organismo di registrazione degli UUID a 128 bit utilizzati e la probabilità che due implementatori generino lo stesso UUID non sia nulla, essa è comunque sufficientemente bassa ($\approx 2.94 \times 10^{-39}$) da poter considerare tale evenienza quasi impossibile e pertanto garantisce l'unicità degli identificativi, anche escludendo gli oltre 4 miliardi di UUID riservati dal SIG.

Questo permette di sviluppare driver capaci di gestire la comunicazione in modo simile a quanto visto per il biosignalsplux, grazie all'uso delle librerie per la gestione dei dispositivi Bluetooth esistenti, anche se a causa della mancanza di una libreria per uso diverso dalle applicazioni mobili è necessario ricorrere a comandi di livello più basso, interfacciarsi tramite gli UUID ed inviare direttamente i comandi di avvio/arresto della trasmissione dati mediante gli OPCODE in formato esadecimale che l'azienda ha definito nel proprio SDK. Dalla documentazione ufficiale emerge come sia possibile ottenere i dati del tracciato ECG, della frequenza del battito cardiaco, dell'intervallo RR ed eventualmente i dati dell'accelerometro [32]. Dal momento che la documentazione disponibile si limita ad esempi realizzati in Python e che le librerie per la gestione delle connessioni Bluetooth generiche forniscono interfacce di più semplice utilizzo in questo linguaggio di programmazione, è stato deciso in questo caso di non realizzare il driver usando il linguaggio C++ ma di farlo appunto in Python. Questa scelta è ulteriormente giustificata dal fatto che il nodo realizzato non esegue operazioni dall'elevata complessità computazionale, e pertanto non si rischia di ottenere performance inadeguate ai requisiti industriali. Eventuali operazioni di post-processing da svolgere sui dati possono infatti essere eseguite da nodi differenti implementati in C++, permettendo migliori prestazioni ed ottimizzazione dell'uso della memoria. La scelta è stata fatta anche per compensare la necessità di interfacciarsi con una libreria sviluppata per applicazioni di tipo diverso da quanto richiesto e pertanto più difficile da utilizzare rispetto a quella fornita per l'altro tipo di sensore.

**GATT Service and Characteristics Declaration**

| Service Name | Characteristics Name | Property | Optional Property | Security Permission | UUID |
|---|---|---|---|---|---|
| PMD Service | NA | | | | FB005C80-02E7-F387-1CAD-8ACD2D8DF0C8 |
| | PMD Control Point | Read, Write, Indicate | | None | FB005C81-02E7-F387-1CAD-8ACD2D8DF0C8 |
| | PMD Control Point Client Characteristic Configuration Descriptor | Read, Write | | None | |
| | PMD Data MTU Characteristic | Notify | Indicate | None | FB005C82-02E7-F3870-1CAD-8ACD2D8DF0C8 |
| | PMD Data MTU Client Characteristics Configuration Descriptor | Read, Write | | None | |

Tabella 4.1: Identificativi GATT definiti da Polar per il proprio servizio personalizzato (Fonte: [32]).

Come possibile osservare in Tab. 4.1, Polar definisce un servizio personalizzato detto PMD ed utilizza il base-ID `XXXXXXXX-02E7-F387-1CAD-8ACD2D8DF0C8` per la definizione dei propri UUID. Sono inoltre presenti due caratteristiche individuate da UUID e pertanto accessibili mediante GATT: la prima, PMD Control Point, è utilizzata per l'invio dei comandi al device ed è individuata tramite l'identificativo a 32 bit `0xFB005C81`, mentre la seconda è identificata con `0xFB005C82` ed è utilizzata per attivare e disattivare la sottoscrizione ai servizi di notifica del device che inviano un messaggio ai subscriber ogni volta che il dato contenuto nella caratteristica varia (ovverosia ad ogni aggiornamento della misura). L'interazione con queste caratteristiche è fondamentale per poter comunicare con il dispositivo ed ottenere i dati non definiti nel servizio standard per i dispositivi che misurano il battito cardiaco, in questo caso ad esempio per ottenere l'ECG.

Tramite il servizio PMD control point è possibile richiedere le impostazioni da utilizzare nel comando di richiesta di avvio stream per un certo tipo di misura, inviando una sequenza di 2 B di tipo `0x01-XX` all'UUID del control point, dove il secondo byte indica il tipo di misura a cui ci si riferisce. Come è possibile vedere da Tab. 4.2, per le misure di ECG la sequenza è `0x01-00`, la quale restituisce la sequenza `0xF0-01-00-00-00-00-01-82-00-01-01-0E-00` che indica che si tratta di una risposta (`0xF0`), oltre a ripetere quali informazioni

**PMD Measurement Types**

| Measurement Types | Description | Unit | Static Requirements |
|---|---|---|---|
| 0 | ECG | Volt (V) | Lifetime |
| 1 | PPG | | Lifetime |
| 2 | Acceleration | Force per unit mass (g) | Lifetime |
| 3 | PP Interval | Second (s) | Lifetime |
| 4 | Reserved for Future Use | | Lifetime |
| 5 | Gyroscope | Degrees per second (dps) | Lifetime |
| 6 | Magnetometer | Gauss (G) | Lifetime |
| 7-255 | Reserved for Future Use | Not defined | Not defined |

Tabella 4.2: Tipi di misure definiti nel PMD service e loro identificativi. Si badi che la loro disponibilità varia a seconda del dispositivo usato (Fonte: [32]).

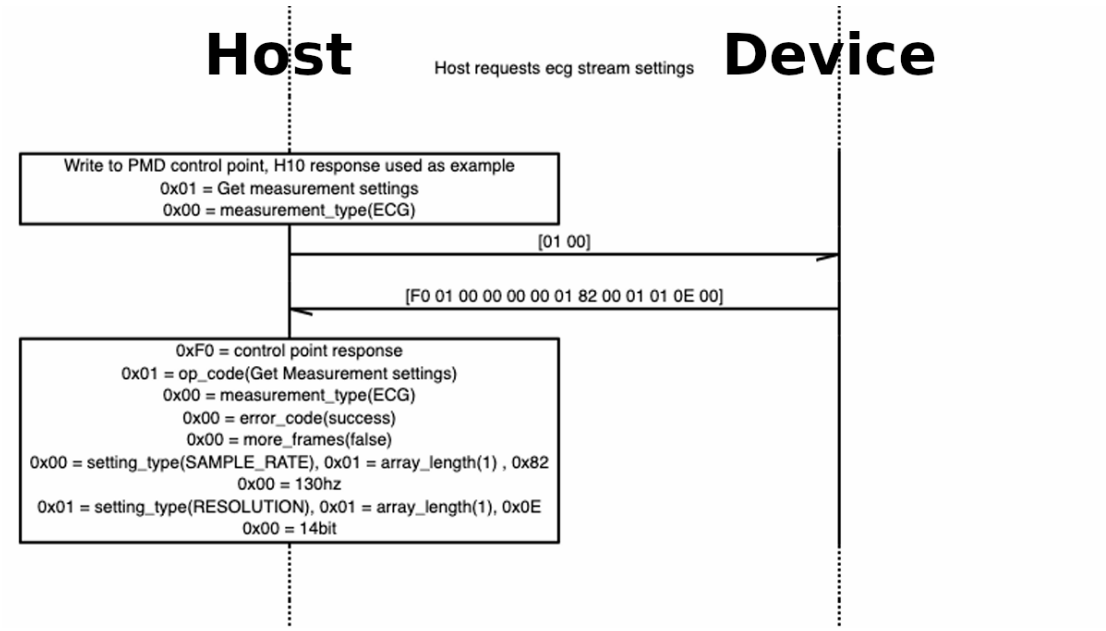



Figura 4.1: Esempio di richiesta di informazioni riguardo le impostazioni per lo stream di dati ECG (Fonte: [32]).

sono state richieste (`0x01-00`) e che non ci sono errori o frame extra (`0x00-00`). Successivamente codifica l'informazione che per l'impostazione del sample rate (`0x00`) è disponibile una sola opzione (`0x01`) ed è pari a `0x82-00`, cioè 130 Hz, come pure per la risoluzione (`0x01`) si ha una sola opzione che corrisponde a `0x0E-00` ovvero 14 bit. Questa descrizione è stata ricavata dallo schema in Fig. 4.1.

Il driver utilizza la libreria `bleak` per la gestione delle connessioni Bluetooth. Per fare questo si sfruttano i metodi `read_gatt_char(UUID)` e `write_gatt_char(UUID,VALUE)` forniti dalla classe `BleakClient`, che rispettivamente leggono o scrivono il valore di una caratteristica identificata per mezzo del suo UUID. In modo simile a quanto visto per il dispositivo EDA, il driver inizializza `bthClient`, un'istanza della classe *BleakClient* fornita dalla libreria, indicando come parametro l'indirizzo fisico del dispositivo. Questo realizza la connessione con il device. Dopodiché si attende la disponibilità del servizio PMD invocando il metodo `read_gatt_char(PMD_CONTROL_SERVICE)` per l'oggetto `bthClient`, dove con `PMD_CONTROL_SERVICE` si indica l'UUID del control point del servizio. Viene successivamente inviato il comando di avvio stream tramite il metodo `write_gatt_char(PMD_CONTROL_SERVICE, WRITE_ECG)`, dove `WRITE_ECG` è definito come l'array di byte `[0x02,0x00,0x00,0x01,0x82,0x00,0x01,0x01,0x0E,0x00]`. Questa sequenza codifica le seguenti informazioni: comando di inizio misurazione (`0x02`), misura di tipo ECG (`0x00`), impostazione della frequenza di campionamento (`0x00`) usando un singolo numero a 16 bit (`0x01`) pari a 130 Hz (`0x82-00`) ed impostazione della risoluzione (`0x01`) usando un numero a 16 bit (`0x01`) con un valore di 14 bit (`0x0E-00`). Si osserva che, sebbene la frequenza di campionamento e la risoluzione della misura siano scelte obbligate nel caso dell'ECG per il sensore Polar H10, esse vanno in ogni caso comunicate in quanto il servizio PMD è pensato per tutta la gamma di sensori con funzionalità di connessione Bluetooth prodotti dalla Polar.

Una volta avviato lo stream dei dati, il nodo si sottoscrive al servizio di notifica del dispositivo invocando il metodo `start_notify(PMD_DATA_MTU,polar_callback)`, dove `PMD_DATA_MTU` indica l'UUID della caratteristica PMD Data MTU (Tab. 4.1) mentre il secondo parametro è il nome della funzione che verrà utilizzata come callback alla ricezione del messaggio. Il messaggio ricevuto dal sensore è una sequenza di byte con la seguente codifica: `0x00-XX-XX-XX-XX-XX-XX-XX-XX-00-YY-YY-YY...`, dove il primo byte sta ad indicare il tipo di misura cioè ECG, i successivi 8 B il timestamp in nanosecondi dell'ultimo campione con-

tenuto nel messaggio, e poi una serie di campioni il cui valore in µV è contenuto in gruppi di 3 B (come da specifica del frame ECG utilizzato). Il codice quindi estrae le informazioni relative al timestamp e ai singoli campioni e li salva in un vettore per poi pubblicarli una volta terminata la lettura del messaggio.

Alla chiusura del nodo viene eseguita la disiscrizione dalle notifiche del dispositivo con il metodo `stop_notify(PMD_DATA_MTU)` e, cosa molto importante, si invia il comando di stop dello stream al dispositivo con un `write_gatt_char` contenente la sequenza `0x03-00`.

Come nel caso del sistema Plux biosignalsplux, il driver implementato è in grado di controllare tutti gli aspetti necessari alla comunicazione tramite Bluetooth con il sensore Polar H10 e di pubblicare i dati rendendoli disponibili su ROS, permettendo di integrare una misura del tracciato ECG nel framework.

# 5 Prove sperimentali

Le performance del framework, in particolare quelle relative al pacchetto di allineamento delle pose, sono state testate attraverso un esperimento presso il laboratorio di dipartimento Ergo-lab, utilizzando un setup che simula l'attività di un lavoratore su una linea di assemblaggio con layout a L seguendo il paradigma del walking-worker. Ciò significa che l'operatore non resta fisso nella propria postazione predefinita attendendo il prodotto da assemblare, ma lo segue lungo tutto il processo di assemblaggio. Questo implica che il lavoratore non esegue solo un sottoinsieme prefissato dei task che compongono l'operazione di assemblaggio, ma svolge tutti i task in ordine dall'inizio alla fine. Questo approccio è considerato vantaggioso per linee produttive in cui la variabilità delle operazioni manuali è elevata e la produzione richiesta potenzialmente molto variabile, in quanto permette di adattare il numero di lavoratori al throughput richiesto senza variare la disposizione della linea. Inoltre permette di rispondere in modo migliore alla variabilità dell'organico, in seguito ad una iniziale inefficienza dovuta alla necessità di insegnare tutti i task della fase di assemblaggio invece di un insieme ridotto delle stesse, tutti i lavoratori della linea sono in grado di portare a termine l'intero processo produttivo [33]. La variabilità stessa delle operazioni da svolgere può essere considerata a sua volta un vantaggio in quanto limita gli effetti dal punto di vista cognitivo che l'esecuzione per lunghi periodi di tempo di pochi task molto ripetitivi hanno sul lavoratore. Gli svantaggi principali che si possono evidenziare sono invece il maggiore livello di affaticamento dal punto di vista fisico dovuto alla necessità dell'operatore di muoversi per seguire il prodotto, oltre alle inefficienze temporali dovute allo spostamento e la presenza di una maggiore curva di apprendimento per imparare tutti i task.

L'operazione svolta da parte di un volontario consiste nell'assemblaggio di un comodino di legno, eseguito ripetutamente in un timeframe di circa 64 min. Poiché ogni ciclo di assemblaggio ha una durata di circa 10 min, in totale sono state eseguite 6 operazioni di assemblaggio complete. Il volontario indossa la tuta Xsens MVN Awinda con l'intero set di 17 sensori per raccogliere i dati del MoCap inerziale, mentre una telecamera Microsoft Azure Kinect lo inquadra posteriormente per ottenere la posa da MoCap markerless. Per quanto riguarda i sensori Polar H10 e Plux biosignalsplux, poiché la valutazione del significato

dei dati ottenuti richiede il parere di esperti nel campo medico ed esula dagli obiettivi del progetto, ci si è limitati a verificare che i driver riuscissero a gestire la comunicazione con i dispositivi e pubblicassero i dati correttamente.

## 5.1 Layout dell'esperimento

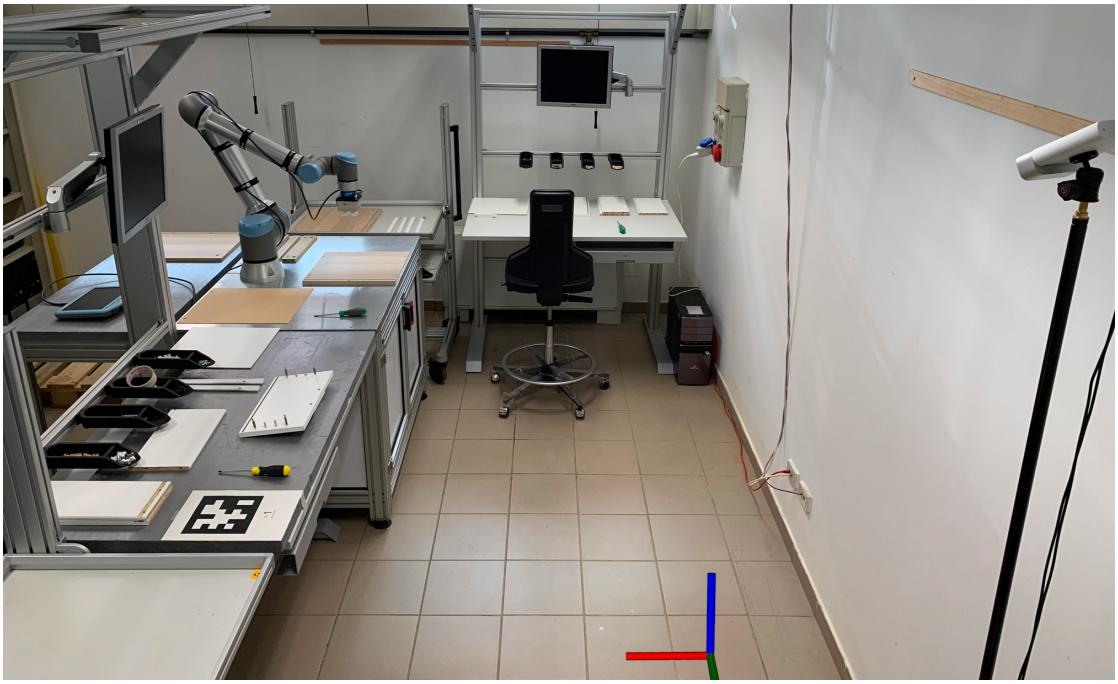


Figura 5.1: Setup sperimentale utilizzato per l'esperimento.

La Fig. 5.1 mostra il layout definito per l'esperimento, formato da quattro stazioni di cui tre utilizzate dal lavoratore in piedi ed una in posizione seduta. La terza stazione include un cobot UR-10e (Universal Robots, Odense - Danimarca) che svolge un task collaborativo in cui aiuta l'operatore fornendogli il materiale dalla vicina unità di storage e tenendo sollevato il prodotto lasciando che questi abbia entrambe le mani libere. La telecamera Kinect è posta in un angolo del setup, da dove è possibile inquadrare posteriormente il lavoratore durante tutta l'operazione di assemblaggio. Tuttavia, le stazioni 1 e 4 presentano una occlusione parziale dell'operatore, rispettivamente di un fianco per la prima e della parte inferiore del corpo per l'altra. Questa scelta è stata fatta proprio per testare la robustezza del sistema di allineamento alle occlusioni, verificando la validità del concetto di fondere dati inerziali e markerless per risolvere le problematiche di drift dei primi sfruttando le informazioni fornite dai secondi anche quando queste non sono completamente o continuamente disponibili. Ciò

ne dimostra anche la potenzialità per usi industriali, dove le occlusioni sono un fenomeno molto probabile data la presenza di layout già definiti e non modificabili che non permettono un posizionamento ideale della telecamera. L'sdr in figura rappresenta il riferimento globale ricavato tramite la procedura di calibrazione della Kinect mediante un AprilTag. Ad esso è stato accuratamente allineato l'sdr Xsens prima dell'inizio dell'esperimento sfruttanto la procedura fornita dal software Xsens MVN Analyze, in modo tale da evitare la presenza di bias nei dati raccolti e permettere così di mostrare l'effettivo fenomeno di drift di cui soffre la tecnologia. È inoltre presente un secondo AprilTag tra la prima e la seconda postazione, utilizzato per una procedura in cui si è chiesto al volontario di appoggiarvi sopra la mano sinistra prima di iniziare il task della postazione 2, per valutare l'efficacia dell'allineamento anche per estremità dello scheletro grazie all'inclusione nell'elenco dei keypoint da allineare degli appositi landmark presenti sul polso.

## 5.2 Analisi dei dati

Durante l'esperimento è stata raccolta una bag di ROS, salvando gli scheletri di entrambe le forme di MoCap assieme a quello ottenuto dal processo di allineamento per poterli confrontare. Sono state inoltre registrate le TF per poter valutare l'entità del drift che affligge il sistema inerziale per periodi di tempo medio-lunghi. Come valore di $w$ da utilizzare nella media pesata è stato scelto 0.95, considerato un buon compromesso tra filtraggio del rumore e velocità di allineamento degli scheletri.

### 5.2.1 Effetto del drift sulle misure

Come è possibile osservare in Fig. 5.2, la posizione del centroide dello scheletro ottenuto dal MoCap inerziale va alla deriva sul piano $xy$ sia in termini di posizione che di rotazione rispetto all'sdr globale definito all'inizio dell'esperimento. Nel complesso, il fenomeno di drifting dello scheletro inerziale è talmente intenso da rappresentare una differenza considerevole tra la posizione reale e quella stimata già nella fase finale del primo ciclo di assemblaggio, ovvero nei primi 10 min. Questo fatto è confermato anche dalla Fig. 5.3, che mostra come all'aumentare del tempo il keypoint corrispondente al polso sinistro dell'operatore è affetto da un drift continuo, visibile addirittura nei pochi minuti in cui il lavoratore è posizionato nelle stazioni e si limita a compiere movimenti con

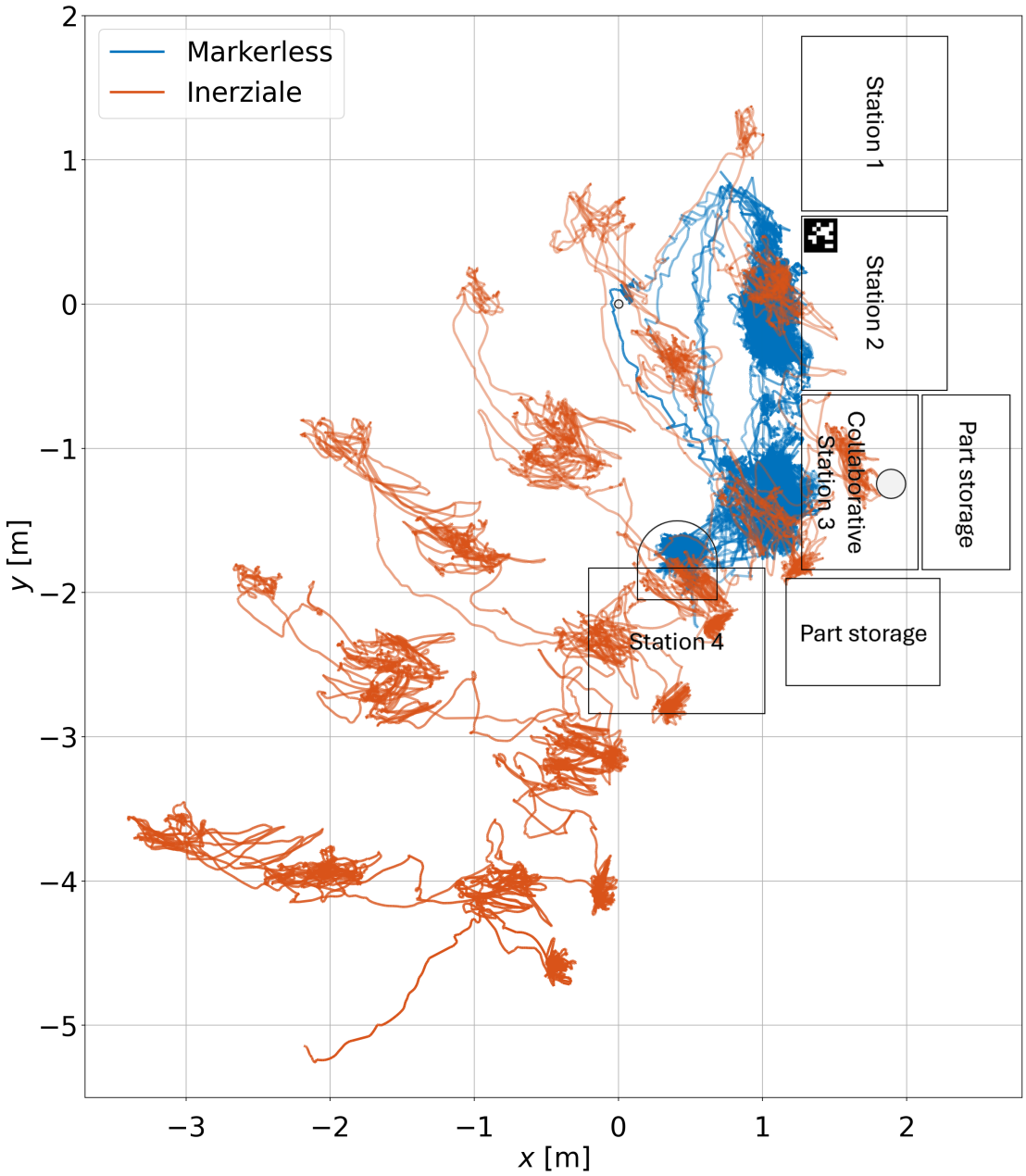



Figura 5.2: Spaghetti chart della posizione del centroide nel piano $xy$. L'opacità del tratto aumenta al progredire del tempo.

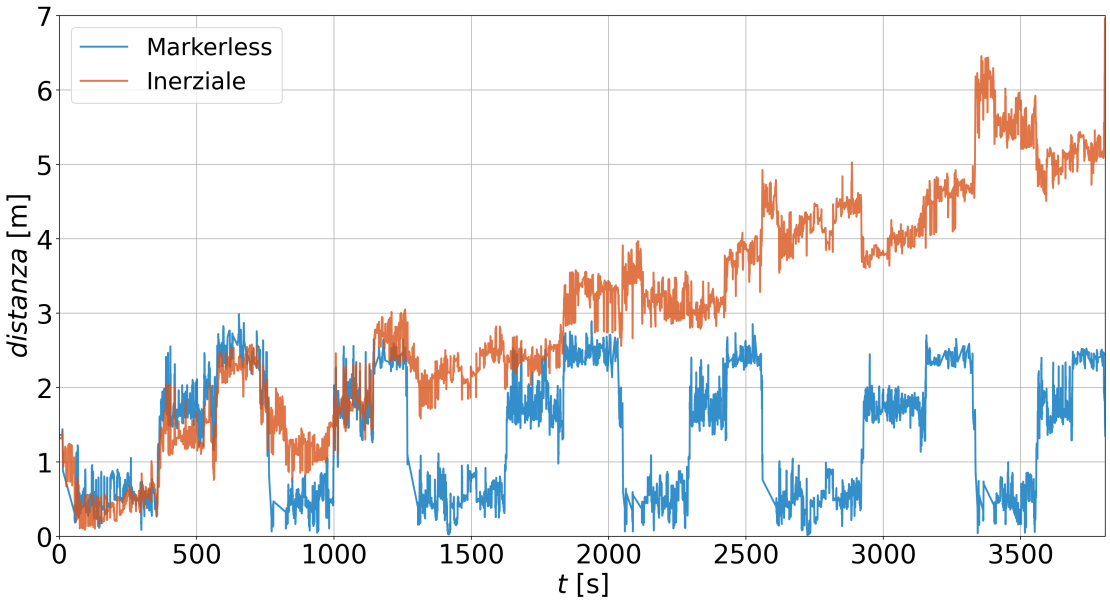



Figura 5.3: Distanza euclidea tra il punto individuato dall'AprilTag nella postazione 2 e il polso sinistro dell'operatore.

gli arti superiori per portare avanti le operazioni di assemblaggio. Questo è un ulteriore indice di come tale fenomeno succeda costantemente e gradualmente, a causa dell'origine dello stesso dovuta all'integrazione di rumore nelle misure accelerometriche dei sensori che non viene corretto. È chiaro come sia necessario il ricorso ad un metodo per correggere tale errore, permettendo di riallineare i dati e rendendo così il MoCap inerziale effettivamente fruibile all'interno di contesti industriali.

### 5.2.2 Effetto dell'allineamento sulle misure

Sfruttando il pacchetto `skeleton_aligner` la situazione cambia considerevolmente. È infatti immediatamente visibile dalla Fig. 5.4 come il sistema riesca a correggere in modo più che adeguato la posizione sul piano $xy$, anche nelle zone come la postazione 1 dove vi è un'importante effetto di occlusione della telecamera RGB-D. Anche dal punto di vista della distanza polso-AprilTag questa segue molto bene il profilo del caso markerless (Fig. 5.5), riuscendo a mantenere una posizione coerente persino nei tratti in cui manca l'informazione fornita dalla telecamera, come è possibile osservare nella parte sinistra del grafico in Fig. 5.5b.

Per quanto riguarda la quantificazione del drift che affligge il sistema inerziale, essa può essere ricavata come le distanze traslazionale e rotazionale dell'sdr

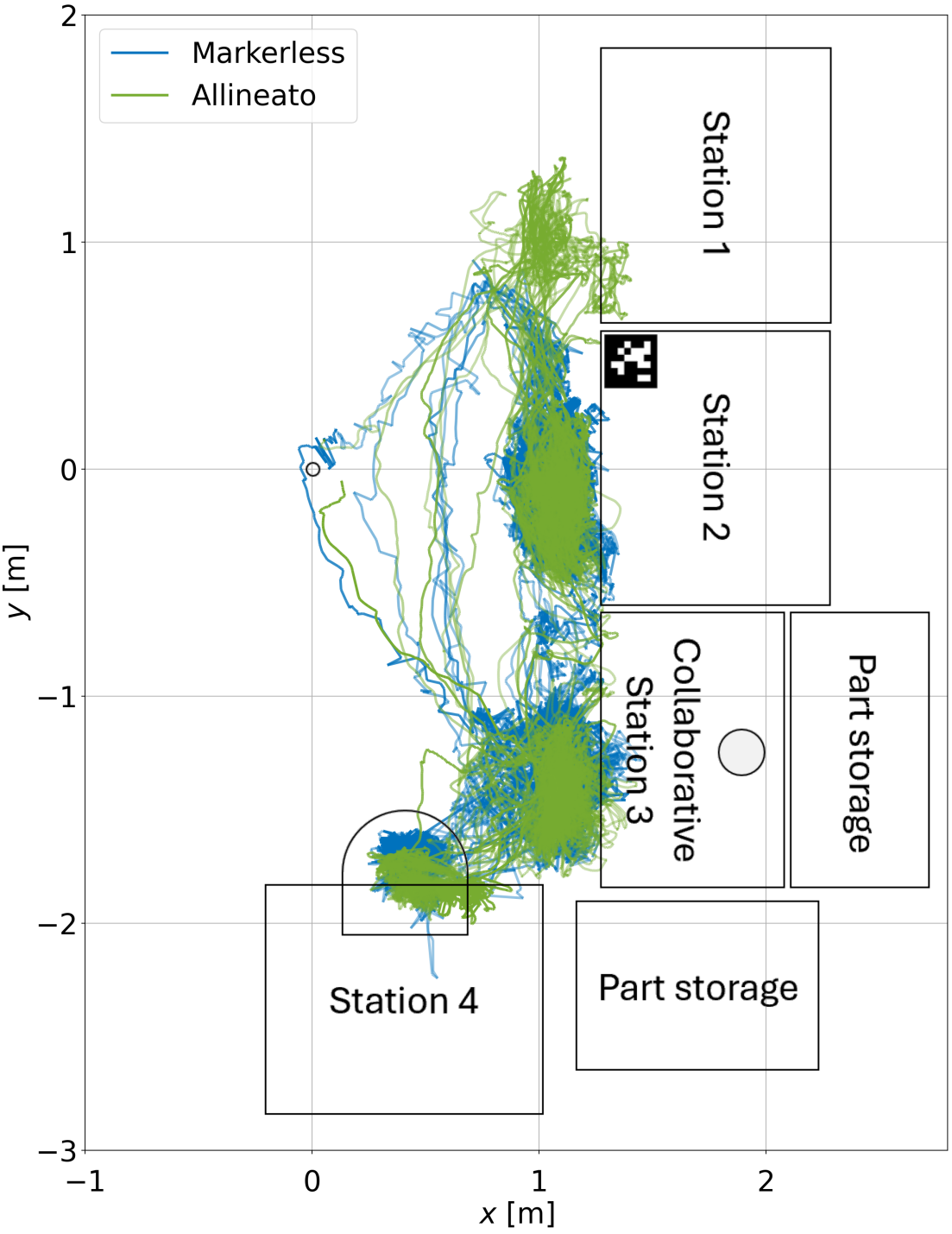
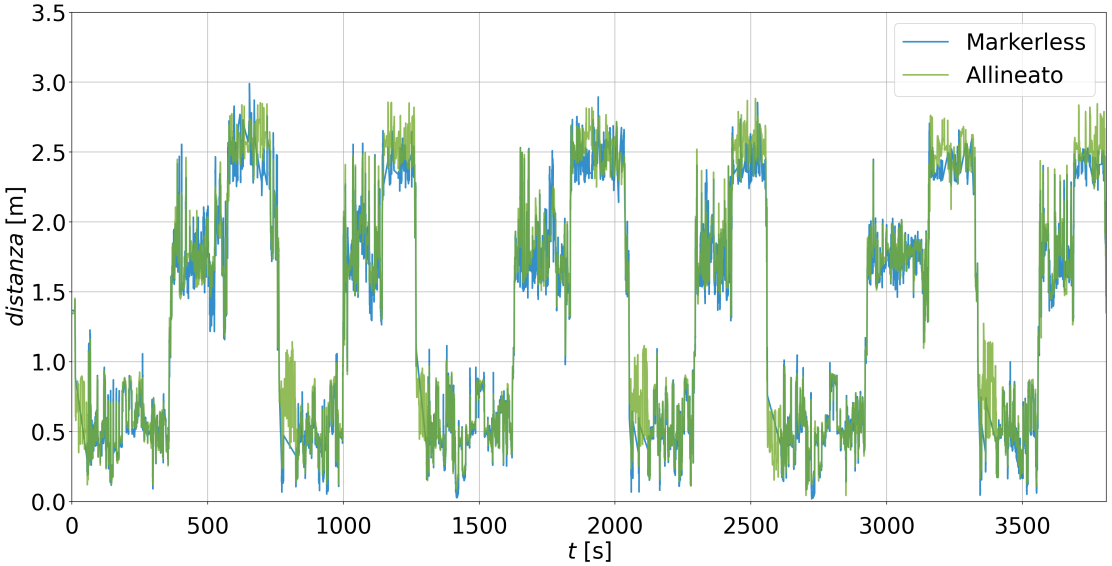
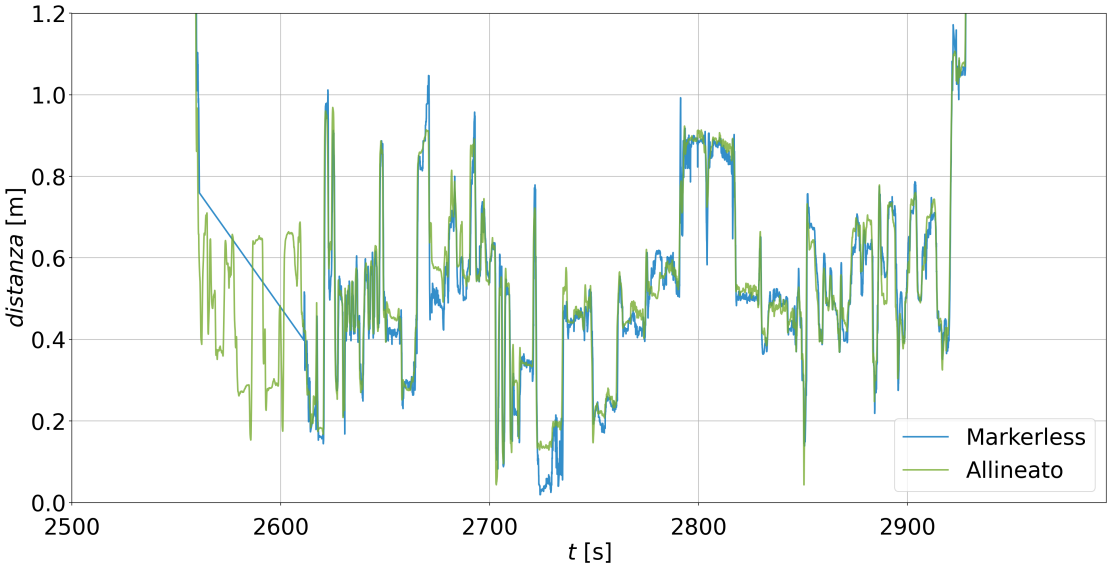



Figura 5.4: Spaghetti chart della posizione del centroide nel piano $xy$. L'opacità del tratto aumenta al progredire del tempo.

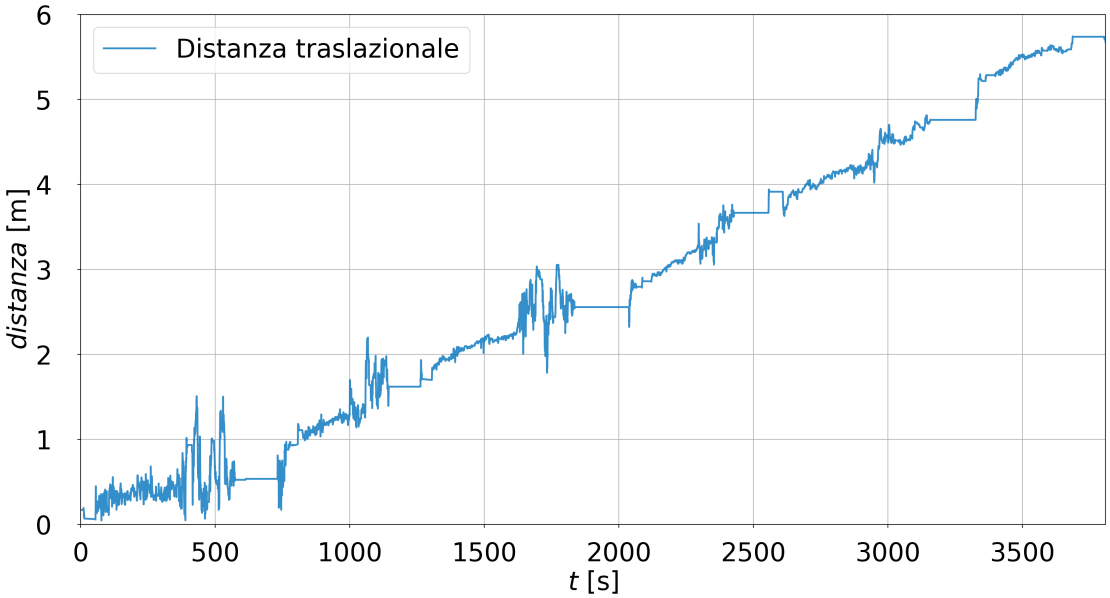



(a) Esperimento completo.

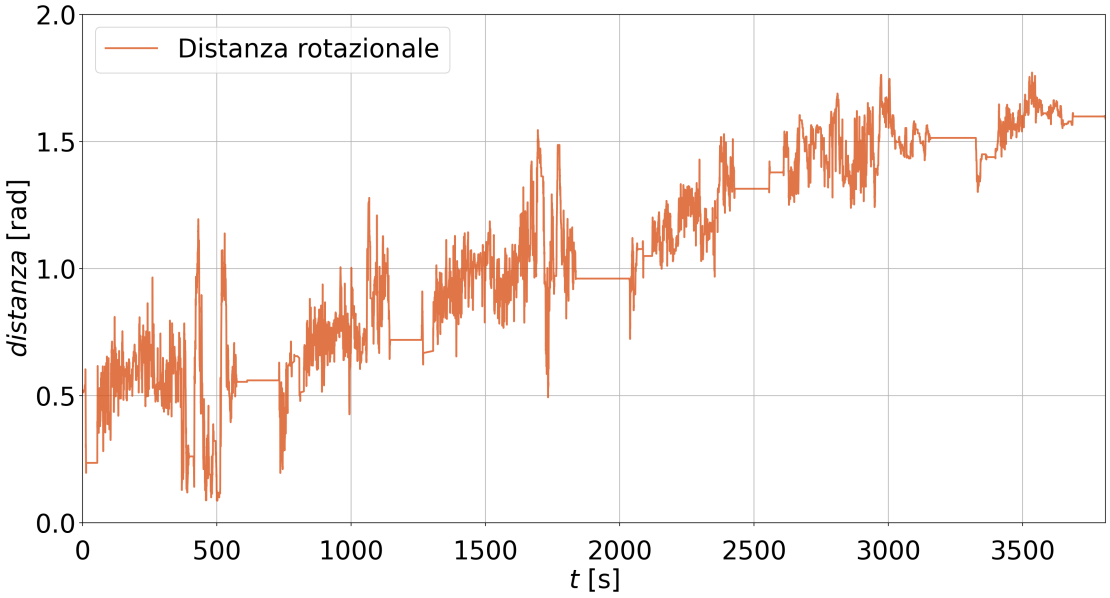



(b) Dettaglio.

Figura 5.5: Distanza euclidea tra il punto individuato dall'AprilTag nella postazione 2 e il polso sinistro dell'operatore




(a) Distanza traslazionale.




(b) Distanza rotazionale.

Figura 5.6: Distanza dell'sdr inerziale rispetto a quello globale ottenuta a partire dalla TF che li lega.

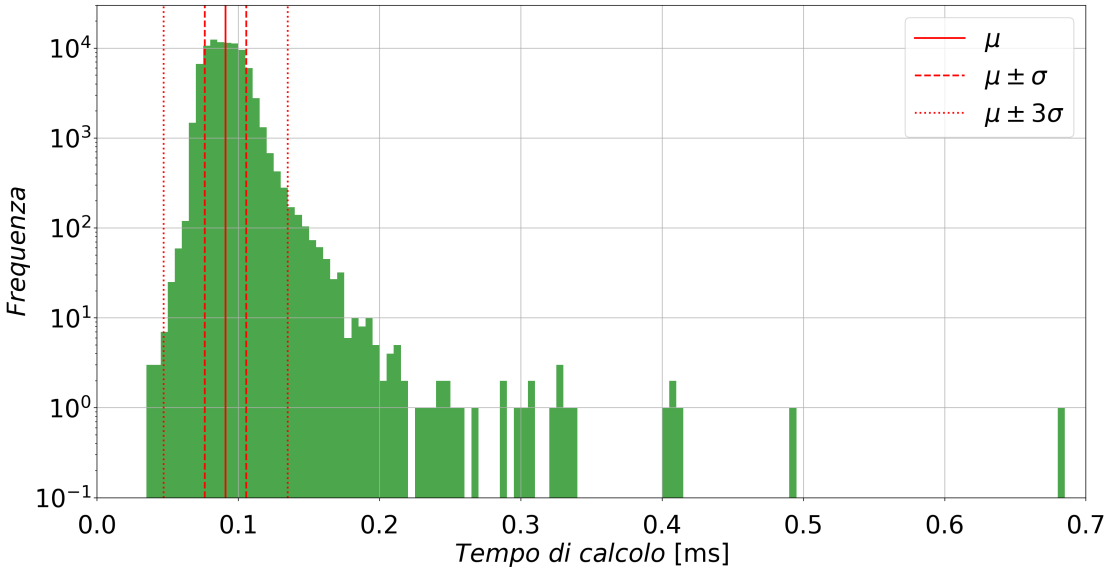



Figura 5.7: Istogramma delle frequenze delle misure di tempo di risposta dell'algoritmo di allineamento. Si faccia attenzione alla scala logaritmica.

xsens rispetto a quello globale, che corrispondono alle distanze euclidea ed angolare della TF (ottenuta dalla procedura di allineamento) rispetto alla trasformazione identità. Come possibile vedere dalla Fig. 5.6, esse sono altamente variabili nel tempo, raggiungendo valori massimi rispettivamente di 5.74 m e 1.77 rad, coerenti con quanto visto nel diagramma in Fig. 5.2. Applicando un semplice fit lineare a questi vettori di distanze si ottengono dei trend di variazione pari a 1.5 mm/s e 0.3 mrad/s. È inoltre possibile osservare che le distanze restano costanti per tratti relativamente lunghi, e confrontandoli con la Fig. 5.5b attorno al sec 1300 si può intuire che essi corrispondono ai periodi di tempo in cui il MoCap markerless non è in grado di fornire una posa adeguata dello scheletro a causa dei fenomeni di occlusione. In tale evenienza il sistema continua a funzionare, ipotizzando come costante l'ultima TF calcolata. Questa ipotesi ovviamente può considerarsi valida solamente se il drift ha entità sufficientemente piccola e/o le interruzioni sono di durata limitata.

### 5.2.3 Performance computazionali

Ricordando che tra i requisiti del progetto figura anche quello di avere performance tali da poter permettere il funzionamento online, si è deciso di calcolare il tempo di risposta del suo componente più complesso, ovvero il pacchetto `skeleton_aligner`. Tale metrica è stata calcolata come il tempo passato tra

l'ottenimento degli scheletri e la restituzione della corrispondente TF. Come possibile osservare in Fig. 5.7, il tempo di calcolo si attesta intorno al valore di $(0.091 \pm 0.015)$ ms, un ottimo risultato ottenuto grazie agli step di ottimizzazione descritti nel Cap. 3. Dal grafico si può notare come siano presenti degli outlier nella parte destra, con situazioni in cui si arriva fino a circa 0.7 m sec, tuttavia è bene notare che è stato scelto di rappresentare il grafico con una scala semilogaritmica e che questi casi vadano dal singolo evento (quasi tutti i valori superiori a 0.2 ms) fino a poco meno di un centinaio di istanze per ciascun intervallo, a fronte di valori centrali della curva rappresentati da svariate migliaia di eventi. Pertanto, essi si possono considerare a tutti gli effetti come dei casi estremamente poco probabili, oltre che di durata trascurabile rispetto alle tempistiche di pubblicazione e ricezione dei dati, che non vanno ad impattare sulle performance generali del programma.

Il computer utilizzato per svolgere l'esperimento è equipaggiato con una CPU Intel Core i7-12700H operante a 2.30 GHz ed un totale di 32 GB di RAM.

# 6 Conclusioni

Questo lavoro di tesi si sviluppa all'interno del contesto definito dal paradigma dell'Industria 5.0, che mira a massimizzare la sostenibilità, la resilienza e l'antropocentrismo della produzione industriale. Tra gli argomenti studiati dalla ricerca in merito, il concetto di HDT appare come un candidato ideale per garantire una migliore integrazione dei lavoratori nei sistemi di produzione, potenzialmente permettendo una costante valutazione dello stato e del benessere fisico e cognitivo dell'operatore. Queste informazioni potrebbero essere utilizzate per informare sia l'operatore sia il sistema di controllo di una postazione di lavoro collaborativa di nuova generazione, capace di adattarsi in modo molto più efficiente e completo all'elemento umano e permettere in questo modo una maggiore inclusività nei luoghi di lavoro del futuro.

Per poter realizzare un HDT è necessario in primo luogo sviluppare un sistema in grado di raccogliere in tempo reale le informazioni da sistemi di sensori potenzialmente molto differenti l'uno dall'altro, di elaborarle ove necessario ed infine di renderle disponibili in modo strutturato ed accessibile alla futura implementazione dell'HDT che ne farebbe uso. Tale sistema è il framework multimodale realizzato in questo lavoro di tesi, dove l'aggettivo sta ad indicarne la caratteristica principale, ovvero quella di permettere l'integrazione di molteplici sensori per la misura di grandezze di tipo ergonomico e fisiologico utili alla definizione dello stato di benessere di un'operatore dell'industria manifatturiera.

Dati i requisiti di compatibilità con una futura evoluzione in un HDT, di utilizzo di tecnologie sensoristiche eterogenee e di performance adeguate alla raccolta in tempo reale di dati, è stato scelto di sfruttare le potenzialità fornite da ROS per la realizzazione di sistemi di calcolo distribuito e la gestione della comunicazione tra i nodi di una rete del genere. Questo ha reso possibile l'integrazione di quattro tipologie di sensori differenti all'interno del framework: una telecamera RGB-D per la raccolta di dati relativi alla stima della posa (Body Pose Estimation, BPE) di una persona mediante MoCap markerless, una tuta di sensori IMU capace di fornire la BPE tramite informazioni di tipo inerziale, un sensore ECG indossato a livello toracico per la misura del tracciato elettrocardiografico o del battito cardiaco ed infine un sensore EDA per la misurazione della

risposta galvanica della pelle ed una possibile stima del tasso di sudorazione della persona.

La scelta di questi sistemi di sensori è stata fatta per permettere la raccolta di dati che non fossero limitati alla sola componente ergonomica, considerata necessaria per la realizzazione di un HDT per la valutazione del benessere in ambito industriale, ma anche di segnali fisiologici che potessero rappresentare lo stato cognitivo della persona dal punto di vista del livello di attività, affaticamento e stress. In particolar modo è stato deciso di ricorrere a due tipologie di MoCap differenti per poterne compensare le rispettive limitazioni implementando un sistema di allineamento delle pose che esse producono. Il MoCap inerziale infatti presenta ottime performance di posizionamento relativo dei giunti nello spazio ma soffre di problemi di drifting che ne riducono l'accuratezza nel posizionamento assoluto su lunghi periodi di tempo, mentre il MoCap inerziale fornisce generalmente un'informazione accurata circa il posizionamento assoluto della persona ma è soggetto a fenomeni di occlusione che riducono la disponibilità di dati.

Sono stati realizzati dei pacchetti ROS aventi il ruolo di driver per gestire la comunicazione con il sensore ECG Polar H10 e con il sistema Plux biosignalsplux utilizzato con sensori EDA. Tali driver realizzano la connessione ai dispositivi mediante Bluetooth, avviano e/o arrestano la lettura e lo stream dei dati da parte del dispositivo, leggono i dati e li convertono in un formato adeguato ad essere pubblicato in appositi topic nella rete di comunicazione generata da ROS, dove sono quindi disponibili alla lettura da parte di tutti i nodi attivi nella stessa rete.

Dal momento che l'analisi di queste tipologie di dati richiede competenze in ambito medico e di interfacciarsi con personale esperto in materia, l'implementazione di questi pacchetti è stata volutamente limitata alle funzionalità necessarie alla gestione della comunicazione. L'obiettivo principale del framework risulta infatti quello di dimostrare la realizzabilità del concetto di raccolta e presentazione real-time di dati multimodali attraverso una singola entità concettuale. Invece, la ricerca di come utilizzare questi (ed eventualmente altri) dati e ricavarne una rappresentazione significativa dello stato di benessere di un lavoratore nel contesto dell'industria manifatturiera appartengono alla futura evoluzione di questo concetto ed al suo utilizzo nello sviluppo di un HDT tramite il quale realizzare una postazione di lavoro collaborativa avanzata. Questa sarebbe in grado di integrare in modo migliore l'elemento umano nei sistemi produttivi, andando a beneficio delle condizioni di salute dello stesso senza tuttavia

comprometterne la produttività.

Il risultato principale del progetto è stato lo sviluppo di un pacchetto ROS chiamato `skeleton_aligner` che permette di svolgere la fusione dei dati di stima della posa (rappresentati come insiemi di giunti interconnessi nello spazio definiti come "scheletri") provenienti dalla telecamera RGB-D e di quelli misurati dal sistema di sensori inerziali, allo scopo di calcolare e tenere sempre aggiornata la trasformazione che esprime nel sistema di riferimento globale la posizione ed orientazione dell'sdr in cui sono espressi gli scheletri inerziali.

Implementando un algoritmo di allineamento basato sull'Analisi di Procuste è stato possibile calcolare la trasformazione che allinea lo scheletro inerziale a quello markerless dopo ciascun aggiornamento degli stessi da parte dei rispettivi sistemi di misura. Implementando un buffer facente uso di un sistema di clustering non supervisionato viene eseguita la separazione delle trasformate in gruppi basata sulla distanza relativa tra le stesse, permettendo di scartare gli outlier dovuti ad effetti come il rumore o l'incertezza causata da occlusioni parziali. Una volta individuato il cluster principale, viene eseguita una media pesata ai minimi quadrati delle trasformazioni sfruttando un algoritmo che non fa perdere di significato la componente rotazionale. Questo algoritmo è stato inoltre ottimizzato per ridurre considerevolmente l'occupazione di memoria richiesta dai dati intermedi ed il numero di operazioni di calcolo, riuscendo nell'obiettivo di migliorare le prestazioni in accordo con i requisiti derivanti dalla possibile applicazione industriale.

Il risultato finale è la stima di una TF che permette in ogni istante di tempo di allineare il sistema di riferimento in cui sono espressi gli scheletri del MoCap inerziale a quello globale ottenuto dalla procedura di calibrazione del MoCap markerless, permettendo di sfruttare al meglio l'accuratezza della stima della posa inerziale in contemporanea alla stabilità nel posizionamento assoluto della posa markerless, risolvendo le problematiche di drifting della prima tecnologia e quelle di occlusione della seconda.

Una prova sperimentale di lunga durata volta a testare il funzionamento del sistema per applicazioni su intervalli di tempo medio-lunghi ha dimostrato la validità del sistema per l'uso nella correzione online degli effetti di deriva che affliggono tipicamente il MoCap ottenuto tramite sensori inerziali, mostrando la capacità di riallineare lo scheletro inerziale a quello markerless e di sfruttare in modo adeguato tali informazioni anche nei periodi in cui la stima proveniente dalla telecamera viene a mancare per effetto delle occlusioni. Anche dal punto

di vista del tempo di calcolo il sistema presenta delle performance più che soddisfacenti, mostrando un tempo di allineamento mediamente al di sotto dei 100 µs e pertanto non impattante sulla pubblicazione dei risultati tramite nodi ROS (operazioni che impiegano un tempo nell'ordine dei ms).

Muovendosi esattamente lungo questa direzione, parte del lavoro descritto in questa tesi è stato incluso in un articolo di prossima pubblicazione nel quale si descrive una struttura concettuale che estende quella del framework sviluppato durante il progetto, ponendo le basi per lo sviluppo di un HDT avente le caratteristiche descritte nella parte introduttiva della tesi [34]. La presenza di un attivo lavoro di ricerca in merito a questo argomento fornisce un'ulteriore prova della necessità di avere a disposizione un sistema come il framework multimodale sviluppato in questo progetto, in grado di raccogliere dati relativi allo stato di benessere psicofisico, risolvere le problematiche della stima della posa inerziale e markerless, e rendere disponibili tali dati in tempo reale per analisi volte alla stima dello stato di benessere psicofisico dei lavoratori dell'industria manifatturiera. Tra gli sviluppi futuri presentati nell'articolo risulta quello di sfruttare le capacità fornite dai metodi di Machine Learning per separare l'uso dell'HDT in una fase di pre-deployment in cui si utilizza l'intero set di sensori per generare un dataset con il quale addestrare un modello personalizzato sull'operatore. Tale modello permetterebbe, durante la fase operativa vera e propria del sistema, di stimare lo stato fisico e cognitivo del lavoratore a partire da un sottoinsieme limitato della sensoristica utilizzata in precedenza, aumentando il comfort per l'utente durante lo svolgimento dei propri task senza per questo ridurre le prestazioni del sistema.

# Bibliografia


[1] X. Xu, Y. Lu, B. Vogel-Heuser e L. Wang, «Industry 4.0 and Industry 5.0 — Inception, conception and perception,» *Journal of Manufacturing Systems*, vol. 61, pp. 530–535, 2021, ISSN: 0278-6125. DOI: 10.1016/j.jmsy.2021.10.006.

[2] J. Leng, W. Sha, B. Wang et al., «Industry 5.0: Prospect and retrospect,» *Journal of Manufacturing Systems*, vol. 65, pp. 279–295, 2022, ISSN: 0278-6125. DOI: 10.1016/j.jmsy.2022.09.017.

[3] E. L. X. Li Da Xu e L. Li, «Industry 4.0: state of the art and future trends,» *International Journal of Production Research*, vol. 56, n. 8, pp. 2941–2962, 2018. DOI: 10.1080/00207543.2018.1444806.

[4] L. Li, B. Lei e C. Mao, «Digital twin in smart manufacturing,» *Journal of Industrial Information Integration*, vol. 26, p. 100 289, 2022, ISSN: 2452-414X. DOI: 10.1016/j.jii.2021.100289.

[5] S. Huang, B. Wang, X. Li, P. Zheng, D. Mourtzis e L. Wang, «Industry 5.0 and Society 5.0 — Comparison, complementation and co-evolution,» *Journal of Manufacturing Systems*, vol. 64, pp. 424–428, 2022, ISSN: 0278-6125. DOI: 10.1016/j.jmsy.2022.07.010.

[6] C. Zhang, Z. Wang, G. Zhou et al., «Towards new-generation human-centric smart manufacturing in Industry 5.0: A systematic review,» *Advanced Engineering Informatics*, vol. 57, p. 102 121, 2023, ISSN: 1474-0346. DOI: 10.1016/j.aei.2023.102121.

[7] J. Zhou, Y. Zhou, B. Wang e J. Zang, «Human–Cyber–Physical Systems (HCPSs) in the Context of New-Generation Intelligent Manufacturing,» *Engineering*, vol. 5, n. 4, pp. 624–636, 2019, ISSN: 2095-8099. DOI: 10.1016/j.eng.2019.07.015.

[8] M. Miller e E. Spatz, «A unified view of a human digital twin,» *Human-Intelligent Systems Integration*, vol. 4, giu. 2022. DOI: 10.1007/s42454-022-00041-x.

[9] B. Wang, H. Zhou, X. Li et al., «Human Digital Twin in the context of Industry 5.0,» *Robotics and Computer-Integrated Manufacturing*, vol. 85, p. 102 626, 2024, ISSN: 0736-5845. DOI: 10.1016/j.rcim.2023.102626.

[10] European Agency for Safety and Health at Work, *Summary - Occupational safety and health in Europe: state and trends 2023 | Safety and health at work EU-OSHA*, osha.europa.eu, mag. 2023. indirizzo: https://osha.europa.eu/en/publications/summary-occupational-safety-and-health-europe-state-and-trends-2023 (visitato il 07/08/2024).

[11] M. Lorenzini, M. Lagomarsino, L. Fortini, S. Gholami e A. Ajoudani, «Ergonomic human-robot collaboration in industry: A review,» *Frontiers in Robotics and AI*, vol. 9, 2023. DOI: 10.3389/frobt.2022.813907.

[12] D. Battini, N. Berti, S. Finco, M. Guidolin, M. Reggiani e L. Tagliapietra, «WEM-Platform: A real-time platform for full-body ergonomic assessment and feedback in manufacturing and logistics systems,» *Computers & Industrial Engineering*, vol. 164, p. 107 881, 2022, ISSN: 0360-8352. DOI: 10.1016/j.cie.2021.107881.

[13] N. Berti e S. Finco, «Digital Twin and Human Factors in Manufacturing and Logistics Systems: State of the Art and Future Research Directions,» *IFAC-PapersOnLine*, vol. 55, n. 10, pp. 1893–1898, 2022, 10th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2022, ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2022.09.675.

[14] M. Quigley, K. Conley, B. Gerkey et al., «ROS: an open-source Robot Operating System,» 3.2, Kobe, Japan, vol. 3, gen. 2009, p. 5. indirizzo: https://ai.stanford.edu/~mquigley/papers/icra2009-ros.pdf (visitato il 14/08/2024).

[15] Open Source Robotics Foundation, Inc, *2023 ROS Metrics Report*, ROS Discourse, gen. 2024. indirizzo: https://discourse.ros.org/t/2023-ros-metrics-report/35837 (visitato il 14/08/2024).

[16] S. Macenski, T. Foote, B. Gerkey, C. Lalancette e W. Woodall, «Robot operating system 2: Design, architecture, and uses in the wild,» *Science robotics*, vol. 7, n. 66, 2022. DOI: 10.1126/scirobotics.abm6074.

[17] F. Zehra, M. Javed, D. Khan e M. Pasha, «Comparative Analysis of C++ and Python in Terms of Memory and Time,» *Preprints*, dic. 2020. DOI: 10.20944/preprints202012.0516.v1.

[18] *Azure Kinect DK*, Microsoft Store. indirizzo: https://www.microsoft.com/en-us/d/azure-kinect-dk/8pp5vxmd9nhq (visitato il 14/08/2024).

[19] *Xsens MVN Awinda*, Movella, 2014. indirizzo: https://shop.movella.com/product-lines/motion-capture/products/xsens-mvn-awinda (visitato il 14/08/2024).

[20] *Polar H10*, Polar.com, 2024. indirizzo: https://www.polar.com/it/sensors/h10-heart-rate-sensor/ (visitato il 14/08/2024).

[21] *4-Channel biosignalsplux Kit*, PLUX Biosignals, 2024. indirizzo: https://www.pluxbiosignals.com/collections/shop/products/4-channel-biosignals-kit (visitato il 14/08/2024).

[22] *Electrodermal Activity Sensor*, PLUX Biosignals, 2024. indirizzo: https://www.pluxbiosignals.com/collections/shop/products/electrodermal-activity-eda-sensor-1 (visitato il 14/08/2024).

[23] A. Ross, «Procrustes analysis,» *Course report, Department of Computer Science and Engineering, University of South Carolina*, vol. 26, pp. 1–8, 2004.

[24] M. Guidolin, L. Tagliapietra, E. Menegatti e M. Reggiani, «Hi-ROS: Open-source multi-camera sensor fusion for real-time people tracking,» *Computer Vision and Image Understanding*, vol. 232, p. 103 694, 2023, ISSN: 1077-3142. DOI: 10.1016/j.cviu.2023.103694.

[25] *AprilTag*, april.eecs.umich.edu. indirizzo: https://april.eecs.umich.edu/software/apriltag (visitato il 19/08/2024).

[26] *Azure Kinect body tracking joints*, learn.microsoft.com, set. 2022. indirizzo: https://learn.microsoft.com/en-us/azure/kinect-dk/body-joints (visitato il 19/08/2024).

[27] *MVN User Manual*, Revision Z. Movella Technologies B.V., mag. 2024. indirizzo: https://www.movella.com/hubfs/MVN_User_Manual.pdf (visitato il 19/08/2024).

[28] J. Wang e E. Olson, «AprilTag 2: Efficient and robust fiducial detection,» in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 4193–4198. DOI: 10.1109/IROS.2016.7759617.

[29] J. C. Gower e G. B. Dijksterhuis, *Procrustes Problems*. Oxford University Press, gen. 2004, ISBN: 9780198510581. DOI: 10.1093/acprof:oso/9780198510581.001.0001.

[30] *Eigen*, eigen.tuxfamily.org. indirizzo: https://eigen.tuxfamily.org/index.php?title=Main_Page (visitato il 24/08/2024).

[31] J. Lawrence, J. Bernal e C. Witzgall, «A Purely Algebraic Justification of the Kabsch-Umeyama Algorithm,» *Journal of Research of the National Institute of Standards and Technology*, vol. 124, ott. 2019. DOI: 10.6028/jres.124.028.

[32] polarofficial, *Polar Measurement Data Specification*, GitHub, 2019. indirizzo: https://github.com/polarofficial/polar-ble-sdk/blob/39c809/technical_documentation/Polar_Measurement_Data_Specification.pdf (visitato il 21/08/2024).

[33] I. Zennaro, M. Calzavara, M. Faccio e A. Persona, «Consideration of the learning effect in the comparison of Fixed Worker and Walking Worker assembly systems,» *IFAC-PapersOnLine*, vol. 56, n. 2, pp. 707–712, 2023, 22nd IFAC World Congress, ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2023.10.1649.

[34] M. Guidolin, N. Berti, P. Gnesotto, D. Battini e M. Reggiani, «Trust the Robot! Enabling Flexible Collaboration With Humans via Multi-Sensor Data Integration,» in *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*, [accepted], IEEE, 2024.