

UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Ingegneria dell'Informazione ed Elettrica
e Matematica Applicata



Tesi di Laurea Magistrale in INGEGNERIA INFORMATICA

Controllo Decentralizzato di manipolatori cooperanti in presenza di operatori umani

Relatore

Prof. Alessandro Marino

Candidato

Ines Sorrentino

Matr. 0622700419

Correlatore

Prof. Pasquale Chiacchio

Anno Accademico 2016/2017

Alla mia famiglia

e a tutti quelli che mi hanno sostenuto

in questo percorso...

Indice

Introduzione	1
1 La robotica e l'Industria 4.0	5
1.1 La quarta rivoluzione industriale	5
1.2 Sistemi multi robot	12
1.3 Sistemi centralizzati/decentralizzati	15
2 Elementi teorici	18
2.1 Teoria algebrica dei grafi	18
2.1.1 Definizioni di base e connettività del grafo	19
2.1.2 Descrizione dei grafi tramite matrici	21
2.1.3 Autostruttura della matrice Laplaciana	21
2.2 Il problema del consenso	23
2.2.1 Definizione di consenso	23
2.2.2 Problema del consenso su grafi diretti	23
2.2.3 Consenso medio su digrafi	25
2.3 Notazione	26
2.4 Modello del singolo robot	27
2.5 Modello del sistema	31
2.6 Decomposizione delle forze	31
2.7 Dinamica dell'oggetto	36
3 Schemi di controllo	38
3.1 Notazione	39
3.2 Funzione di task globale	40

3.2.1	Baricentro e formazione	41
3.3	Algoritmo di controllo decentralizzato	42
3.3.1	I livello. Osservatore globale	43
3.3.2	II livello. Legge di controllo adattativa locale	46
3.3.3	Controllo distribuito puramente cinematico	48
3.3.4	Controllo distribuito di interazione	50
4	Simulazione e risultati	55
4.1	Comau Smart-Six	55
4.2	Simulazione in ambiente Matlab	59
4.2.1	Task desiderato	63
4.2.2	Implementazione dell'algoritmo di controllo	66
4.2.3	Risultati ottenuti	68
4.2.4	Risultati sul controllo puramente cinematico	68
4.2.5	Risultati sul controllo di interazione	74
5	Eperimenti	79
5.1	Descrizione dello scenario	82
5.1.1	Robot UR10	83
5.2	Robot Operating System	84
5.3	Kinect	85
5.3.1	Calibrazione kinect	85
5.3.2	Detection e Tracking dello skeleton	89
5.4	Architettura del sistema	90
5.5	Pianificazione del moto del robot UR10	93
5.5.1	Orientamento desiderato	98
5.5.2	Posizione desiderata	103
5.6	Task cooperativo	106
5.7	Risultati ottenuti	108
Conclusioni e sviluppi futuri		117

Introduzione

La robotica è oggi una realtà sempre più vicina alla nostra vita quotidiana, al punto che si può affermare che le macchine possono svolgere operazioni in totale autonomia. Mentre fino a pochi anni fa i robot erano principalmente impiegati nella fabbrica automatica, dove la macchina sostituiva l'uomo, oggi assistiamo a tutta una serie di innovazioni evolutive grazie alle quali il braccio meccanico diventa un robot in grado di cooperare con l'uomo e capace di operare in maniera sicura anche in ambienti incerti, utilizzando strutture dotate di capacità sensoriali avanzate. Oggi si parla di robotica avanzata destinata a coinvolgere diverse discipline e la sostanziale novità sta nel fatto che, agevolmente, i robot, nati inizialmente per eseguire compiti pericolosi, ripetitivi e di precisione, si stanno avvicinando all'ambiente umano. Infatti se fino al recente passato il paradigma universalmente adottato per la robotica industriale prevedeva la rigida segregazione dei robot in ambienti protetti per mezzo di infrastrutture fisiche, oggi emerge con sempre maggiore forza la consapevolezza dei vantaggi che si possono ottenere da una collaborazione diretta tra uomo e robot. Vi sono infatti operazioni troppo complesse per poter essere eseguite dal robot e viceversa operazioni ripetitive in cui la precisione e l'affidabilità del robot non possono essere egualiate dall'uomo. Avere quindi la possibilità di condividere gli spazi tra uomo e robot, e in una certa misura anche di farli collaborare nell'esecuzione di sequenze di operazioni, è considerato uno degli obiettivi più importanti della fabbrica del futuro. Facendo riferimento a tipiche operazioni di assemblaggio di parti, la robotica collaborativa si colloca in una fascia intermedia tra assemblaggi completamente automatizzati, tipici degli scenari di produzione con grandi lotti e piccole varianti, e gli assemblaggi completamente manuali, tipici degli scenari con piccoli lotti di produzione.

ed elevato numero di varianti. Questa evoluzione porta ad avere la necessità di robot molto più flessibili per adattarsi a task che cambiano frequentemente (è stato detto che nel passato i robot venivano introdotti nelle fabbriche per svolgere sempre lo stesso lavoro), robot robusti rispetto ai guasti ed inoltre in grado di svolgere task molto più complessi rispetto a quelli richiesti dalla vecchia industria, e spesso in collaborazione tra loro.

In vista di questo cambio di paradigma industriale, sempre più studi vengono svolti sulla possibilità di avere collaborazione tra l'uomo e le macchine, così come vengono svolti sulla possibilità di utilizzare robot di tipologie diverse che si adattano allo stato attuale della fabbrica e hanno la possibilità di svolgere un task comune in modo cooperativo. Tutto questo si vuole che avvenga senza la presenza di un'unità centrale, ma in modo decentralizzato, con scambio di informazioni tra robot vicini o appartenenti al gruppo cooperante. La necessità di un'industria decentralizzata nasce infatti dal fatto che l'Industry 4.0, termine con cui si indica il radicale cambiamento di paradigma che il settore manifatturiero sta affrontando in questi ultimi anni grazie alla diffusione delle tecnologie digitali e alla loro integrazione nella filiera produttiva, prevede che la fabbrica non sia progettata per svolgere un determinato insieme di attività in modo continuo e statico, ma che questa e i robot al suo interno si adattino al tipo di produzione attuale, ovvero al prodotto che in un determinato istante si chiede di realizzare. Questo significa che i robot devono essere dinamici e non programmati staticamente, devono saper infatti svolgere task di diversa tipologia, devono saper collaborare e coordinarsi con altri robot e considerando team composti da un diverso numero di robot per applicazione e devono sapersi muovere all'interno di ambienti non noti in partenza. Tutto questo è reso più semplice dall'utilizzo di un'architettura decentralizzata, in quanto ci sono diversi vantaggi. Ad esempio non è necessario che un robot conosca lo stato di tutti gli altri all'interno della fabbrica, ma solo lo stato del team con cui sta cooperando. Questo rende inutile connettere tutti i robot ad un'unità centrale potendo evitare così di dover realizzare un'infrastruttura di comunicazione onerosa. Inoltre un'architettura decentralizzata presenta, complessivamente, un notevole vantaggio in termini di scalabilità e robustezza ai

guasti rispetto al caso centralizzato.

Il seguente lavoro di tesi si colloca nel quadro appena descritto e prevede infatti, per quanto riguarda la prima parte di lavoro, la realizzazione di un'architettura decentralizzata per il coordinamento di robot mobili ciascuno dei quali è un'entità del sistema che esegue autonomamente un algoritmo di controllo. L'obiettivo di questo algoritmo è stabilire quali azioni ogni robot deve compiere per ottenere il comportamento complessivo desiderato il quale dipende dallo stato e dal comportamento di tutti gli altri robot. La cooperazione tra team di robot permette l'esecuzione di task complessi che possono essere di vario genere. Negli ambienti industriali infatti un tipico esempio è l'assemblaggio su larga scala. L'algoritmo sviluppato si presta a qualsiasi attività di manipolazione cooperante sia su vasta scala che su scala ridotta. L'algoritmo di controllo e l'architettura sono implementati e simulati in Matlab e la visualizzazione del task portato a termine dai robot è stata possibile attraverso un tool chiamato V-REP, il quale permette di mettere in piedi diversi scenari rappresentativi della realtà e visualizzare le simulazioni effettuate in Matlab.

Una seconda fase del lavoro di tesi ha riguardato l'implementazione reale di uno scenario di interazione uomo-robot che sfrutta quanto sviluppato nella fase precedente, ovvero la cooperazione tra robot per la manipolazione di oggetti e inoltre aggiunge la presenza di un robot di supervisione e operatori umani che svolgono altre attività. Lo scopo è avere contemporaneamente negli stessi ambienti robot che svolgono determinati task di cooperazione, robot di supervisione e persone che possono invadere questi spazi liberamente. Questo scenario ha l'obiettivo di simulare quanto accadrà con la nuova visione di fabbrica, in quanto, come già detto precedentemente, si vuole la collaborazione tra macchine, robot e operatori umani. Anche se non avviene stretta collaborazione, ovvero macchine e umani svolgono lavori differenti, può succedere che questi siano presenti all'interno della stessa cella di lavoro, potendosi influenzare ed ostacolare a vicenda. Per questo motivo lo scenario considerato, messo in piedi nel Laboratorio di Automatica dell'Università degli Studi di Salerno, prevede all'interno di una stessa cella di

lavoro, la presenza sia di due robot Comau Smart-Six che svolgono un task di cooperazione, sia di operatori umani, sia del robot UR10 che ha il compito di supervisionare la cella per accorgersi della presenza degli operatori e di conseguenza di modificare il task eseguito dai robot Comau in base a quanto l'uomo si sta avvicinando ad essi, modificando anche il proprio comportamento.

Capitolo 1

La robotica e l'Industria 4.0

I robot, negli ultimi anni, sono sempre più utilizzati e continuano a diffondersi sia in ambiente industriale che in ambiente domestico. Inoltre, il prezzo sempre più basso e le ridotte dimensioni di un hardware relativamente complesso, hanno portato da un lato alla miniaturizzazione dei robot, dall'altro alla possibilità di iniziare a studiare sistemi complessi composti da un numero sempre maggiore di robot, spesso mobili, in grado di cooperare tra loro per portare a termine un determinato compito, come il trasporto cooperativo, l'esplorazione o compiti di ricerca e soccorso in ambienti ostili all'uomo. Tutta questa evoluzione si deve perlopiù ad una rivoluzione molto più vasta chiamata Industry 4.0 (o quarta rivoluzione industriale), la quale ha lo scopo di far crescere la quantità e la qualità delle produzioni industriali unendo nuove tecnologie e infrastrutture informatiche.

1.1 La quarta rivoluzione industriale

Il termine Industry 4.0 è stato introdotto per la prima volta nel 2011 presso la Fiera di Hannover in Germania ed indica un processo di trasformazione ed evoluzione che porterà ad una produzione industriale del tutto automatizzata e interconnessa. Tale rivoluzione ha assunto notevole importanza negli ultimi anni, tanto da essere considerata una rivoluzione industriale al pari delle precedenti e ciò è evidenziato anche dalla figura sottostante (Figura 1.1).

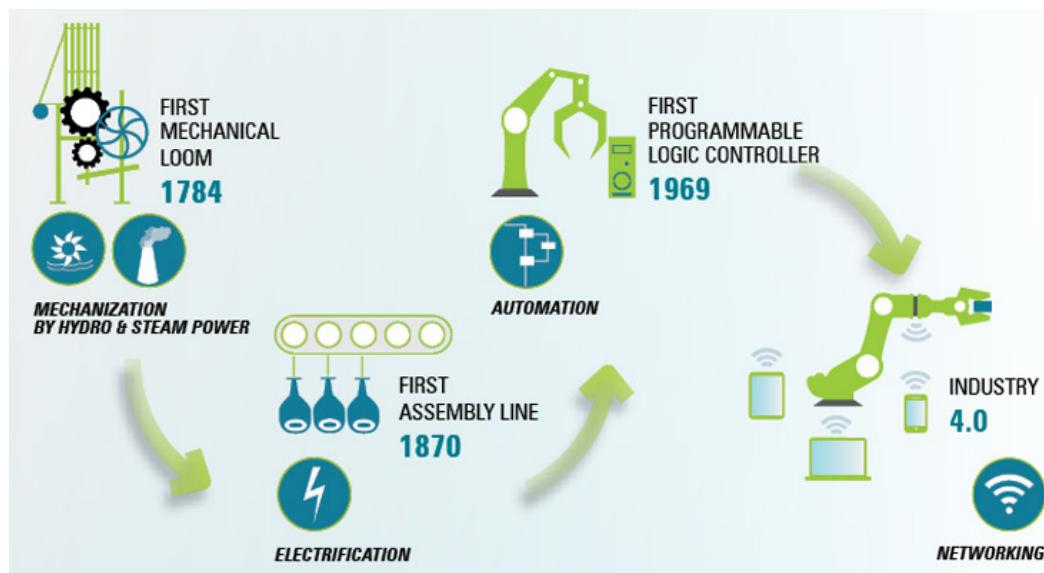


Figura 1.1: Le quattro rivoluzioni industriali

Fino a qualche anno fa e in alcuni casi tutt'oggi, la maggior parte degli impianti di produzione era controllata in modo centralizzato e gerarchico e i componenti degli impianti erano dotati di scarsa intelligenza. L'idea alla base della quarta rivoluzione industriale è pertanto, quella di cambiare il modo di intendere l'industria, migrando l'attenzione dal processo produttivo al prodotto. Sarà quest'ultimo a conoscere quali sono gli step necessari al proprio completamento e per questo sarà in grado di richiedere alle varie macchine di produzione il servizio di cui man mano necessita al fine di completare la sua fabbricazione. Questo cambio di paradigma fa sì che le fabbriche diventino delle *Smart Factory* con sistemi di produzione *Cyber Fisici* (CPS) [10]. I CPS sono sistemi innovativi i quali prevedono che i prodotti fisici e la loro rappresentazione nel mondo digitale siano strettamente integrati. Si nota dunque un passaggio radicale da sistemi centralizzati a sistemi decentralizzati, in quanto non esiste più un'unità centrale che si occupa di gestire l'intera fabbrica, ma il processo di produzione è specializzato per ogni prodotto. In sostanza le industrie smart consentono un approccio nuovo alla produzione: ogni singolo prodotto è identificabile nello spazio e nel tempo, adeguabile in tempo reale alle esigenze del cliente, tracciabile a partire dalla materia prima, fino alla consegna.

I principali vantaggi ottenibili dalla rivoluzione sono i seguenti:

- capacità di adattamento alle necessità dei clienti;
- riduzione dei costi energetici grazie ad una maggiore efficienza e allo smart control degli impianti;
- tempi di decision making e di risposta ridotti e ottimizzati grazie alla disponibilità di informazioni provenienti dal tracciamento continuo del singolo prodotto e dall'intelligenza fornita al prodotto stesso;
- maggiore flessibilità, grazie alla struttura più dinamica dei processi industriali.

La quarta rivoluzione industriale prevede quindi di interconnettere sensori, macchine e prodotti per creare i già citati CPS, i quali possono anche interagire tra loro utilizzando protocolli standard di Internet e analizzando dati per predire fallimenti, per riconfigurarsi e per adattarsi a cambiamenti sulla base dello stato attuale del sistema. Tale logica è descritta dalla figura 1.2.

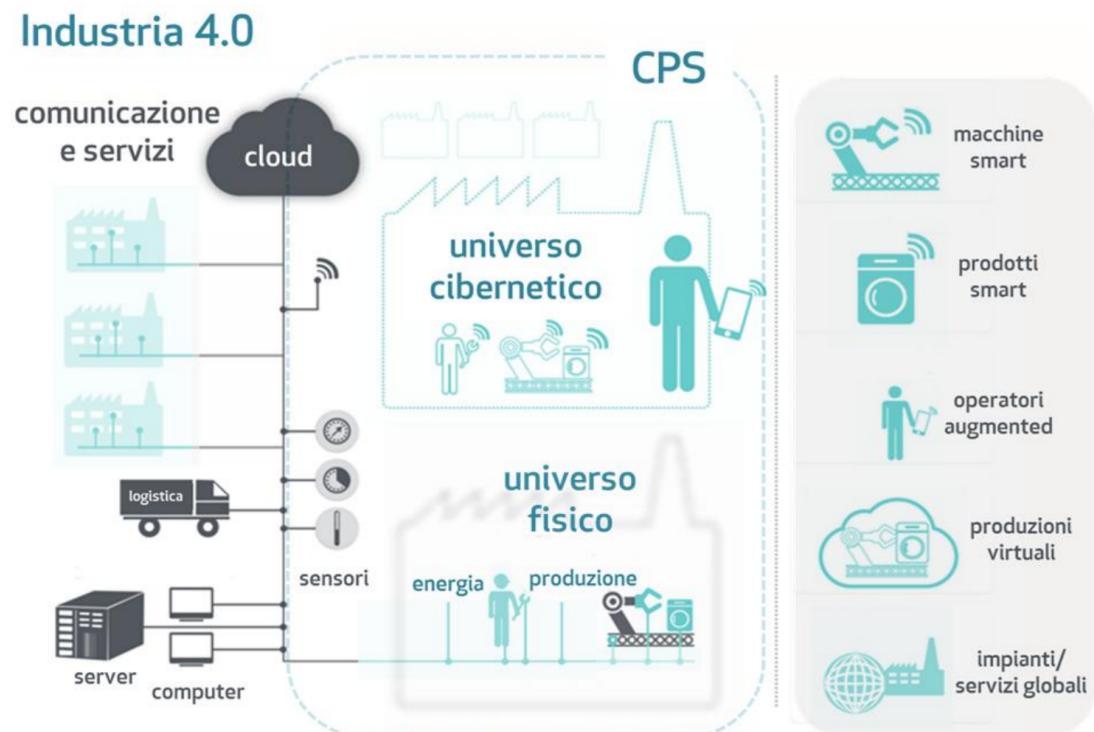


Figura 1.2: Rappresentazione della nuova industria

La trasformazione che porterà ad ottenere un'industria totalmente rinnovata, si fonda su nove progressi tecnologici di seguito analizzati e mostrati in Figura 1.3 [11].

- Big Data and Analytics: nella nuova industria ci si basa su decisioni real-time prese a partire dall'analisi di una grande mole di dati provenienti da diverse fonti e diverse macchine (come sensori, algoritmi di apprendimento automatico e gli stessi sistemi di produzione) al fine di automatizzare i processi industriali e sviluppare un insieme di regole per gestire ogni possibile evento.
- Autonomous Robots: i produttori di molti settori hanno da tempo utilizzato robot per affrontare compiti complessi, ma i robot si stanno evolvendo per un'utilità ancora maggiore. Infatti stanno diventando più autonomi, flessibili e cooperativi; interagiranno tra loro e lavoreranno in sicurezza fianco a fianco con gli umani riuscendo anche ad imparare da loro. Questi robot costeranno meno e avranno una gamma più ampia di funzionalità rispetto a quelli utilizzati oggi nella produzione.
- Simulazione: i simulatori di materiali, prodotti e processi produttivi, sono già diffusi, ma in futuro tali simulatori avranno un ruolo diverso, infatti mostreranno il mondo fisico aggiornato in real-time e questo permetterà agli operatori di testare e ottimizzare le impostazioni delle macchine in simulazione e solo in seguito nell'ambiente reale.
- Integrazione di sistema orizzontale e verticale: integrazione orizzontale significa networking tra macchine e parti di impianto o unità di produzione mentre l'integrazione verticale va al di là della tradizionale gerarchia dei livelli di produzione coprendo dunque un insieme di elementi molto più vasto, che può andare dal sensore fino al livello business dell'azienda.
- Industrial IoT: l'espressione IoT descrive uno scenario (già in parte concreto) in cui ogni oggetto che usiamo quotidianamente può diventare intelligente ("smart", cioè con capacità di auto identificazione, localizzazio-

ne, auto-diagnosi dello stato, acquisizione dati, elaborazione, attuazione) e connesso tramite protocolli di comunicazione standard.

- Cybersecurity: con l'aumento della connettività e l'uso dei protocolli di comunicazione standard nell'Industry 4.0, la necessità di proteggere i sistemi industriali critici e le linee di produzione dalle minacce, aumenta notevolmente. Di conseguenza, sono essenziali comunicazioni sicure e affidabili, nonché una sofisticata gestione degli accessi.
- Cloud: grazie alla quarta rivoluzione industriale la necessità di condivisione dei dati all'interno della stessa azienda o oltre i confini aziendali aumenta sempre di più. Per questo motivo il cloud assumerà sempre più importanza e sarà sempre più ottimizzato in modo da ottenere tempi di reazione molto brevi per l'accesso ai dati e per la condivisione, e inoltre offrirà sempre più funzionalità che non saranno più implementate su macchine fisiche ma saranno virtualizzate. Tutto questo dà la possibilità di gestire l'hardware e le risorse dinamicamente in modo da non avere la necessità di richiedere una quantità di hardware sempre maggiore con l'aumento delle richieste e dei servizi.
- Produzione additiva: la produzione additiva, come la stampa 3D, viene utilizzata nell'Industry 4.0, sia per generare prototipi di prodotti sia per avere la possibilità di personalizzarli.
- Realtà aumentata: tale tecnologia apporterà alle imprese notevoli benefici ottimizzando la realizzazione dei prodotti. Un esempio di come la realtà aumentata può essere applicata è il seguente: molti operatori sono costretti a ricorrere alle istruzioni per riparare una macchina, sia che esse siano cartacee sia che siano digitali. La realtà aumentata può rendere il procedimento molto più semplice e rapido, permettendo all'addetto alla riparazione ad esempio di indossare un paio di occhiali per proiettare direttamente le istruzioni sulla macchina e far vedere esattamente dove intervenire.

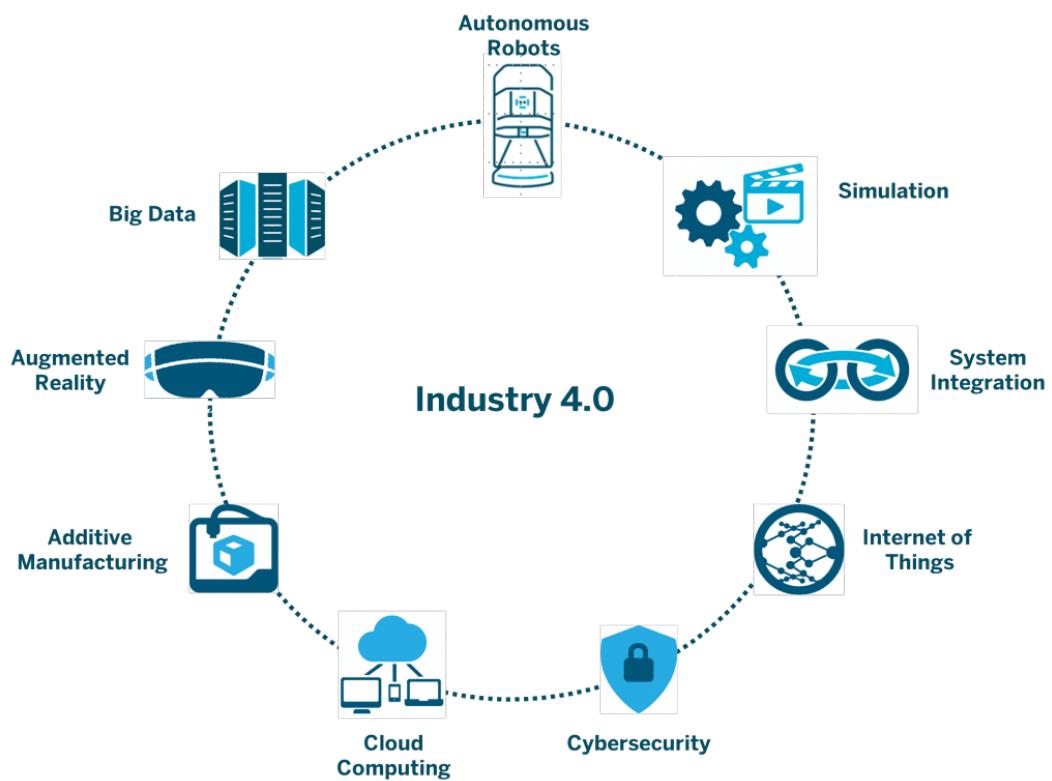


Figura 1.3: Le nove tecnologie che stanno trasformando la produzione industriale

È stato detto che l'Industry 4.0 prevede sistemi di automazione “auto-organizzati”, composti da prodotti e risorse produttive intelligenti ed autonomi che interagiscono all’interno di architetture decentralizzate. I sistemi autonomi sono in grado di identificare dinamicamente la strategia ottimale di controllo del sistema produttivo, massimizzandone le performance in ciascuna condizione operativa. Decentralizzazione, autonomia e negoziazione automatizzata dei servizi costituiranno quindi pilastri fondamentali dei sistemi di automazione e controllo per garantire flessibilità, robustezza, efficienza produttiva ed ottimizzazione dei consumi energetici. Per superare i limiti delle odierne tecniche di controllo basate su regole predeterminate e codificate dal progettista del sistema di automazione, le quali si dimostrano altamente inefficaci nella gestione di processi industriali caratterizzati da variabilità o articolati su molteplici unità produttive, se ne introdurranno delle nuove. Inoltre, al fine di compensare disturbi o deviazioni dovute a fattori contingenti o comunque non sufficientemente predibili, l’azione di controllo e l’azione di ottimizzazione verranno applicate iterativamente e con continuità nel

tempo, ricalcolando automaticamente la migliore azione di compensazione.

Manca infine da sottolineare il ruolo dei robot nell'Industry 4.0. Nella nuova visione industriale i robot assistono gli operatori umani, ovvero le macchine non sostituiscono più l'uomo ma ne completano le capacità svolgendo i lavori più pesanti e complessi che l'uomo non è in grado di svolgere, come ad esempio il sollevamento di carichi pesanti, il raggiungimento di zone non facilmente accessibili. Dunque non esiste più la distinzione dei posti di lavoro tra automatizzati e manuali, ma l'uomo e i robot collaborano per svolgere il proprio ruolo in modo ottimale. Scompaiono dunque anche tutte quelle strutture di recinzione che rendevano sicure le celle di lavoro in cui i robot erano immersi. I nuovi robot per essere adatti a questi nuovi lavori vengono dotati di tecnologie intelligenti e sensoristica e tecnologie software più avanzate. Essi assumono flessibilità in termini di mobilità e in termini di capacità di esecuzione di attività diversificate, in modo da poter essere impiegati individualmente ovunque e per qualsiasi esigenza di produzione. Un esempio di robot progettato e realizzato per svolgere questo nuovo tipo di lavoro è il KUKA LBR iiwa. Questo robot ad esempio è in grado di stare a stretto contatto con un operatore umano rendendo sicura la collaborazione. Infatti è dotato di sensoristica per la precisione nell'assemblaggio e, in caso di contatto accidentale con l'operatore umano, è in grado di riconoscerlo subito e di ridurre immediatamente la forza e la velocità per non danneggiarlo [2].

Chiaramente i vantaggi della collaborazione non si limitano a un unico settore industriale: le soluzioni robotiche collaborative offrono benefici praticamente in ogni ambito. In primo luogo, la collaborazione consente di automatizzare processi che ancora richiedono la presenza di addetti umani e che non possono essere completamente automatizzati con le tecnologie esistenti. In secondo luogo l'automazione pienamente collaborativa senza restrizioni ai fini della sicurezza e in spazi ristretti agevola le installazioni in fabbrica negli spazi esistenti. Inoltre la programmazione in ambito collaborativo può essere effettuata "addestrando" il robot invece di utilizzare un linguaggio in codice. Gli operatori possono comunque utilizzare un linguaggio di programmazione ma la possibilità di interagire con il

robot mostrandogli come muoversi dal punto A al punto B consente di addestrare la macchina in pochi minuti invece che ore. In terzo luogo, i robot collaborativi nobilitano il lavoro delle persone, che possono dedicarsi ad attività più concettuali e fisicamente meno gravose. La collaborazione fra uomini e robot spesso si traduce in livelli di precisione e velocità superiori a quelli raggiungibili dai soli esseri umani, e quindi prodotti di qualità superiore e meno sprechi. Riassumendo con l'arrivo dei robot collaborativi riconosciamo il fatto che uomini e robot hanno ciascuno i propri punti di forza e, quando lavoreranno insieme e in modo sicuro, i luoghi di lavoro saranno più flessibili, forniranno prodotti di qualità migliore, garantiranno maggiore sicurezza, miglioreranno la qualità di vita dei lavoratori.

1.2 Sistemi multi robot

I sistemi multi robot sono stati introdotti nel mondo industriale in quanto il loro utilizzo può offrire diversi vantaggi ai processi di produzione. I principali aspetti che portano a preferire sistemi multi robot a sistemi a singolo robot, sono i seguenti:

- presa un'attività da portare a termine questa può essere spesso decomposta in diverse attività più semplici che i sistemi multi robot possono svolgere in parallelo riducendo il tempo di esecuzione totale e spesso riducendo il costo computazionale;
- utilizzando un sistema multi robot è possibile progettare robot diversi e specializzati, possibilmente più semplici e meno costosi di un singolo agente complesso;
- in molti casi si possono ottenere sistemi molto più robusti e affidabili a fronte di guasti.

Ovviamente i sistemi multi robot non introducono solo vantaggi ma anche diverse problematiche come:

- il coordinamento di un elevato numero di robot;

- la gestione di agenti che possono essere molto diversi tra loro;
- la possibilità di immagazzinare informazioni localmente e non in modo centralizzato;
- la gestione delle comunicazioni tra i robot.

I sistemi multi robot sono adatti a diversi domini applicativi che richiedono l'esecuzione di compiti molto complessi. Esempi di applicazioni di questi sistemi sono: assemblaggio su larga scala, il raggiungimento di ambienti ostili come ad esempio un luogo incendiato, esplorazione di ambienti inaccessibili all'uomo (come l'esplorazione di superfici planetarie) o attività di ricerca e soccorso su vasta scala [5]. Di grande interesse è la valutazione e lo studio delle possibili modalità di comunicazione e coordinamento dei robot all'interno del sistema. Per poter fare una classificazione facciamo riferimento alla Figura 1.4 che mostra l'architettura gerarchica dei sistemi multi robot e i livelli di cui si compone vengono analizzati di seguito.

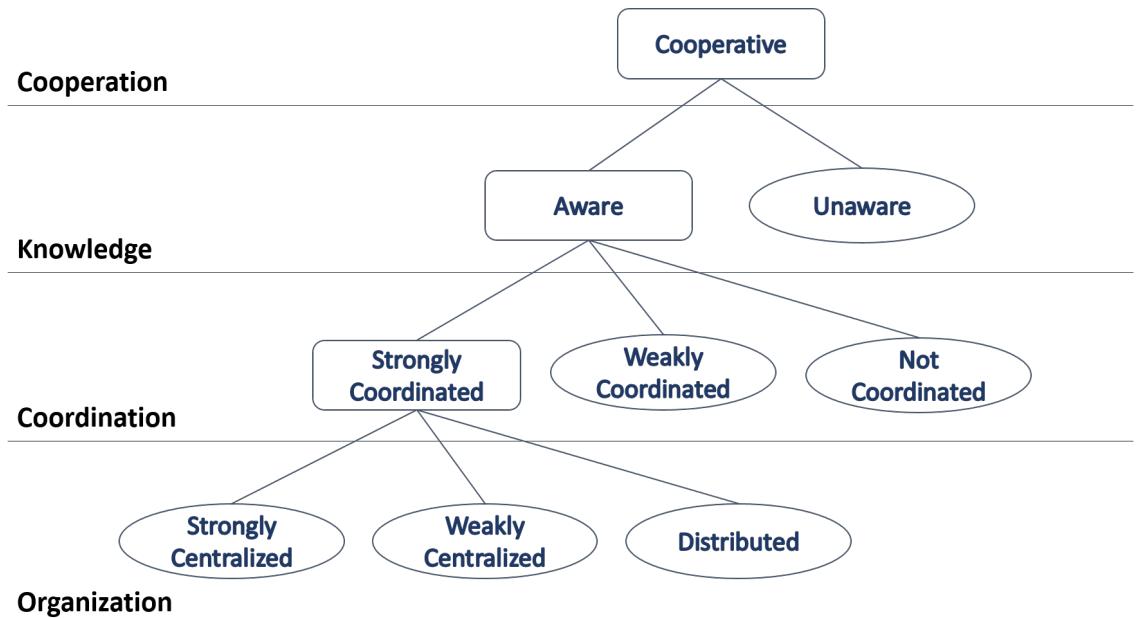


Figura 1.4: Architettura gerarchica dei sistemi multi robot

- *Cooperation:* il primo livello riguarda la capacità del sistema di cooperare al fine di svolgere un compito specifico. A livello di cooperazione, distinguiamo

i sistemi cooperativi da quelli non cooperativi. Un sistema cooperativo è composto da robot che operano insieme per svolgere compiti globali. In questo lavoro di tesi siamo interessati solo ai sistemi multi robot cooperativi.

- *Knowledge*: il secondo livello della gerarchia riguarda la conoscenza che ciascun robot del team ha degli altri. I robot *aware* hanno una qualche conoscenza degli altri robot, mentre i robot *unaware* agiscono senza alcuna informazione sul rimanente sistema. Bisogna osservare che la nozione di conoscenza non è equivalente a quella di comunicazione: infatti, l'utilizzo di un meccanismo di comunicazione non implica conoscenza e viceversa.
- *Coordination*: il terzo livello riguarda i meccanismi usati per la cooperazione. Definiamo il coordinamento come una cooperazione che prevede che ciascun robot prenda in considerazione le azioni eseguite dagli altri robot in modo tale che l'intero risultato sia coerente con il task globale da raggiungere e si abbiano elevate prestazioni. Tuttavia, ci sono diversi modi in cui un robot può prendere in considerazione le azioni degli altri membri del team. La funzione sottostante la cooperazione è il protocollo di coordinamento, definito come un insieme di regole che i robot devono seguire per interagire l'un l'altro nell'ambiente. Pertanto, possiamo ulteriormente classificare un sistema multi robot coordinato in base al tipo di protocollo di coordinamento. Consideriamo il coordinamento forte (debole) come una forma di coordinamento che si basa (non si basa) su un protocollo di coordinamento.
- *Organization*: il quarto livello della struttura gerarchica riguarda il modo in cui il sistema decisionale viene realizzato all'interno del sistema multi robot. Il livello organizzativo introduce una distinzione nelle forme di coordinamento, distinguendo gli approcci centralizzati da quelli distribuiti. Per sistema distribuito si intende l'autonomia di ciascun componente nel sistema nel prendere decisioni sulle azioni da eseguire. In particolare, un sistema centralizzato ha un agente leader che è incaricato di organizzare il lavoro degli altri agenti; il leader è coinvolto nel processo decisionale per l'intera squadra, mentre gli altri membri possono agire solo secondo le indicazioni

del leader. D'altra parte, un sistema distribuito è composto da agenti che sono completamente autonomi nel processo decisionale; in questa classe di sistemi un leader non esiste. La classificazione dei sistemi centralizzati può essere ulteriormente perfezionata a seconda del modo in cui viene gestita la leadership del gruppo. Nello specifico, una forte centralizzazione viene utilizzata per caratterizzare un sistema in cui le decisioni vengono prese dallo stesso agente leader predefinito durante l'intera durata della missione, mentre in un sistema debolmente centralizzato è consentito a più di un agente di assumere il ruolo di leader durante la missione.

1.3 Sistemi centralizzati/decentralizzati

Nell'ultimo decennio c'è stato un crescente interesse nel campo dei sistemi robotici decentralizzati grazie alle sue promettenti applicazioni in molti campi. Questi sistemi sono altamente desiderabili nella realizzazione di sistemi robot avanzati che svolgono compiti di alto livello come compiti di manutenzione in ambienti ostili e pericolosi o compiti per cui sia richiesta un'elevata affidabilità.

Tipicamente all'aumentare della complessità del task di interesse è richiesta sempre più cooperazione e coordinazione degli agenti del sistema. Nasce però la necessità di scegliere come la coordinazione deve avvenire, quale architettura (centralizzata o decentralizzata) utilizzare. Un possibile approccio è considerare il team come se fosse un singolo robot (un singolo sistema) con più gradi di libertà. Un computer centrale coordina il gruppo per eseguire in modo ottimo il task specificato. Questo unico dispositivo di controllo ha il ruolo di supervisore e il resto del team deve eseguire quanto stabilito dall'agente di controllo. Esistono dei grandi svantaggi però nell'utilizzo di un'architettura centralizzata.

- Innanzitutto è presente un collo di bottiglia in quanto un unico agente deve coordinare il comportamento di tutti gli altri. Infatti se il numero dei robot costituenti il team diviene troppo elevato si crea un carico computazionale troppo elevato sull'unità centrale.

- Se l'unità centrale subisce un guasto il sistema complessivo incorre in un blocco in quanto verrebbe a mancare l'entità coordinatrice. Questo aspetto può essere mitigato utilizzando un numero ridondante di unità centrali che con meccanismi di sincronizzazione siano in grado di assumere immediatamente il compito dell'unità che si è guastata.
- Nel caso in cui i robot debbano essere distribuiti su una vasta area geografica l'uso di una soluzione centralizzata richiederebbe la realizzazione di una onerosa infrastruttura di comunicazione che consenta di coprire lunghe distanze e di realizzare le comunicazioni punto-punto tra ciascun robot del team e l'agente centrale di controllo.

Gli approcci decentralizzati superano i problemi di quelli centralizzati assumendo che ogni robot operi in modo indipendente sfruttando informazioni disponibili localmente e acquisite attraverso i propri sensori. Un robot si può coordinare con altri robot vicini dividendo il task in diversi sotto task o collaborando per task che non possono essere svolti da un singolo robot. Sono richieste però anche in questo caso delle comunicazioni tra robot dal momento che questi hanno necessità di interagire con i propri vicini. Questo approccio fa sì che i robot siano in grado di adattarsi a cambiamenti dello stato dell'ambiente siccome i propri sensori gli permettono di percepirla ed acquisire informazioni su di essa in modo continuo e inoltre agiscono nell'ambiente in modo locale [4]. Questo rappresenta in parte un vantaggio e in parte uno svantaggio. Infatti mentre con l'approccio centralizzato l'unità coordinatrice conosce lo stato dell'intero sistema e può prendere decisioni sulla base di questo, in un approccio decentralizzato la conoscenza locale fa sì che sia richiesta una logica di controllo più complessa per poter svolgere task globali.

Alla luce di quanto detto è quindi evidente che l'architettura decentralizzata presenta complessivamente un notevole vantaggio in termini di scalabilità e robustezza ai guasti rispetto al caso centralizzato. Chiaramente la sola adozione di una struttura decentralizzata non rende automaticamente il sistema robusto ai guasti ma semplicemente aumenta la propensione del sistema a soddisfare tale proprietà. Tale architettura, infatti, è solo più “adatta” per implementare me-

canismi che garantiscano robustezza ai guasti in quanto i singoli agenti hanno ruoli paritari, pertanto il fault di uno di essi non equivale al caso critico di guasto dell'agente coordinatore. Scompare quindi anche la necessità di avere unità ridondanti e pronte all'utilizzo.

Non esiste, dunque, un'architettura definibile in assoluto migliore bensì la bontà delle diverse tipologie di architetture dipende fortemente dalla applicazione di interesse e dai vincoli da essa posti. La sfida attuale però è realizzare soluzioni basate su architetture decentralizzate altamente flessibili. Tale necessità si manifesta chiaramente nella realizzazione delle Smart Factory dell'Industry 4.0 come sottolineato nel paragrafo 1.1.

Capitolo 2

Elementi teorici

Nello studio di problemi di coordinamento e sincronizzazione di sistemi multi robot, la comunicazione tra robot assume un ruolo fondamentale. Il sistema dinamico complessivo infatti è il risultato dell’interconnessione di un certo numero di agenti (robot), che comunicano tra loro secondo una precisa topologia di comunicazione. È possibile tradurre tale topologia in termini matematici, utilizzando la teoria dei grafi. Il grafo rappresenta l’informazione che può fluire tra gli agenti. L’obiettivo del controllo cooperativo è progettare protocolli di controllo che garantiscono un comportamento sincronizzato desiderato dello stato di tutti i robot basandosi però sui soli scambi di informazione permessi dalla topologia del grafo. Cioè il protocollo di controllo per ogni robot si basa solo sulle informazioni che provengono da alcuni robot del team, chiamati vicini. Le restrizioni di comunicazione imposte dalla topologia del grafo possono limitare seriamente ciò che i protocolli di controllo distribuito possono fare a livello di team. Quindi diventa di fondamentale importanza per i sistemi cooperativi su grafi lo studio dei comportamenti complessivi sotto l’influenza del flusso di informazione permesso nel grafo.

2.1 Teoria algebrica dei grafi

Di seguito verranno presentati alcuni concetti di base della teoria dei grafi che sono essenziali nello studio dei sistemi dinamici multi agente.

2.1.1 Definizioni di base e connettività del grafo

Un *grafo* è una coppia $G = (V, E)$, dove $V = \{v_1, \dots, v_N\}$ è un insieme di N nodi o vertici ed E è l'insieme degli archi. Gli elementi di E sono indicati con la coppia (v_i, v_j) che indica l'arco che parte da v_i e termina in v_j . Si dice grafo semplice un grafo che non contiene cappi o auto-cicli, ovvero tale che $(v_i, v_i) \notin E, \forall i$, e che non contiene archi multipli tra la stessa coppia di nodi. L'arco (v_i, v_j) è detto uscente rispetto al nodo v_i ed entrante rispetto al nodo v_j , inoltre il nodo v_i è detto padre mentre il nodo v_j è detto figlio. Viene detto grado entrante (o in-degree) del nodo v_i il numero di archi entranti in v_i , mentre il grado uscente (o out-degree) dello stesso nodo, indica il numero di archi uscenti da esso. L'insieme dei vicini del nodo v_i è $\mathcal{N}_i = \{v_j : (v_j, v_i) \in E\}$, cioè l'insieme dei nodi collegati a v_i con archi entranti in v_i . Il numero di vicini $|\mathcal{N}_i|$ del nodo v_i è uguale al suo grado entrante.

Se il grado entrante è uguale al grado uscente per tutti i nodi $v_i \in V$, il grafo si dice *bilanciato*. Se $(v_i, v_j) \in E \Rightarrow (v_j, v_i) \in E, \forall i, j$, allora il grafo è detto *bidirezionale*, altrimenti il grafo è detto *orientato, diretto o digrafo*.

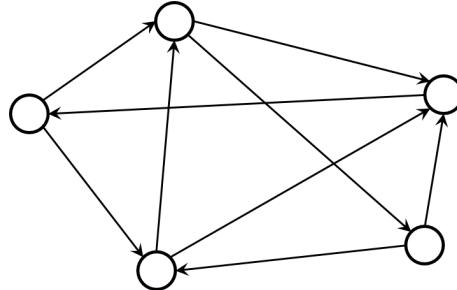


Figura 2.1: Grafo orientato o diretto

Associamo ad ogni arco $(v_j, v_i) \in E$ un peso a_{ij} , assumendo che i pesi non nulli siano strettamente positivi. Un grafo è detto *non orientato* se $a_{ij} = a_{ji}, \forall i, j$, ovvero se è bidirezionale e i pesi degli archi $(v_i, v_j) \in E$ e $(v_j, v_i) \in E$ sono uguali.

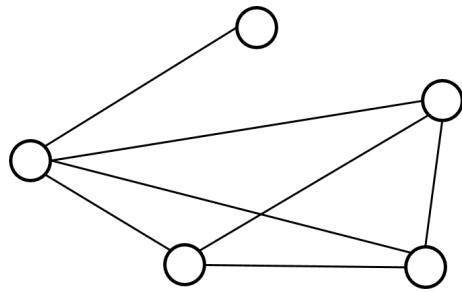


Figura 2.2: Grafo non orientato

Un cammino orientato è una sequenza di nodi v_0, v_1, \dots, v_r tali che $(v_i, v_{i+1}) \in E$, $i \in \{0, 1, \dots, r - 1\}$. Il nodo v_i è detto connesso al nodo v_j se esiste un cammino diretto da v_i a v_j . La *distanza* tra v_i e v_j è la lunghezza del cammino più corto da v_i a v_j .

Un grafo G è detto *fortemente connesso* se v_i e v_j sono connessi per tutti i nodi distinti $v_i, v_j \in V$. Per grafi bidirezionali o non orientati, se c'è un cammino diretto da v_i a v_j , allora c'è un cammino diretto da v_j a v_i e il termine fortemente si omette. Un grafo orientato (o diretto) si dice *connesso* se esiste un nodo dal quale è possibile raggiungere tutti gli altri nodi attraverso un percorso diretto.

Un albero diretto è un digrafo connesso in cui ogni nodo, eccetto uno chiamato radice, ha grado entrante pari a uno. Uno spanning tree di un digrafo è un albero diretto formato dai soli archi del grafo necessari a connettere tra loro tutti i vertici (o nodi) del grafo stesso, con un solo cammino. Quindi si dice che un grafo ha uno spanning tree se un sottoinsieme dei suoi archi forma un albero orientato (o diretto). Un grafo può avere più spanning tree. Se un grafo è fortemente connesso, esso contiene almeno uno spanning tree.

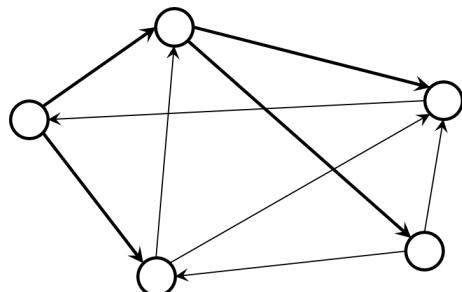


Figura 2.3: In evidenza lo spanning tree del digrafo

2.1.2 Descrizione dei grafi tramite matrici

La struttura e le proprietà di un grafo possono essere studiate esaminando le proprietà di alcune matrici associate al grafo stesso. Tale studio è conosciuto come teoria algebrica dei grafi.

Dati i pesi a_{ij} degli archi di un grafo, quest'ultimo può essere rappresentato da una matrice di adiacenza o di connettività $A = [a_{ij}]$, con pesi $a_{ij} > 0$ se $(v_j, v_i) \in E$ e $a_{ij} = 0$ altrimenti. Notare che $a_{ii} = 0$. Definiamo il grado entrante di un nodo v_i come la somma dell' i -esima riga di A :

$$d_i = \sum_{j=1}^N a_{ij} \quad (2.1)$$

e il grado uscente del nodo v_i come la somma dell' i -esima colonna di A :

$$d_i^o = \sum_{j=1}^N a_{ji} \quad (2.2)$$

La matrice di adiacenza A di un grafo non diretto è simmetrica, $A = A^T$. Un grafo non diretto è detto bilanciato se il grado entrante è uguale al grado uscente per ogni i .

Definiamo la matrice diagonale dei gradi entranti come $D = \text{diag}\{d_i\}$ e la matrice Laplaciana del grafo come $L = D - A$. Notare che L ha tutte le righe la cui somma è uguale a zero. Molte proprietà di un grafo possono essere studiate in termini del suo Laplaciano. In particolare, il Laplaciano è estremamente importante nello studio dei sistemi dinamici multi-agente basati su grafi.

2.1.3 Autostruttura della matrice Laplaciana

Abbiamo detto che la matrice Laplaciana del grafo gioca un ruolo chiave nell'analisi dei sistemi dinamici su grafi.

Definiamo la forma normale di Jordan della matrice Laplaciana del grafo come $L = M J M^{-1}$, con la matrice in forma di Jordan J e la matrice di trasformazione M date da:

$$J = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_N \end{bmatrix}, \quad M = \begin{bmatrix} v_1 & v_2 & \cdots & v_N \end{bmatrix} \quad (2.3)$$

dove gli autovalori λ_i e gli autovettori destri v_i soddisfano

$$(\lambda_i I - L) v_i = 0 \quad (2.4)$$

dove I è la matrice identità.

In generale i λ_i non sono scalari ma sono blocchi di Jordan nella forma:

$$\begin{bmatrix} \lambda_i & 1 & & \\ & \lambda_i & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{bmatrix} \quad (2.5)$$

Il numero dei blocchi di Jordan associati ad uno stesso autovalore λ_i è noto come molteplicità geometrica dell'autovalore λ_i . La somma delle dimensioni di tutti i blocchi di Jordan associati a λ_i è detta la sua molteplicità algebrica. Se tutti gli autovalori di L sono distinti, allora la forma di Jordan è semplice, ovvero è diagonale con tutti i blocchi di Jordan di dimensione 1.

Assumiamo ora che tutti gli autovalori siano ordinati in modo crescente $|\lambda_1| \leq |\lambda_2| \leq \cdots \leq |\lambda_N|$. Ogni grafo non orientato ha $L = L^T$ e quindi tutti gli autovalori sono reali e possono essere ordinati come $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$.

Dal momento che L ha tutte le righe la cui somma è nulla, allora si può scrivere:

$$L\underline{1} = 0 \quad (2.6)$$

con $\underline{1} = [1 \quad \dots \quad 1]^T \in R^N$ vettore unitario e c costante. Quindi $\lambda_1 = 0$ è un autovalore con autovettore destro $\underline{1}c$, ovvero $\underline{1}c \in N(L)$. Se la dimensione del nullo di L è uguale ad 1, allora il rango di L ha dimensione $N-1$ e dunque $\lambda_1 = 0$ è un autovalore non ripetuto e $\underline{1}c$ è il solo vettore nel nullo di L . Giungiamo al seguente risultato, molto importante: *L ha rango $N-1$, cioè $\lambda_1 = 0$ è non ripetuto, se e solo se il grafo G ha uno spanning tree* [6].

2.2 Il problema del consenso

Il processo decisionale distribuito per il coordinamento di reti di agenti dinamici ha interessato numerosi ricercatori negli ultimi anni. Ciò è in parte dovuto all'ampia applicazione dei sistemi multi-agente in molte aree; alcuni esempi sono: il controllo cooperativo di veicoli aerei senza equipaggio, la formazione di veicoli sottomarini, reti sensoriali distribuite.

2.2.1 Definizione di consenso

Consideriamo un grafo non orientato G , con L laplaciano associato ad esso, simmetrico e definito positivo. La *funzione di disaccordo* (anche chiamata potenziale laplaciano) associata a G è definita come segue:

$$\Phi_G(x) = x^T L x = \frac{1}{2} \sum_{ij \in E} (x_j - x_i)^2 \quad (2.7)$$

dove x_i denota il valore associato al nodo v_i . Il valore associato ad un nodo rappresenta quantità fisiche come ad esempio temperatura, posizione, velocità, voltaggio e così via. Si dice che due nodi distinti v_i e v_j sono in accordo se e solo se $x_i = x_j$. Quindi $\Phi_G(x)$ è una funzione che quantifica il disaccordo del gruppo in una rete.

2.2.2 Problema del consenso su grafi diretti

Consideriamo una rete di N integratori:

$$\dot{x}_i = u_i, \quad i \in I = \{1, 2, \dots, N\}, \quad x_i, u_i \in \mathbb{R} \quad (2.8)$$

con *topologia* $G = (V, E, A)$. Il problema del consenso asintotico può essere descritto come segue. Bisogna definire un protocollo che garantisce che lo stato della rete converga asintoticamente ad uno stato di equilibrio $x^* \in \mathbb{R}^n$ con elementi identici, cioè $x_i^* = x_j^* = \alpha, \forall i, j \in I, i \neq j$. L'elemento α che determina x^* è chiamato valore di decisione del gruppo. Un problema di consenso in cui $\alpha = \text{Ave}(x(0))$ è chiamato *Problema del Consenso Medio* con $\text{Ave}(x) = (\sum_{i=1}^n x_i) / n$.

Ci concentriamo sulla soluzione del problema del consenso medio utilizzando il seguente protocollo di accordo:

$$u_i(t) = \sum_{j \in N_i} a_{ij} (x_j(t) - x_i(t)), \quad i \in I \quad (2.9)$$

Dato il protocollo 2.9, lo stato della rete evolve in accordo al seguente sistema lineare

$$\dot{x}(t) = -Lx(t) \quad (2.10)$$

dove L è il Laplaciano del grafo G .

La chiave per l'analisi di stabilità del sistema risiede nelle proprietà spettrali della matrice Laplaciana del grafo. Il lemma seguente discende dal teorema dei dischi di Gershgorin ed è chiamato localizzazione spettrale.

Lemma 2.1. *Sia $G=(V,E,A)$ un grafo diretto con Laplaciano L . Indichiamo il nodo con grado uscente maggiore con $\delta(G) = \max_i d_i^o$. Tutti gli autovalori di L sono localizzati nel seguente cerchio:*

$$D(G) = \{z \in C : \|z - \delta(G)\| \leq \delta(G)\} \quad (2.11)$$

centrato in $z = \delta(G) + 0j$ nel piano complesso. Quindi la parte reale degli autovalori di $-L$ è non positiva, in quanto $\lambda_1 = 0$ e $\lambda_2, \dots, \lambda_N > 0$.

Questo garantisce la stabilità del sistema $\dot{x}(t) = -Lx(t)$ e dunque la convergenza

del protocollo di accordo per grafi diretti 2.9.

Lemma 2.2. *Consideriamo una rete di integratori con flusso di informazione regolato da G che è un digrafo fortemente connesso. Allora il protocollo 2.9 risolve globalmente asintoticamente un problema di consensus, cioè la soluzione converge asintoticamente all'equilibrio x^* tale che $x_i^* = x_j^*, \forall i, j, i \neq j$.*

È importante sottolineare che il lemma 2.2 non stabilisce se il valore di convergenza di ogni nodo è uguale a $\text{Ave}(x(0))$ o no, in altre parole tale lemma non risolve il problema del consenso medio. Una condizione sufficiente affinché il valore di decisione di ogni nodo sia uguale a $\text{Ave}(x(0))$ è che $\sum_{i=1}^n u_i \equiv 0$. Se G è non diretto (cioè $a_{ij} = a_{ji} > 0, \forall i, j : a_{ij} \neq 0$), automaticamente la condizione $\sum_{i=1}^n u_i = 0, \forall x$ è verificata e $\text{Ave}(x(t))$ è una quantità invariante. Questa proprietà non è però soddisfatta per digrafi generici.

2.2.3 Consenso medio su digrafi

Per la risoluzione del problema del consenso medio su digrafi viene utilizzata una particolare classe di grafi, ovvero i grafi bilanciati già definiti nel paragrafo 2.1.1. Di seguito alcuni importanti risultati[12].

Teorema 2.3. *Consideriamo una rete di integratori con flusso di informazione regolato da $G = (V, E, A)$ che è un grafo fortemente connesso. Allora G risolve globalmente asintoticamente il problema del consenso medio utilizzando il protocollo 2.9 se e solo se G è bilanciato.*

Lemma 2.4. *Consideriamo una rete di integratori con flusso di informazione diretto $G = (V, E, A)$ che è fortemente connesso. Allora, il digrafo G risolve globalmente asintoticamente il problema del consenso medio utilizzando il protocollo 2.9 se e solo se $\underline{1}^T L = 0$*

Corollario 2.5. *Assumiamo che tutte le condizioni del lemma 2.4 siano soddisfatte. Supponiamo che L abbia un autovettore sinistro $\gamma = (\gamma_1, \dots, \gamma_n)^T$, associato a $\lambda = 0$, dove γ è un vettore non negativo appartenente ad \mathbb{R}^n che soddisfa*

$\sum_i \gamma_i > 0$. Allora, il valore di decisione del gruppo dopo il raggiungimento del consenso è dato da

$$\alpha = \frac{\sum_i \gamma_i x_i(0)}{\sum_i \gamma_i} \quad (2.12)$$

2.3 Notazione

Nella tabella 2.1 sono riportati i principali simboli utilizzati nel resto del capitolo.

N	Numero di robot
$\mathbf{x}_i \in \mathbb{R}^l$	Configurazione dell'end-effector dell' i -esimo manipolatore
$\mathbf{q}_i \in \mathbb{R}^{n_i}$	Configurazione ai giunti dell' i -esimo manipolatore
$\boldsymbol{\tau}_i \in \mathbb{R}^{n_i}$	Coppie in ingresso all' i -esimo manipolatore
$\boldsymbol{\pi}_i \in \mathbb{R}^p$	Vettore dei parametri dinamici dell' i -esimo manipolatore
$\mathbf{f} \in \mathbb{R}^3$	Generico vettore delle forze
$\boldsymbol{\mu} \in \mathbb{R}^3$	Generico vettore dei momenti
$\mathbf{h} \in \mathbb{R}^3$	Generico vettore delle forze e dei momenti
$\mathbf{p} \in \mathbb{R}^3$	Generico vettore posizione
$\mathbf{r}_i \in \mathbb{R}^3$	Braccio virtuale relativo all' i -esimo manipolatore
\mathcal{T}	Generico frame
\mathbf{I}_m	Matrice identità di dimensione $m \times m$
\mathbf{O}_m	Matrice nulla di dimensione $m \times m$
\mathbf{a}_m	Vettore colonna di dimensione $m \times 1$ con tutti gli elementi uguali ad a

Tabella 2.1: Tabella contenente le principali variabili utilizzate nel seguente capitolo

2.4 Modello del singolo robot

L'analisi cinematica della struttura meccanica di un robot riguarda la descrizione del moto rispetto ad un sistema di riferimento cartesiano fisso senza che siano prese in considerazione le forze e i momenti che provocano il moto della struttura. È significativa la distinzione tra cinematica e cinematica differenziale. La cinematica descrive analiticamente le relazioni che intercorrono tra posizioni dei giunti e la posizione e l'orientamento dell'organo terminale, mentre la cinematica differenziale consente di esprimere le relazioni che legano, in termini di velocità, il moto dei giunti al moto dell'organo terminale, attraverso lo Jacobiano del manipolatore. I legami cinematici consentono di affrontare due problemi cardine della robotica: il problema cinematico diretto e il problema cinematico inverso. Il primo riguarda la determinazione di un metodo sistematico e generalizzato per descrivere, mediante strumenti di algebra lineare, il moto dell'organo terminale a partire dalla conoscenza del moto dei giunti, qualunque sia la struttura di manipolazione. Il secondo riguarda il problema inverso, la cui soluzione è di notevole importanza per tradurre le specifiche di moto, naturalmente assegnate all'organo terminale nello spazio di lavoro, nei moti corrispondenti ai giunti del manipolatore.

Per semplicità di scrittura, in questo paragrafo verrà omesso il pedice i vicino a ciascuna variabile, ma è importante ricordare che tutte le quantità di seguito sono riferito all' i -esimo manipolatore.

La cinematica diretta permette dunque di legare lo spazio operativo (ovvero quello in cui viene specificato il moto dell'organo terminale) e lo spazio giunti (ovvero quello in cui è definito il vettore delle $n_i \times 1$ variabili di giunto).

$$\mathbf{x} = \mathbf{k}(\mathbf{q}) \quad (2.13)$$

dove

$$\mathbf{x} = \begin{bmatrix} \mathbf{p} \\ \boldsymbol{\phi} \end{bmatrix}, \quad \mathbf{q} = \begin{bmatrix} q_1 \\ \vdots \\ q_n \end{bmatrix} \quad (2.14)$$

in cui \mathbf{p} rappresenta la posizione dell'end-effector, $\boldsymbol{\phi}_e$ rappresenta l'orientamento dell'end-effector espresso in termini di angoli di Eulero e q_j rappresenta la posizione del j -esimo giunto dell' i -esimo robot.

Dunque la funzione $\mathbf{k} : \mathbb{R}^{n_i} \rightarrow \mathbb{R}^l$ che in generale è non lineare, permette il calcolo delle variabili nello spazio operativo $\mathbf{p} \in \mathbb{R}^l$ a partire dalla conoscenza delle variabili nello spazio giunti $\mathbf{q} \in \mathbb{R}^{n_i}$.

Ciò che invece lega velocità ai giunti e le corrispondenti velocità lineari e angolari dell'organo terminale, come sudetto è la cinematica differenziale. Tali legami sono descritti da una matrice di trasformazione denominata Jacobiano Geometrico. Per altra via, se la posizione dell'end-effector è espressa facendo riferimento ad una rappresentazione in forma minima nello spazio operativo, ovvero attraverso posizione e angoli di Eulero, è possibile calcolare un diverso Jacobiano mediante un'operazione di differenziazione della funzione cinematica diretta rispetto alle variabili di giunto. Lo Jacobiano che si ottiene prende il nome di *Jacobiano analitico*. Lo Jacobiano è molto importante in quanto permette la caratterizzazione del manipolatore ed è utile per analisi di ridondanza, singolarità, per i legami tra forze applicate all'organo terminale e coppie ai giunti e per gli algoritmi di inversione cinematica.

L'equazione di cinematica differenziale è la seguente:

$$\mathbf{v} = \begin{bmatrix} \dot{\mathbf{p}} \\ \boldsymbol{\omega} \end{bmatrix} = \begin{bmatrix} \mathbf{J}_P(\mathbf{q}) \\ \mathbf{J}_O(\mathbf{q}) \end{bmatrix} \dot{\mathbf{q}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \quad (2.15)$$

dove \mathbf{J} rappresenta lo Jacobiano geometrico.

Allo stesso tempo si può ricavare un'altra equazione di cinematica differenziale a

partire dall'equazione di cinematica diretta 2.13:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{p}} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \mathbf{J}_P(\mathbf{q}) \\ \mathbf{J}_\phi(\mathbf{q}) \end{bmatrix} \dot{\mathbf{q}} = \mathbf{J}_A(\mathbf{q}) \dot{\mathbf{q}} \quad (2.16)$$

dove lo Jacobiano analitico è calcolato come:

$$\mathbf{J}_A(\mathbf{q}) = \frac{\partial \mathbf{k}(\mathbf{q})}{\partial \mathbf{q}} \quad (2.17)$$

Mentre la cinematica esprime il legame tra le posizioni dei giunti di un manipolatore e la posizione e l'orientamento dell'end-effector, la dinamica consente di calcolare le relazioni esistenti tra le coppie di attuazione ai giunti e il moto della struttura. La formulazione di Lagrange e la formulazione di Newton-Eulero consentono ricavare l'espressione del modello dinamico di un manipolatore nello spazio giunti:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{F}\dot{\mathbf{q}} + \mathbf{F}_s \text{sgn}(\dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau} - \mathbf{J}^T(\mathbf{q})\mathbf{h}_e \quad (2.18)$$

dove \mathbf{q} , $\dot{\mathbf{q}}$ e $\ddot{\mathbf{q}}$ rappresentano posizioni, velocità e accelerazioni degli n giunti dell' i -esimo robot, $\boldsymbol{\tau} \in \mathbb{R}^{n_i}$ è il vettore contenente le coppie di attuazione ai giunti, $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n_i \times n_i}$ è la matrice di inerzia simmetrica e definita positiva, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{n_i \times n_i}$ è la matrice che tiene conto degli effetti dinamici dei vari bracci cioè delle forze centrifughe e delle forze di Coriolis, $\mathbf{F} \in \mathbb{R}^{n_i \times n_i}$ è la matrice diagonale semidefinita positiva dei coefficienti di attrito viscoso, $\mathbf{F}_s \in \mathbb{R}^{n_i \times n_i}$ è la matrice diagonale semidefinita positiva dei coefficienti di attrito statico, $\mathbf{g}(\mathbf{q}) \in \mathbb{R}^{n_i}$ è il vettore dei termini gravitazionali e $\mathbf{J}^T(\mathbf{q})\mathbf{h}_e \in \mathbb{R}^{n_i}$ sono le coppie di attuazione necessarie per bilanciare le coppie ai giunti che vengono indotte quando l'organo terminale del manipolatore è in contatto con un ambiente.

Due proprietà notevoli del modello dinamico sono le seguenti.

- La matrice $\mathbf{N}(\mathbf{q}, \dot{\mathbf{q}}) = \dot{\mathbf{M}}(\mathbf{q}) - 2\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ risulta essere anti-simmetrica ovvero, dato un qualsiasi vettore $\mathbf{w} \in \mathbb{R}_i^n$ vale la relazione $\mathbf{w}^T \mathbf{N}(\mathbf{q}, \dot{\mathbf{q}}) \mathbf{w} = 0$.

- Il modello dinamico è lineare rispetto ad un insieme opportuno di parametri dinamici $\boldsymbol{\pi}$. È possibile scrivere quindi

$$\boldsymbol{\tau} = \mathbf{Y}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\boldsymbol{\pi} \quad (2.19)$$

dove $\boldsymbol{\pi} \in \mathbb{R}^p$ è un vettore di parametri costanti che rappresentano le caratteristiche di bracci e rotorì dei manipolatori come massa, attrito viscoso, e altro e $\mathbf{Y} \in \mathbb{R}^{n_i \times p}$ è una matrice che prende il nome di regressore ed è funzione delle posizioni, delle velocità e delle accelerazioni dei giunti e dipende dalla geometri del manipolatore.

Nello studio della dinamica di un manipolatore è significativo individuare la soluzione a due tipi di problemi che si pongono in relazione al calcolo della dinamica diretta e della dinamica inversa. Il problema della dinamica diretta consiste nel determinare le accelerazioni risultanti ai giunti $\ddot{\mathbf{q}}$ e quindi di conseguenza $\dot{\mathbf{q}}$ e \mathbf{q} , quando sono assegnate le coppie ai giunti $\boldsymbol{\tau}$ ed eventualmente le forze all'organo terminale \mathbf{h}_e . Il problema della dinamica inversa consiste nel determinare le coppie ai giunti $\boldsymbol{\tau}$ necessarie alla generazione del movimento specificato, assegnando le accelerazioni $\ddot{\mathbf{q}}$, le velocità $\dot{\mathbf{q}}$ e le posizioni \mathbf{q} dei giunti, note le eventuali forze all'organo terminale \mathbf{h}_e .

Molto spesso il modello dinamico del manipolatore di cui si dispone non è esatto, ma è un'approssimazione, una stima, del modello reale.

$$\hat{\mathbf{M}}(\mathbf{q})\ddot{\mathbf{q}} + \hat{\mathbf{C}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \hat{\mathbf{F}}\dot{\mathbf{q}} + \hat{\mathbf{g}}(\mathbf{q}) = \boldsymbol{\tau} = \mathbf{Y}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\hat{\boldsymbol{\pi}} \quad (2.20)$$

$\hat{\mathbf{M}}(\mathbf{q})$, $\hat{\mathbf{C}}(\mathbf{q}, \dot{\mathbf{q}})$, $\hat{\mathbf{F}}$, $\hat{\mathbf{g}}(\mathbf{q})$ e $\hat{\boldsymbol{\pi}}$ rappresentano rispettivamente le stime delle quantità omonime.

Nel seguito l'attrito statico sarà trascurato in quanto complesso da modellare per la sua natura fortemente non lineare.

2.5 Modello del sistema

Consideriamo ora l'intero sistema multi-robot, definendo nuove quantità riferite all'insieme di tutti gli agenti. Si definiscono i vettori \mathbf{q} e \mathbf{x} che rappresentano rispettivamente la collezione delle posizioni dei giunti di tutti i robot e la collezione delle configurazioni degli organi terminali di tutti i robot, ovvero: $\mathbf{q} = \begin{bmatrix} \mathbf{q}_1^T & \mathbf{q}_2^T & \dots & \mathbf{q}_N^T \end{bmatrix}^T \in \mathbb{R}^n$ con $n = \sum_{i=1}^N n_i$ e $\mathbf{x} = \begin{bmatrix} \mathbf{x}_1^T & \mathbf{x}_2^T & \dots & \mathbf{x}_N^T \end{bmatrix}^T \in \mathbb{R}^{lN}$. Le derivate temporali $\dot{\mathbf{q}}$ e $\dot{\mathbf{x}}$ sono legate, per quanto detto nel paragrafo precedente, dalla relazione:

$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \quad (2.21)$$

dove $\mathbf{J}(\mathbf{q}) = \text{diag}\{\mathbf{J}_1 \ \mathbf{J}_2 \ \dots \ \mathbf{J}_N\} \in \mathbb{R}^{Nl \times n}$.

Ai nostri scopi, facciamo le seguenti assunzioni:

- ciascun robot ha sempre norma dello Jacobiano limitata superiormente, ovvero esiste una costante positiva α tale che $\|\mathbf{J}_i(\mathbf{q}_i)\| \leq \alpha < \infty, \forall \mathbf{q}_i (i = 1, 2, \dots, N)$;
- ciascun robot è sempre lontano dalle configurazioni di singolarità ed ha quindi matrice Jacobiana avente rango pieno. Formalizzando: $\mathbf{J}_i(\mathbf{q}_i)$ è tale che $\lambda_{\min}(\mathbf{J}_i(\mathbf{q}_i)\mathbf{J}_i(\mathbf{q}_i)^T) > 0 (i = 1, 2, \dots, N)$, dove λ_{\min} è il più piccolo autovalore della matrice risultante dal prodotto in parentesi.

2.6 Decomposizione delle forze

Nel caso di task che richiedono una presa stretta e rigida da parte dei manipolatori dell'oggetto rigido da trasportare, un controllo puramente posizionale può causare stress interni e danneggiare l'oggetto e/o i manipolatori, specialmente nel caso in cui è assente un'unità centrale. Quindi diventa necessario considerare le forze generalizzate di interazione. In particolare, si richiede che i manipolatori si coordinino per trasportare l'oggetto ma allo stesso tempo regolino le forze applicate su esso. Questo è un problema che è stato ampiamente affrontato negli ultimi decenni nello studio delle soluzioni centralizzate, ma nel caso di soluzioni

distribuite si pongono diverse difficoltà.

Sia $\mathbf{h}_i \in \mathbb{R}^6$ il vettore delle forze generalizzate agenti sull' i -esimo end-effector:

$$\mathbf{h}_i = \begin{bmatrix} \mathbf{f}_i \\ \boldsymbol{\mu}_i \end{bmatrix} \quad (2.22)$$

dove \mathbf{f}_i ed $\boldsymbol{\mu}_i$ denotano, rispettivamente, le forze e i momenti.

Invocando il principio dei lavori virtuali, può essere derivata la relazione duale alla 2.15:

$$\boldsymbol{\tau}_i = \mathbf{J}^T(\mathbf{q}_i) \mathbf{h}_i \quad (2.23)$$

dove $\boldsymbol{\tau}_i \in \mathbb{R}^{n_i}$ rappresenta il vettore delle forze/coppie agenti sui giunti dell' i -esimo manipolatore.

Per semplicità consideriamo un sistema di due robot cooperanti i quali manipolano un oggetto comune. Sia C un punto fissato sull'oggetto da trasportare, ad esempio il suo centro di massa, la cui posizione espressa nel frame base è data da \mathbf{p}_c . Inoltre sia \mathcal{T}_C il frame posizionato sull'oggetto, avente origine nel punto C fissato.

Definiamo il braccio virtuale come il vettore \mathbf{r}_i ($i = 1, 2$) il quale determina la posizione del frame \mathcal{T}_C rispetto al frame \mathcal{T}_i ($i = 1, 2$). Assumiamo che \mathbf{r}_i sia un braccio rigido fissato all' i -esimo end-effector (Figura 2.4). Ogni braccio virtuale è dunque un vettore costante se l'oggetto trasportato è rigido e strettamente e rigidamente afferrato da ogni manipolatore. Nel caso descritto, la cinematica diretta di ogni manipolatore può essere espressa in termini di frame virtuali relativi agli end-effector, indicati con $\mathcal{T}_{S,i} = \mathcal{T}_C$, aventi stesso orientamento di \mathcal{T}_C e origine in $\mathbf{p}_{S,i} = \mathbf{p}_i + \mathbf{r}_i = \mathbf{p}_C$. Quindi posizione e orientamento dell'estremo di ogni braccio virtuale ($i = 1, 2$) sono dati rispettivamente da:

$$\mathbf{p}_{S,i} = \mathbf{p}_C, \quad \mathbf{R}_{S,i} = \mathbf{R}_C \quad (2.24)$$

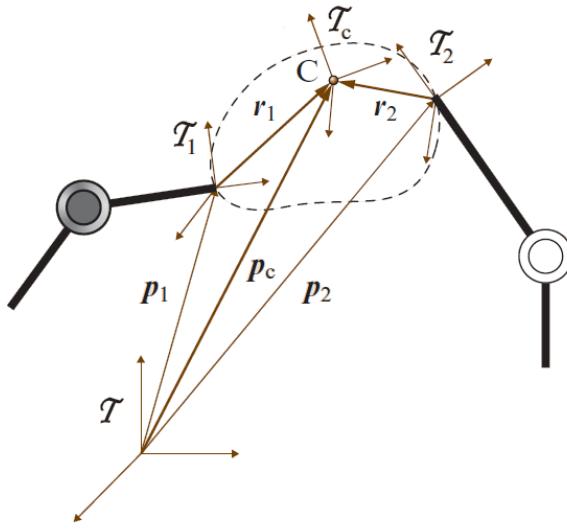


Figura 2.4: Geometria del grasping per un sistema composto da due manipolatori cooperanti che trasportano un oggetto comune

Sia $\mathbf{h}_{S,i}$ il vettore delle forze generalizzate agenti sulla punta dell' i -esimo braccio virtuale; si può facilmente verificare che vale la seguente relazione:

$$\mathbf{h}_{S,i} = \begin{pmatrix} \mathbf{I}_3 & \mathbf{O}_3 \\ -\mathbf{S}(\mathbf{r}_i) & \mathbf{I}_3 \end{pmatrix} \mathbf{h}_i = \mathbf{G}_i(\mathbf{r}_i) \mathbf{h}_i \quad (2.25)$$

dove \mathbf{O}_3 ed \mathbf{I}_3 sono rispettivamente la matrice nulla e la matrice identità di dimensioni 3×3 ed $\mathbf{S}(\mathbf{r}_i)$ è una matrice antisimmetrica di dimensioni 3×3 che esegue l'operazione del prodotto vettoriale. È importante sottolineare che \mathbf{G}_i è sempre a rango pieno.

Definiamo ora le *forze esterne* come il vettore $\mathbf{h}_E \in \mathbb{R}^6$ definito come:

$$\mathbf{h}_e = \mathbf{h}_{S,1} + \mathbf{h}_{S,2} = \mathbf{G}_S \mathbf{h}_S \quad (2.26)$$

con $\mathbf{G}_S = (\mathbf{I}_6 \quad \mathbf{I}_6)$ e $\mathbf{h}_S = (\mathbf{h}_{S,1}^T \quad \mathbf{h}_{S,2}^T)^T$; in altre parole \mathbf{h}_e rappresenta il vettore delle forze generalizzate che causano il moto dell'oggetto. Dalla 2.25 e 2.26 segue che \mathbf{h}_E può essere espresso in termini di forze all'end-effector, come:

$$\mathbf{h}_e = \mathbf{G}_1 \mathbf{h}_1 + \mathbf{G}_2 \mathbf{h}_2 = \mathbf{G} \mathbf{h} \quad (2.27)$$

con $\mathbf{G} = \begin{pmatrix} \mathbf{G}_1 & \mathbf{G}_2 \end{pmatrix}$ e $\mathbf{h}_S = \begin{pmatrix} \mathbf{h}_1^T & \mathbf{h}_2^T \end{pmatrix}^T$. \mathbf{G} è una matrice 6×12 avente rango pari a 6 e quindi immagine di dimensione 6 e nullo che di conseguenza ha anch'esso dimensione 6. Tale matrice prende il nome di *matrice di grasp*.

La forma della matrice \mathbf{G} si ricava a partire dalle equazioni della dinamica di un corpo rigido, le quali prendono il nome di equazione di Newton ed equazione di Eulero. Tali equazioni sono associate rispettivamente al moto traslatorio e al modo rotazionale e sono così definite:

$$\begin{cases} \mathbf{f}_O = \sum_{i=1}^2 \mathbf{f}_i \\ \boldsymbol{\mu}_O = \sum_{i=1}^2 \boldsymbol{\mu}_i + \sum_{i=1}^2 \mathbf{S}(\mathbf{r}_i) \mathbf{f}_i \end{cases} \quad (2.28)$$

È possibile riscrivere il sistema in forma matriciale:

$$\begin{aligned} \mathbf{h}_O &= \begin{bmatrix} \mathbf{f}_O \\ \boldsymbol{\mu}_O \end{bmatrix} = \sum_{i=1}^2 \left(\begin{bmatrix} \mathbf{I}_3 & \mathbf{O}_3 \\ -\mathbf{S}(\mathbf{r}_i) & \mathbf{I}_3 \end{bmatrix} \begin{bmatrix} \mathbf{f}_i \\ \boldsymbol{\mu}_i \end{bmatrix} \right) = \\ &= \sum_{i=1}^2 \mathbf{G}(\mathbf{r}_i) \mathbf{h}_i \\ &= \begin{bmatrix} \mathbf{G}_1 & \mathbf{G}_2 \end{bmatrix} \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \end{bmatrix} \\ &= \mathbf{G}\mathbf{h} \end{aligned} \quad (2.29)$$

La soluzione inversa alla 2.26 è data da:

$$\mathbf{h}_S = \mathbf{G}_S^\dagger \mathbf{h}_e + \mathbf{V}_S \mathbf{h}_{int} = \mathbf{U}_S \mathbf{h}_O \quad (2.30)$$

dove $\mathbf{G}_S^\dagger = \mathbf{G}_S^T (\mathbf{G}_S \mathbf{G}_S^T)^{-1}$ è la pseudo-inversa di \mathbf{G}_S ed è pari a:

$$\mathbf{G}_S^\dagger = \frac{1}{2} \begin{pmatrix} \mathbf{I}_6 \\ \mathbf{I}_6 \end{pmatrix} \quad (2.31)$$

\mathbf{V}_S è una matrice le cui colonne sono una base del nullo di \mathbf{G}_S .

Il secondo termine a destra della 2.30, cioè $\mathbf{V}_S \mathbf{h}_{int}$, rappresenta un vettore di forze

generalizzate, riferite all'estremo dei bracci virtuali, il quale si trova nel nullo di \mathbf{G}_S , quindi tali forze non contribuiscono al moto dell'oggetto. Esso rappresenta dunque, il carico interno dell'oggetto (cioè gli stress meccanici) ed è chiamato *vettore delle forze interne*.

Una diversa parametrizzazione delle soluzioni inverse alla 2.26 e 2.27 è data, rispettivamente, da:

$$\mathbf{h}_S = \mathbf{G}_S^\dagger \mathbf{h}_e + (\mathbf{I}_{12} - \mathbf{G}_S^\dagger \mathbf{G}_S) \mathbf{h}_S^* \quad (2.32)$$

e

$$\mathbf{h} = \mathbf{G}^\dagger \mathbf{h}_e + (\mathbf{I}_{12} - \mathbf{G}^\dagger \mathbf{G}) \mathbf{h}^* \quad (2.33)$$

dove \mathbf{h}_S^* (\mathbf{h}^*) è un vettore arbitrario di dimensioni 12×1 di forze generalizzate agenti sull'estremità dell' i -esimo braccio virtuale (i -esimo end-effector) proiettato nel nullo della matrice \mathbf{G}_S (\mathbf{G}) attraverso il termine $\mathbf{I}_{12} - \mathbf{G}_S^\dagger \mathbf{G}_S$ ($\mathbf{I}_{12} - \mathbf{G}^\dagger \mathbf{G}$) [1].

Generalizzando quanto detto fino ad ora, consideriamo N manipolatori e scriviamo:

$$\mathbf{h} = \mathbf{h}_e + \mathbf{h}_{int} = \mathbf{G}^\dagger \mathbf{G} \mathbf{h} + (\mathbf{I}_{Np} - \mathbf{G}^\dagger \mathbf{G}) \mathbf{h} \quad (2.34)$$

Basandoci sull'equazione 2.34, il contributo agli stress interni dovuto all' i -esimo manipolatore può essere calcolato come:

$$\begin{aligned} \mathbf{h}_{int,i} &= \boldsymbol{\Gamma}_i \mathbf{h}_{int} \\ &= \mathbf{h}_i - \boldsymbol{\Gamma}_i \mathbf{G}^\dagger \mathbf{G} \mathbf{h} \\ &= \mathbf{h}_i - \boldsymbol{\Gamma}_i \mathbf{G}^\dagger (\mathbf{G}_i \mathbf{h}_i + \sum_{j \neq i} \mathbf{G}_j \mathbf{h}_j) \\ &= \underbrace{(\mathbf{I}_p - \boldsymbol{\Gamma}_i \mathbf{G}^\dagger \mathbf{G}_i) \mathbf{h}_i}_{\text{locali}} - \underbrace{\boldsymbol{\Gamma}_i \mathbf{G}^\dagger \sum_{j \neq i} \mathbf{G}_j \mathbf{h}_j}_{\text{esterne}} \end{aligned} \quad (2.35)$$

dove $\boldsymbol{\Gamma}_i = \{\mathbf{O}_p \ \cdots \ \underbrace{\mathbf{I}_p}_{i\text{-esimo robot}} \ \cdots \ \mathbf{O}_p\} \in \mathbb{R}^{p \times Np}$ e p è il numero di variabili di spazio operativo che è possibile controllare.

L'ultimo membro dell'equazione 2.35 mostra che il contributo da parte dell' i -esimo manipolatore agli stress interni esercitati sull'oggetto, è dato dalla somma di

un contributo locale (che è noto) e un contributo esterno (non noto in framework decentralizzati) il quale è dipendente dalle forze esercitate dagli altri manipolatori e necessita di essere stimato localmente.

2.7 Dinamica dell'oggetto

Consideriamo un oggetto rigido rigidamente trasportato da N robot (esempio in Figura 2.5 con $N = 4$).

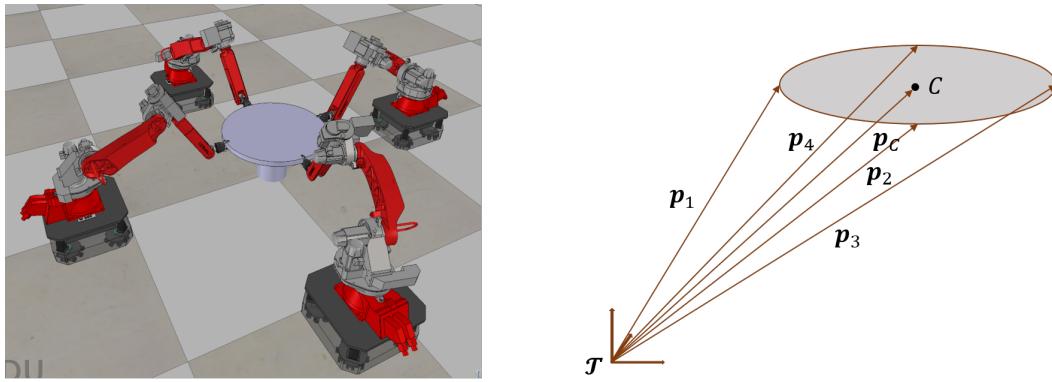


Figura 2.5: Oggetto rigido trasportato da $N = 4$ manipolatori

La configurazione dell'oggetto è data da $\mathbf{x}_o \in \mathbb{R}^l$ e la sua dinamica può essere descritta da:

$$\mathbf{M}_o \ddot{\mathbf{x}}_o + \mathbf{C}(\mathbf{x}_o, \dot{\mathbf{x}}_o) + \mathbf{g}_o(\mathbf{x}_o) = \mathbf{h}_o \quad (2.36)$$

dove $\mathbf{M}_o \in \mathbb{R}^{l \times l}$ è la matrice di inerzia, $\mathbf{C}_o \in \mathbb{R}^{l \times l}$ contiene i termini centrifughi e di Coriolis, $\mathbf{g}_o \in \mathbb{R}^l$ le forze gravitazionali, e $\mathbf{h}_o \in \mathbb{R}^l$ è la risultante delle forze generalizzate esercitate dai manipolatori sull'oggetto, tali che:

$$\mathbf{h}_o(\mathbf{x}) = \mathbf{G}(\mathbf{x}, \mathbf{x}_o)\mathbf{h} \quad (2.37)$$

dove $\mathbf{h} \in \mathbb{R}^{Nl}$ è il vettore di tutte le forze generalizzate agli end-effector dei manipolatori:

$$\mathbf{h} = \begin{bmatrix} \mathbf{h}_1^T & \mathbf{h}_2^T & \dots & \mathbf{h}_N^T \end{bmatrix}^T \in \mathbb{R}^{Nl} \quad (2.38)$$

e $\mathbf{G} = [\mathbf{G}_1 \ \dots \ \mathbf{G}_N] \in \mathbb{R}^{l \times Nl}$ è la matrice di grasp definita nel paragrafo 2.6.

Se si riferisce la cinematica di ogni robot al punto $\mathbf{p}_o \equiv \mathbf{p}_C$, allora $\mathbf{G}_i \forall i$, diventa indipendente dal vettore \mathbf{r}_i (vedi Figura 2.4): $\mathbf{G}_i = \mathbf{I}_p$.

Si assume che ogni manipolatore conosca la geometria e i parametri dinamici dell'oggetto da trasportare e riferisca la sua cinematica ad uno stesso punto \mathbf{p}_o .

Capitolo 3

Schemi di controllo

Il lavoro sviluppato nella fase simulativa presenta un'architettura decentralizzata che consente di eseguire complesse attività di manipolazione in cui i robot fanno affidamento solo su informazioni disponibili localmente. Vengono considerate due tipologie di controllo, il controllo cinematico puro e il controllo di interazione. La prima tipologia di controllo potrebbe essere utile per il coordinamento di robot o per la manipolazione di oggetti deformabili e flessibili; mentre la seconda è necessaria per gestire in modo cooperativo oggetti rigidi ed evitare stress interni.

Il task di movimento cooperativo è espresso da opportune variabili di task il cui valore dipende dalle configurazioni degli end-effector di tutti i robot del team. Queste variabili possono essere utili per specificare, ad esempio, il centroide della formazione degli end-effector, la posizione dell'oggetto o le configurazioni relative degli organi terminali.

Viene adottata un'architettura a strati per poter effettuare un controllo distribuito. Al primo livello, ciascun manipolatore aggiorna continuamente la propria stima dello stato complessivo del sistema, adottando un osservatore basato sul consenso. Al secondo livello, tale stima viene utilizzata per calcolare l'input al controllo adattativo impiegato per inseguire il task cooperativo.

Per quanto riguarda il caso della manipolazione di un oggetto rigido comune da parte di un team di robot, possono sorgere forze interne generalizzate che non contribuiscono al movimento dell'oggetto, specialmente nel caso in cui non esista

un controllore centralizzato.

3.1 Notazione

Nella *tabella 3.1* sono riportati i principali simboli utilizzati nel resto del capitolo.

\mathbf{v}_i	Variabile generica \mathbf{v} relativa all' i -esimo agente
\mathbf{v}	Vettore contenente le variabili \mathbf{v}_i di tutti gli agenti, cioè $\mathbf{v} = \begin{bmatrix} \mathbf{v}_1^T & \mathbf{v}_2^T & \dots & \mathbf{v}_N^T \end{bmatrix}^T$
${}^i\hat{\mathbf{v}}$	Stima di \mathbf{v} fatta dall' i -esimo agente
$\hat{\mathbf{v}}^*$	Insieme dei vettori delle stime dei diversi agenti, cioè $\hat{\mathbf{v}}^* = \begin{bmatrix} {}^1\hat{\mathbf{v}}^T & {}^2\hat{\mathbf{v}}^T & \dots & {}^N\hat{\mathbf{v}}^T \end{bmatrix}^T$
${}^i\tilde{\mathbf{v}}$	Errore di stima commesso dall' i -esimo agente, cioè ${}^i\tilde{\mathbf{v}} = \mathbf{v} - {}^i\hat{\mathbf{v}}$
$\tilde{\mathbf{v}}^*$	Insieme degli errori di stima dei diversi agenti, cioè $\tilde{\mathbf{v}}^* = \begin{bmatrix} {}^1\tilde{\mathbf{v}}^T & {}^2\tilde{\mathbf{v}}^T & \dots & {}^N\tilde{\mathbf{v}}^T \end{bmatrix}^T$
$\boldsymbol{\sigma}(\mathbf{x}) \in \mathbb{R}^m$	Variabile di task dipendente dallo stato del sistema
$\boldsymbol{\sigma}_d \in \mathbb{R}^m$	Valore desiderato della variabile di task cooperativo
$\tilde{\boldsymbol{\sigma}} \in \mathbb{R}^m$	Errore sul task $\tilde{\boldsymbol{\sigma}} = \boldsymbol{\sigma}_d - \boldsymbol{\sigma}$
$\boldsymbol{\zeta}_d \in \mathbb{R}^m$	$\boldsymbol{\zeta}_d = \dot{\boldsymbol{\sigma}}_d + \mathbf{k}_\sigma \boldsymbol{\sigma}_d$
$\boldsymbol{\gamma} \in \mathbb{R}^m$	$\boldsymbol{\gamma} = \boldsymbol{\zeta}_d - \mathbf{k}_\sigma \boldsymbol{\sigma}(\mathbf{x}) = \dot{\boldsymbol{\sigma}}_d + \mathbf{k}_\sigma (\boldsymbol{\sigma}_d - \boldsymbol{\sigma})$
${}^i\hat{\boldsymbol{\gamma}} \in \mathbb{R}^m$	Stima di $\boldsymbol{\gamma}$ fatta dal robot i -esimo e definita come ${}^i\hat{\boldsymbol{\gamma}} = \boldsymbol{\zeta}_d - \mathbf{k}_\sigma \boldsymbol{\sigma}({}^i\hat{\mathbf{x}})$

Tabella 3.1: Tabella contenente le principali variabili utilizzate nel seguente capitolo

3.2 Funzione di task globale

La funzione di task globale, indicata con $\sigma(\mathbf{x}) \in \mathbb{R}^m$ viene utilizzata per specificare un comportamento desiderato (task desiderato) del sistema multi-robot complessivo. Tale funzione dipende dallo stato complessivo \mathbf{x} del sistema, ovvero dalla configurazione degli end-effector dei robot. È possibile esprimere dunque, mediante la funzione $\sigma(\mathbf{x})$, il comportamento cooperante dei robot del team. Un esempio di task desiderato è quello che viene utilizzato in questo lavoro di tesi ed è costituito da due elementi, il baricentro e la formazione del team di robot.

Introduciamo la matrice Jacobiana del task $\mathbf{J}_\sigma \in \mathbf{R}^{m \times Np}$; è possibile esprimere, tramite lo jacobiano \mathbf{J}_σ , il task e le sue derivate prima e seconda, in funzione dello stato \mathbf{x} dell'intero sistema:

$$\begin{cases} \sigma(\mathbf{x}) = \mathbf{J}_\sigma \mathbf{x} \\ \dot{\sigma}(\mathbf{x}) = \mathbf{J}_\sigma \dot{\mathbf{x}} \\ \ddot{\sigma}(\mathbf{x}) = \mathbf{J}_\sigma \ddot{\mathbf{x}} \end{cases} \quad (3.1)$$

Il comportamento desiderato del sistema complessivo è definito in termini di funzione di task globale desiderato $\sigma_d(t)$, definendo anche le sue derivate, prima e seconda, e viene assunto che tale funzione sia nota a tutti i robot. Questa scelta non è vincolante, in quanto non si hanno limiti nel caso in cui il task sia noto solo ad un sottinsieme di agenti del sistema. In tal caso è possibile utilizzare meccanismi di stima del task, dove la stima è eseguita dal sottinsieme di robot, chiamati *follower*, che non conosce $\sigma_d(t)$. I restanti robot invece prendono il nome di *leader* [8].

Dopo che l'utente ha assegnato il valore desiderato della funzione di task $\sigma_d(t)$, l'obiettivo è calcolare gli ingressi di controllo di ciascun robot τ_i ($i = 1, 2, \dots, N$) tali che $\sigma(\mathbf{x})$ converga asintoticamente a $\sigma_d(t)$ in assenza di un'unità di controllo centrale.

È possibile definire l'errore di inseguimento della traiettoria nello spazio del task

come:

$$\tilde{\boldsymbol{\sigma}} = \boldsymbol{\sigma}_d(t) - \boldsymbol{\sigma}(\mathbf{x}) \quad (3.2)$$

Si vuole dunque, in seguito a quanto detto, che tale errore converga asintoticamente all'origine.

Nel seguito saranno omesse le dipendenze di $\boldsymbol{\sigma}_d(t)$, $\dot{\boldsymbol{\sigma}}_d(t)$ e $\ddot{\boldsymbol{\sigma}}_d(t)$ dal tempo e di $\tilde{\boldsymbol{\sigma}}(\mathbf{x})$ dallo stato del sistema.

3.2.1 Baricentro e formazione

Molti task di utilizzo pratico, come la movimentazione di un oggetto da parte di più bracci meccanici o un moto di puro coordinamento tra robot, possono essere facilmente descritti nello spazio del task facendo uso di un insieme di variabili *absolute* e *relative*.

- Baricentro. Viene scelto come baricentro il centroide delle configurazioni degli end-effector, il quale è dato dalla media delle configurazioni stesse. Tale media esprime le posizioni e gli orientamenti del frame associato al centroide dell'oggetto, rappresentando dunque una variabile assoluta ed è calcolata come:

$$\boldsymbol{\sigma}_1(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i = \mathbf{J}_{\sigma,1} \mathbf{x} \quad (3.3)$$

dove $\boldsymbol{\sigma}_1(\mathbf{x}) \in \mathbb{R}^p$ e $\mathbf{J}_{\sigma,1} = \frac{1}{N} \mathbf{1}_N^T \otimes \mathbf{I}_p \in \mathbb{R}^{p \times Np}$ è la matrice jacobiana del task relativo al centroide.

Le variabili assolute sono utilizzate per controllare la traiettoria dell'oggetto trasportato.

- Formazione. Un task di formazione può essere espresso tramite variabili relative, le quali rappresentano gli spostamenti relativi tra gli end-effector dei robot. La funzione del task, nel caso di task di formazione, può essere descritta come:

$$\boldsymbol{\sigma}_2(\mathbf{x}) = \begin{bmatrix} (\mathbf{x}_2 - \mathbf{x}_1)^T & (\mathbf{x}_3 - \mathbf{x}_2)^T & \cdots & (\mathbf{x}_N - \mathbf{x}_{N-1})^T \end{bmatrix}^T = \mathbf{J}_{\sigma,2} \mathbf{x} \quad (3.4)$$

dove $\boldsymbol{\sigma}_2(\mathbf{x}) \in \mathbb{R}^{(N-1)p}$ e $\mathbf{J}_{\sigma,2} \in \mathbb{R}^{(N-1)p \times Np}$ è la matrice jacobiana del task relativo alla formazione ed è definita come:

$$\mathbf{J}_{\sigma,2} = \begin{bmatrix} -\mathbf{I}_p & \mathbf{I}_p & \mathbf{O}_p & \cdots & \mathbf{O}_p & \mathbf{O}_p \\ \mathbf{O}_p & -\mathbf{I}_p & \mathbf{I}_p & \cdots & \mathbf{O}_p & \mathbf{O}_p \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{O}_p & \mathbf{O}_p & \mathbf{O}_p & \cdots & -\mathbf{I}_p & \mathbf{I}_p \end{bmatrix} \quad (3.5)$$

La funzione di task complessiva, per controllare sia baricentro che formazione, è, quindi, data semplicemente dalla concatenazione delle precedenti funzioni di task ovvero:

$$\boldsymbol{\sigma}(\mathbf{x}) = \begin{bmatrix} \boldsymbol{\sigma}_1(\mathbf{x}) \\ \boldsymbol{\sigma}_2(\mathbf{x}) \end{bmatrix} = \mathbf{J}_{\sigma}\mathbf{x} \in \mathbb{R}^{Np} \quad (3.6)$$

dove

$$\mathbf{J}_{\sigma} = \begin{bmatrix} \mathbf{J}_{\sigma,1} \\ \mathbf{J}_{\sigma,2} \end{bmatrix} \in \mathbb{R}^{Np \times Np} \quad (3.7)$$

Si osservi che considerando un task di controllo di baricentro e formazione la matrice \mathbf{J}_{σ} assume rango pieno; pertanto la dimensione complessiva m dello spazio del task è pari a Np . Analizzando il numero di gradi di libertà del sistema complessivo, si evince che esso è pari proprio a Np in quanto, per ognuno degli N robot vi sono p gradi di libertà dati dalla configurazione dell'organo terminale. Con tale task si esauriscono, quindi, tutti i gradi di libertà del sistema sui quali è possibile agire per realizzare il task cooperante.

3.3 Algoritmo di controllo decentralizzato

Lo scopo dell'algoritmo di controllo decentralizzato che è stato implementato in questo lavoro, è riuscire a calcolare l'input di controllo per ciascun manipolatore del team, senza che ognuno conosca lo stato complessivo del sistema, e quindi dovendo effettuare una stima di esso, riuscendo a realizzare il task cooperativo desiderato. Proprio perché lo stato complessivo del sistema non è noto, viene adottata un'architettura a due livelli, come già sottolineato precedentemente.

Al primo livello ciascun controllore implementa ed esegue un osservatore distribuito che stima lo stato totale del sistema.

Al secondo livello, la stima viene utilizzata per calcolare l'input di controllo locale come avverrebbe se la soluzione adottata fosse centralizzata. In un sistema di controllo centralizzato, infatti, l'entità coordinatrice conosce lo stato di tutti i robot del sistema e stabilisce quali azioni ognuno di essi debba compiere per ottenere il comportamento complessivo desiderato.

In aggiunta, dal momento che i parametri dinamici dei manipolatori non possono essere noti perfettamente, viene utilizzata una legge di controllo locale adattativa, al fine di contrastare l'incertezza.

La soluzione implementata è valida non solo per compiti di puro coordinamento tra robot, ma anche per casi in cui i manipolatori formano una catena cinematica chiusa con l'oggetto che stanno manipolando. In quest'ultimo caso, l'obiettivo è spostare l'oggetto limitando però gli stress interni ad esso che tuttavia non possono essere calcolati localmente in modo semplice in quanto sarebbe necessaria la conoscenza perfetta dello stato dell'intero sistema da parte di ogni robot. Per ovviare a ciò, la componenete di forza esercitata da ciascun robot viene decomposta in due contributi, uno esterno che contribuisce al movimento dell'oggetto e uno interno che è stimato a livello locale e di conseguenza compensato.

3.3.1 I livello. Osservatore globale

Come sudetto il primo livello dell'architettura di controllo di ciascun robot implementa un osservatore globale mediante il quale viene stimato lo stato complesivo del sistema multi-robot, ovvero vengono stimate le configurazioni di tutti gli end-effector del team, il quale ha il compito di svolgere il task cooperativo.

Introduciamo il vettore

$${}^i \hat{\mathbf{x}} = \begin{bmatrix} {}^i \hat{\mathbf{x}}_1^T & {}^i \hat{\mathbf{x}}_2^T & \dots & {}^i \hat{\mathbf{x}}_N^T \end{bmatrix}^T \in \mathbb{R}^{Np} \quad (3.8)$$

dove ${}^i\hat{\mathbf{x}}_j \in \mathbb{R}^p$ è la stima dello stato \mathbf{x}_j del robot j -esimo fatta dal robot i -esimo, mentre ${}^i\hat{\mathbf{x}} \in \mathbb{R}^{Np}$ è la stima dello stato \mathbf{x} dell'intero sistema fatta dal robot i -esimo.

Per stimare lo stato complessivo del sistema, ciascun robot può utilizzare solo:

- informazioni locali, ovvero il proprio stato \mathbf{x}_i determinato mediante sensori di bordo;
- informazioni ottenute mediante comunicazione con i robot 'vicini', i quali sono determinati dalla topologia del grafo.

L'obiettivo dell'utilizzo del livello dell'osservatore globale, è far convergere asintoticamente la stima ${}^i\hat{\mathbf{x}}$ effettuata da ogni robot, allo stato effettivo del sistema $\mathbf{x} = [\mathbf{x}_1^T \quad \mathbf{x}_2^T \quad \dots \quad \mathbf{x}_N^T]^T$. Dunque l'osservatore implementa la seguente legge di aggiornamento dello stato:

$${}^i\dot{\hat{\mathbf{x}}} = k_o \left(\sum_{j \in \mathcal{N}_i} ({}^j\hat{\mathbf{x}} - {}^i\hat{\mathbf{x}}) + \boldsymbol{\Pi}_i (\mathbf{x} - {}^i\hat{\mathbf{x}}) \right) + {}^i\hat{\mathbf{u}} \quad (3.9)$$

dove:

- $k_o \in \mathbb{R}^+$ è un guadagno positivo che deve essere progettato opportunamente;
- $\boldsymbol{\Pi}_i = \boldsymbol{\Gamma}_i^T \boldsymbol{\Gamma}_i = \text{diag}\{\mathbf{O}_p \underbrace{\dots \mathbf{I}_p}_{i\text{-esimo robot}} \dots \mathbf{O}_p\} \in \mathbb{R}^{Np \times Np}$ è una matrice diagonale, con $\boldsymbol{\Gamma}_i$ definito nel paragrafo 2.6;
- ${}^i\hat{\mathbf{u}}({}^i\hat{\mathbf{x}}, t) = \mathbf{J}_\sigma^\dagger {}^i\hat{\gamma} \in \mathbb{R}^{Np}$ è la stima, da parte del robot i -esimo, dell'ingresso di controllo del sistema, ovvero la stima dei riferimenti in velocità agli end-effector. Vale che:
 - ${}^i\hat{\gamma} = \boldsymbol{\zeta}_d - k_\sigma \boldsymbol{\sigma}({}^i\hat{\mathbf{x}})$ con $\boldsymbol{\sigma}({}^i\hat{\mathbf{x}}) = \mathbf{J}_\sigma {}^i\hat{\mathbf{x}}$ e $\boldsymbol{\zeta}_d = \dot{\boldsymbol{\sigma}}_d + k_\sigma \boldsymbol{\sigma}_d$ e di conseguenza ottenendo ${}^i\hat{\gamma} = \dot{\boldsymbol{\sigma}}_d + k_\sigma \tilde{\boldsymbol{\sigma}}({}^i\hat{\mathbf{x}})$. Tale quantità contiene dunque la velocità desiderata nello spazio del task e un termine proporzionale alla stima dell'errore di inseguimento del task desiderato;
 - $k_\sigma \in \mathbb{R}^+$ è un guadagno positivo da progettare opportunamente.

Può sembrare che il robot i -esimo utilizzi l'intero stato \mathbf{x} . In realtà la matrice $\boldsymbol{\Pi}_i$ seleziona solo l' i -esima componente del vettore \mathbf{x} , ovvero le uniche informazioni utilizzate sono quelle locali che corrispondono allo stato \mathbf{x}_i che è noto.

Siccome la legge di aggiornamento della stima dello stato del sistema è uguale per ogni robot, è possibile riscriverla per l'intero sistema. Definiamo il vettore delle stime dello stato del sistema come:

$$\hat{\mathbf{x}}^* = \begin{bmatrix} {}^1\hat{\mathbf{x}}^T & {}^2\hat{\mathbf{x}}^T & \dots & {}^N\hat{\mathbf{x}}^T \end{bmatrix}^T \in \mathbb{R}^{N^2p} \quad (3.10)$$

e l'errore collettivo di stima come:

$$\begin{aligned} \tilde{\mathbf{x}}^* &= \mathbf{1}_N \otimes \mathbf{x} - \hat{\mathbf{x}}^* \\ &= \left[(\mathbf{x} - {}^1\hat{\mathbf{x}})^T \quad (\mathbf{x} - {}^2\hat{\mathbf{x}})^T \quad \dots \quad (\mathbf{x} - {}^N\hat{\mathbf{x}})^T \right]^T \\ &= \begin{bmatrix} {}^1\tilde{\mathbf{x}}^T & {}^2\tilde{\mathbf{x}}^T & \dots & {}^N\tilde{\mathbf{x}}^T \end{bmatrix}^T \end{aligned} \quad (3.11)$$

La dinamica della stima collettiva è data da:

$$\dot{\hat{\mathbf{x}}}^* = -k_o(\mathbf{L} \otimes \mathbf{I})\hat{\mathbf{x}}^* + k_o\boldsymbol{\Pi}^*\tilde{\mathbf{x}}^* + \hat{\mathbf{u}}^* \quad (3.12)$$

dove $\hat{\mathbf{u}}^* = \begin{bmatrix} {}^1\hat{\mathbf{u}}^T & {}^2\hat{\mathbf{u}}^T & \dots & {}^N\hat{\mathbf{u}}^T \end{bmatrix}^T \in \mathbb{R}^{N^2p}$ e $\boldsymbol{\Pi}^* = diag\{\boldsymbol{\Pi}_1 \ \boldsymbol{\Pi}_2 \ \dots \ \boldsymbol{\Pi}_N\} \in \mathbb{R}^{N^2p \times N^2p}$.

La dinamica dell'errore di stima è:

$$\dot{\tilde{\mathbf{x}}}^* = \mathbf{1}_N \otimes \dot{\mathbf{x}} - \dot{\hat{\mathbf{x}}}^* = -k_o\tilde{\mathbf{L}}^*\tilde{\mathbf{x}}^* + \mathbf{1}_N \otimes \dot{\mathbf{x}} - \hat{\mathbf{u}}^* \quad (3.13)$$

con

$$\tilde{\mathbf{L}}^* = \mathbf{L} \otimes \mathbf{I}_{Np} + \boldsymbol{\Pi}^* \in \mathbb{R}^{N^2p \times N^2p} \quad (3.14)$$

dove \mathbf{L} è il Laplaciano del grafo di connettività e $-\tilde{\mathbf{L}}^*$ è una matrice di Hurwitz perché ha autovalori a parte reale negativa.

3.3.2 II livello. Legge di controllo adattativa locale

Una volta che ciascun robot ha ottenuto la stima dello stato dell'intero sistema, bisogna calcolare gli input di controllo, ovvero le coppie di attuazione τ_i ($i = 1, 2, \dots, N$), affinché si abbia che $\sigma(\mathbf{x})$ converga asintoticamente a σ_d .

Come detto nell'introduzione del paragrafo 3.3, pur essendo nota la struttura del modello dinamico dei manipolatori espressa dal regressore \mathbf{Y}_i , si ha incertezza sui parametri dinamici rappresentati dal vettore π_i . La legge di controllo per l' i -esimo manipolatore dunque è la seguente:

$$\tau_i = \hat{\mathbf{M}}_i(\mathbf{q}_i)\ddot{\mathbf{q}}_{\sigma,i} + \hat{\mathbf{C}}_i(\mathbf{q}_i, \dot{\mathbf{q}}_i)\dot{\mathbf{q}}_{\sigma,i} + \hat{\mathbf{F}}_i\dot{\mathbf{q}}_{\sigma,i} + \hat{\mathbf{g}}(\mathbf{q}_i) = \mathbf{Y}_i(\mathbf{q}_i, \dot{\mathbf{q}}_i, \dot{\mathbf{q}}_{\sigma,i}, \ddot{\mathbf{q}}_{\sigma,i})\hat{\pi}_i \quad (3.15)$$

Per contrastare tale incertezza ed adattare i parametri a quelli reali è possibile adottare una soluzione che permette di adattare in linea il modello computazionale al modello dinamico reale del manipolatore. Viene adottata dunque una legge di controllo di tipo adattativo che si può ricavare grazie ad una proprietà del modello dinamico del manipolatore, ovvero la linearità nei parametri dinamici, descritta nel paragrafo 2.4. Infatti è sempre possibile esprimere le equazioni non lineari del moto in forma lineare rispetto ad un opportuno insieme di parametri dinamici costanti. Pertanto la legge di controllo per l' i -esimo manipolatore si modifica rispetto alla legge precedente (3.15) in:

$$\begin{aligned} \tau_i &= \hat{\mathbf{M}}_i(\mathbf{q}_i)\ddot{\mathbf{q}}_{\sigma,i} + \hat{\mathbf{C}}_i(\mathbf{q}_i, \dot{\mathbf{q}}_i)\dot{\mathbf{q}}_{\sigma,i} + \hat{\mathbf{F}}_i\dot{\mathbf{q}}_{\sigma,i} + \hat{\mathbf{g}}(\mathbf{q}_i) + k_q\dot{\tilde{\mathbf{q}}}_{\sigma,i} \\ &= \mathbf{Y}_i(\mathbf{q}_i, \dot{\mathbf{q}}_i, \dot{\mathbf{q}}_{\sigma,i}, \ddot{\mathbf{q}}_{\sigma,i})\hat{\pi}_i + k_q\dot{\tilde{\mathbf{q}}}_{\sigma,i} \end{aligned} \quad (3.16)$$

dove

- $k_q \in \mathbb{R}^+$ è un guadagno scalare positivo da progettare in modo opportuno;
- $\dot{\mathbf{q}}_{\sigma,i} = \mathbf{J}_i^\dagger (\Gamma_i \mathbf{J}_\sigma^\dagger {}^i \hat{\gamma}) + \dot{\mathbf{q}}_{n,i}$ rappresenta i riferimenti in velocità per i giunti e dipende sia dalla velocità desiderata imposta nello spazio del task, sia dall'errore in posizione espresso anch'esso nel task space. Nella definizione di $\dot{\mathbf{q}}_{\sigma,i}$ compaiono i seguenti termini:

- $\mathbf{J}_i^\dagger = \mathbf{J}_i^T (\mathbf{J}_i \mathbf{J}_i^T)^{-1} \in \mathbb{R}^{n_i \times p}$ è la matrice pseudo-inversa dello Jacobiano del manipolatore \mathbf{J}_i ;
- $\dot{\mathbf{q}}_{n,i}$ è un termine che rappresenta una velocità ai giunti addizionale e si mappa nel nullo dello Jacobiano \mathbf{J}_i (cioè $\mathbf{J}_i \dot{\mathbf{q}}_{n,i} = \mathbf{0}_p$). Tale termine può essere usato per evitare movimenti dei giunti interni, ovvero che non contribuiscono al moto dell'end-effector, o per soddisfare vincoli o obiettivi secondari come il tenersi lontani da configuarazioni di singolarità o dagli ostacoli che si presentano nell'ambiente circostante;
- $\ddot{\mathbf{q}}_{\sigma,i} = \mathbf{J}_i^\dagger \Gamma_i \mathbf{J}_\sigma^\dagger i \dot{\gamma} + \dot{\mathbf{J}}_i^\dagger \mathbf{J}_i \dot{\mathbf{q}}_{\sigma,i} + \ddot{\mathbf{q}}_{n,i}$ rappresenta i riferimenti in accelerazione ai giunti del robot i -esimo;
- $\dot{\tilde{\mathbf{q}}}_{\sigma,i} = \dot{\mathbf{q}}_{\sigma,i} - \dot{\mathbf{q}}_i$ rappresenta l'errore di inseguimento dei riferimenti in velocità;
- $\ddot{\tilde{\mathbf{q}}}_{\sigma,i} = \ddot{\mathbf{q}}_{\sigma,i} - \ddot{\mathbf{q}}_i$ rappresenta l'errore di inseguimento dei riferimenti in accelerazione;
- $\dot{\hat{\pi}}_i = \mathbf{K}_{\pi_i}^{-1} \mathbf{Y}_i^T \dot{\tilde{\mathbf{q}}}_{\sigma,i}$, è la legge di aggiornamento dei parametri dinamici, con $K_{\pi_i} \in \mathbb{R}^{n_{\pi_i} \times n_{\pi_i}}$ matrice di guadagni, simmetrica e definita positiva;

È importante notare che in seguito alla legge di adattamento dei parametri, non necessariamente $\hat{\pi}$ tenderà a π . La convergenza dei parametri al valore vero dipende infatti dalla struttura della matrice $Y_i(\mathbf{q}_i, \dot{\mathbf{q}}_i, \dot{\mathbf{q}}_{\sigma,i}, \ddot{\mathbf{q}}_{\sigma,i})$ e dalla leggi di moto desiderate ed eseguite. D'altrocanto l'approccio adottato è orientato alla soluzione di un problema di controllo adattativo diretto, mirato all'individuazione di una legge di controllo efficiente e non alla determinazione dei parametri che caratterizzano il modello matematico del manipolatore.

Nella legge di controllo si individuano, riassumendo, tre contributi principali:

1. $\mathbf{Y}_i(\mathbf{q}_i, \dot{\mathbf{q}}_i, \dot{\mathbf{q}}_{\sigma,i}, \ddot{\mathbf{q}}_{\sigma,i}) \hat{\pi}_i$ che esprime un'azione riconducibile ad una strategia di controllo a dinamica inversa e assicura in modo approssimato la compensazione degli effetti non lineari e il disaccoppiamento tra i giunti;

2. $k_q \dot{\tilde{\mathbf{q}}}_{\sigma,i}$ che introduce un'azione lineare stabilizzante di tipo PD sull'errore di inseguimento;
3. il vettore di stima dei parametri $\hat{\boldsymbol{\pi}}$ che viene aggiornato secondo una legge adattativa di tipo gradiente, in modo da assicurare asintoticamente la compensazione dei termini del modello dinamico del manipolatore; la matrice \mathbf{K}_π determina la velocità di convergenza dei parametri al loro valore asintotico.

La legge di controllo adattativo sui parametri richiede la disponibilità di un modello computazionale completo e non prevede azioni mirate a ridurre gli effetti di disturbi esterni. Infatti se questi si verificano, gli effetti indotti sulle grandezze di uscita sono attribuiti dal controllore a disadattamento sui parametri stimati. Di conseguenza la legge di controllo tenta di contrastare tali effetti agendo su grandezze che in realtà non li hanno provocati.

Sostituendo la legge di controllo presentata (3.16), all'interno del modello dinamico del manipolatore (2.18), si ha:

$$\mathbf{M}_i(\mathbf{q}_i)\ddot{\tilde{\mathbf{q}}}_{\sigma,i} + \mathbf{C}_i(\mathbf{q}_i, \dot{\mathbf{q}})\dot{\tilde{\mathbf{q}}}_{\sigma,i} + \mathbf{F}_i\dot{\tilde{\mathbf{q}}}_{\sigma,i} = \quad (3.17)$$

$$\tilde{\mathbf{M}}_i(\mathbf{q}_i)\ddot{\mathbf{q}}_{\sigma,i} + \tilde{\mathbf{C}}_i(\mathbf{q}_i, \dot{\mathbf{q}})\dot{\mathbf{q}}_{\sigma,i} + \tilde{\mathbf{F}}_i\dot{\mathbf{q}}_{\sigma,i} + \tilde{\mathbf{g}}(\mathbf{q}_i) - k_q \dot{\tilde{\mathbf{q}}}_{\sigma,i} + \mathbf{J}_i^T \mathbf{h}_i = \quad (3.18)$$

$$Y_i(\mathbf{q}_i, \dot{\mathbf{q}}_i, \dot{\mathbf{q}}_{\sigma,i}, \ddot{\mathbf{q}}_{\sigma,i})\tilde{\boldsymbol{\pi}}_i - k_q \dot{\tilde{\mathbf{q}}}_{\sigma,i} + \mathbf{J}_i^T \mathbf{h}_i \quad (3.19)$$

dove $\tilde{\boldsymbol{\pi}} = \hat{\boldsymbol{\pi}} - \boldsymbol{\pi}$, $\tilde{\mathbf{M}} = \hat{\mathbf{M}} - \mathbf{M}$, $\tilde{\mathbf{C}} = \hat{\mathbf{C}} - \mathbf{C}$, $\tilde{\mathbf{F}} = \hat{\mathbf{F}} - \mathbf{F}$ e $\tilde{\mathbf{g}} = \hat{\mathbf{g}} - \mathbf{g}$.

3.3.3 Controllo distribuito puramente cinematico

Nel caso del controllo puramente cinematico è importante controllare in modo preciso la formazione degli end-effector. In questo caso, i manipolatori sono debolmente accoppiati o possono essere adattati per il trasporto o la manipolazione di oggetti deformabili o flessibili per i quali è più importante controllare la loro forma piuttosto che le forze esercitate su di essi.

Bisogna progettare in modo opportuno i guadagni presentati nelle leggi di controllo dei paragrafi precedenti. Si vuole ottenere la convergenza asintotica all'origine,

delle seguenti quantità: $\dot{\tilde{\mathbf{q}}}$, $\tilde{\mathbf{x}}^*$ e $\tilde{\boldsymbol{\sigma}}$. Affinchè questo sia possibile, i guadagni k_o , k_σ e k_q sono scelti in modo che rispettino:

$$\begin{cases} k_o \lambda_L - \rho_1 > 0 \Leftrightarrow k_\sigma < \frac{k_o \lambda_L}{2N^2} \\ k_q > \frac{N\alpha^2}{4(k_o \lambda_L - \rho_1)} \end{cases} \quad (3.20)$$

con

$$\begin{cases} \lambda_L = \lambda_{min}(\tilde{L}^*) \\ \lambda_1 = \lambda_{min}(\mathbf{F} + k_q \mathbf{I}_{Nn}) > 0 \\ \rho_1 = 2N^2 > 0 \end{cases} \quad (3.21)$$

La condizione su k_o è solo sufficiente, ma si dimostra che la dinamica dell'osservatore deve essere più veloce di quella del controllore. Riguardo λ_L , si ha che il tuning richiede di conoscere la topologia del grafo di comunicazione in anticipo; se questo non è possibile può essere considerata la topologia nel caso peggiore (cioè la topologia che minimizza λ_L), ottenendo dunque un tuning conservativo.

I componenti principali dell'architettura dell'algoritmo di controllo utilizzata per il robot i -esimo, sono mostrati in Figura 3.1. È strutturata nel modo seguente:

- l'osservatore globale (3.9) dà in output una stima ${}^i\hat{\mathbf{x}}$ dello stato complessivo del sistema, sulla base delle informazioni provenienti dai vicini e sulla base dello stato corrente \mathbf{x}_i ;
- la stima è poi usata dal controllo di task globale (presentato nel paragrafo 3.3.1) per stimare i riferimenti in velocità ${}^i\hat{\mathbf{u}}({}^i\hat{\mathbf{x}})$ per gli end-effector;
- la legge di controllo adattativa (presentata nel paragrafo 3.3.2), sulla base della stima delle velocità di riferimento all'end-effector e della configurazione dei giunti, stabilisce le coppie di attuazione per i giunti del manipolatore;
- il manipolatore (3.16) riceve in input le coppie di attuazione e aggiorna il proprio stato \mathbf{x}_i in base ad esse.

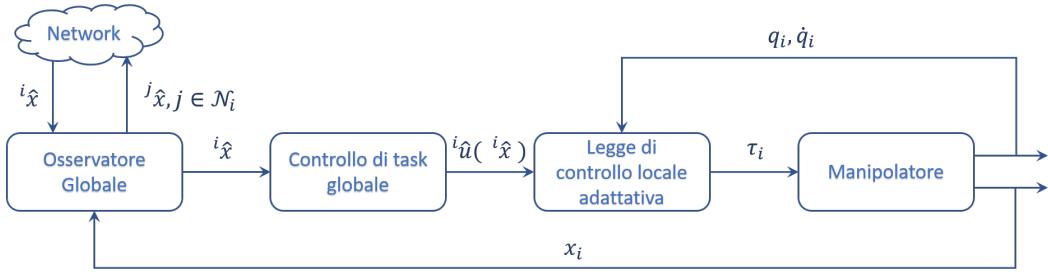


Figura 3.1: Schema di controllo per il controllo puramente cinematico, che mostra i principali componenti della soluzione progettata con riferimento all' i -esimo manipolatore

3.3.4 Controllo distribuito di interazione

Nel caso di task che richiedono un connessione forte tra i manipolatori e la manopilazione di oggetti rigidi, un controllo puramente cinematico può causare stress interni e danneggiare l'oggetto o i manipolatori. Quindi bisogna tener conto in modo esplicito delle forze di interazione generalizzate. In questo caso applicativo, si suppone che su ogni robot sia montato un sensore di forza per misurare le forze di interazione generalizzate \mathbf{h}_i e dunque queste ultime saranno note. Saranno invece non noti i contributi in forza dovuti agli altri robot, ovvero: $\sum_{j \neq i} \mathbf{G}_j \mathbf{h}_j$.

A questo scopo, dal punto di vista dell' i -esimo manipolatore e considerando l'equazione 2.37, l'equazione 2.36 può essere riscritta come

$$\mathbf{M}_o(\mathbf{x}_o) \ddot{\mathbf{x}}_o = \mathbf{G}_i \mathbf{h}_i + \sum_{j \neq i} \mathbf{G}_j \mathbf{h}_j - \mathbf{C}_o(\mathbf{x}_o, \dot{\mathbf{x}}_o) - \mathbf{g}_o(\mathbf{x}_o) \quad (3.22)$$

dove, in particolare, $\mathbf{G}_i \mathbf{h}_i$ è il vettore delle forze generalizzate esercitate dall' i -esimo manipolatore e $\sum_{j \neq i} \mathbf{G}_j \mathbf{h}_j$ è il vettore che tiene conto delle forze generalizzate esercitate da tutti gli altri manipolatori. La stima di quest'ultimo termine, per essere utilizzata, deve essere calcolata dall' i -esimo manipolatore adottando il seguente approccio. Definiamo il vettore $\boldsymbol{\theta}_i(t) \in \mathbb{R}^p$ come:

$$\boldsymbol{\theta}_i(t) = \mathbf{K} \left(\int_{t_0}^t (\boldsymbol{\alpha} - \mathbf{G}_i \mathbf{h}_i - \boldsymbol{\theta}_i) d\tau + \mathbf{m}(t) \right) \quad (3.23)$$

dove $\mathbf{K} \in \mathbb{R}^{p \times p}$ è una matrice costante diagonale definita positiva, $\mathbf{m}(t) = \mathbf{M}_o(\mathbf{x}_o) \dot{\mathbf{x}}_o$ è il momento generalizzato dell'oggetto e $\boldsymbol{\alpha} = \mathbf{g}_o - \frac{1}{2} \dot{\mathbf{x}}_o^T \frac{\partial \mathbf{M}_o}{\partial \mathbf{x}_o} \dot{\mathbf{x}}_o$.

Si dimostra che [3]

$$\dot{\boldsymbol{\theta}}_i(t) = -\mathbf{K}\boldsymbol{\theta}_i(t) + \mathbf{K} \sum_{j \neq i} \mathbf{G}_j \mathbf{h}_j \quad (3.24)$$

il quale rappresenta un filtro passa-basso che può essere reso arbitrariamente veloce selezionando valori alti della matrice $\mathbf{K} \in \mathbb{R}^{p \times p}$, ottenendo asintoticamente $\boldsymbol{\theta}_i \approx \sum_{j \neq i} \mathbf{G}_j \mathbf{h}_j$ e da

$$\mathbf{h}_{int,i} = (\mathbf{I}_p - \boldsymbol{\Gamma}_i \mathbf{G}^\dagger \mathbf{G}_i) \mathbf{h}_i - \boldsymbol{\Gamma}_i \mathbf{G}^\dagger \sum_{j \neq i} \mathbf{G}_j \mathbf{h}_j \quad (3.25)$$

si ottiene

$$\mathbf{h}_{int,i} = \boldsymbol{\Gamma}_i (\mathbf{I}_{Np} - \mathbf{G}^\dagger \mathbf{G}) \mathbf{h} \approx (\mathbf{I}_p - \boldsymbol{\Gamma}_i \mathbf{G}^\dagger \mathbf{G}_i) \mathbf{h}_i - \boldsymbol{\Gamma}_i \mathbf{G}^\dagger \boldsymbol{\theta}_i \quad (3.26)$$

In virtù di 3.25, l'errore approssimato, fatto in 3.26 nel calcolo di $\mathbf{h}_{int,i}$ è

$$\mathbf{h}_{int,i} - \left((\mathbf{I}_p - \boldsymbol{\Gamma}_i \mathbf{G}^\dagger \mathbf{G}_i) \mathbf{h}_i - \boldsymbol{\Gamma}_i \mathbf{G}^\dagger \boldsymbol{\theta}_i \right) = \boldsymbol{\Gamma}_i \mathbf{G}^\dagger \left(\boldsymbol{\theta}_i - \sum_{j \neq i} \mathbf{G}_j \mathbf{h}_j \right) \quad (3.27)$$

che è trascurabile solo quando l'input al filtro (ossia $\sum_{j \neq i} \mathbf{G}_j \mathbf{h}_j$) ha una banda molto più piccola della frequenza di taglio del filtro. Una pratica scelta è prendere \mathbf{K} in 3.23 il più alto possibile col vincolo che il filtro debba essere implementabile in digitale.

È importante sottolineare che, a partire da 3.26, il contributo del robot i -esimo ad \mathbf{h}_o (cioè al moto dell'oggetto) può essere stimato dal robot i come

$$\mathbf{h}_i - \mathbf{h}_{int,i} \approx \boldsymbol{\Gamma}_i (\mathbf{G}^\dagger \mathbf{G}_i \mathbf{h}_i + \mathbf{G}^\dagger \boldsymbol{\theta}_i) \quad (3.28)$$

che può essere calcolato localmente.

Nel caso del controllo di interazione, l'input di controllo $\boldsymbol{\tau}_i$, ($i = 1, 2, \dots, N$) si modifica nel seguente modo:

$$\begin{aligned} \boldsymbol{\tau}_i &= \hat{\mathbf{M}}_i(\mathbf{q}_i) \ddot{\mathbf{q}}_{\sigma,i} + \hat{\mathbf{C}}_i(\mathbf{q}_i, \dot{\mathbf{q}}_i) \dot{\mathbf{q}}_{\sigma,i} + \hat{\mathbf{F}}_i \dot{\mathbf{q}}_{\sigma,i} + \hat{\mathbf{g}}(\mathbf{q}_i) + \Delta \boldsymbol{\tau}_i \\ &= \mathbf{Y}_i(\mathbf{q}_i, \dot{\mathbf{q}}_i, \dot{\mathbf{q}}_{\sigma,i}, \ddot{\mathbf{q}}_{\sigma,i}) \hat{\boldsymbol{\pi}}_i + \Delta \boldsymbol{\tau}_i \end{aligned} \quad (3.29)$$

dove:

- $\Delta\tau_i \in \mathbb{R}^{n_i}$ è un input addizionale che sarà definito in seguito;
- $\dot{\mathbf{q}}_{\sigma,i}$ si modifica rispetto alla $\dot{\mathbf{q}}_{\sigma,i}$ definita nel Paragrafo 3.3.2, in $\dot{\mathbf{q}}_{\sigma,i} = \mathbf{J}_i^\dagger (\Gamma_i \mathbf{J}_\sigma^\dagger {}^i \hat{\gamma} + \mathbf{u}_{f,i}) + \dot{\mathbf{q}}_{n,i}$;
- $\ddot{\mathbf{q}}_{\sigma,i}$ si modifica rispetto alla $\ddot{\mathbf{q}}_{\sigma,i}$ definita nel Paragrafo 3.3.2, in $\ddot{\mathbf{q}}_{\sigma,i} = \mathbf{J}_i^\dagger (\Gamma_i \mathbf{J}_\sigma^\dagger {}^i \dot{\hat{\gamma}} + \dot{\mathbf{u}}_{f,i}) + \mathbf{J}_i^\dagger \mathbf{J}_i \dot{\mathbf{q}}_{\sigma,i} + \ddot{\mathbf{q}}_{n,i}$;
- $\mathbf{u}_{f,i} \in \mathbb{R}^p$ è un input di controllo aggiuntivo che sarà spiegato di seguito.

Quindi, per l'inseguimento del task desiderato σ_d è necessario che ciascun robot calcoli e tenga conto delle forze interne $\mathbf{h}_{int,i}$, al fine di ottenere la convergenza di tali forze a dei valori desiderati $\mathbf{h}_{int,i}^d$ (ad esempio imponendo $\mathbf{h}_{int,i}^d = \mathbf{0}_p$ si sta chiedendo che gli stress interni all'oggetto siano nulli). L'errore sulle forze interne calcolate localmente, si può quindi calcolare come segue: $\tilde{\mathbf{h}}_{int,i} = \mathbf{h}_{int,i}^d - h_{int,i}$ e il vettore degli errori sulle forze interne, collettivo, è dato da: $\tilde{\mathbf{h}}_{int} = [\tilde{\mathbf{h}}_{int,1}^T \dots \tilde{\mathbf{h}}_{int,N}^T]^T \in \mathbb{R}^{Np}$. Vale il seguente teorema.

Teorema 3.1. *Consideriamo il modello in 2.18 e l'input di controllo in 3.29 con legge di aggiornamento dei parametri dinamici definita nel Paragrafo 3.3.2 e con legge di aggiornamento dell'osservatore espressa in 3.9, allora $\dot{\mathbf{q}}_\sigma$, $\tilde{\mathbf{x}}^*$ e $\tilde{\mathbf{h}}_{int}$ convergono asintoticamente all'origine se $\mathbf{u}_{f,i}$ è scelto come:*

$$\mathbf{u}_{f,i} = k_f \int_{t_o}^t \tilde{\mathbf{h}}_{int,i} d\tau \quad (3.30)$$

con $k_f > 0$ e k_o, k_q, k_σ, k_f scelti in modo che:

$$\begin{cases} k_o \lambda_L - \rho_1 > 0 \Leftrightarrow k_\sigma < \frac{k_o \lambda_L}{2N^2} \\ k_q > \frac{N\alpha^2}{4(k_o \lambda_L - \rho_1)} \\ k_f > \frac{\lambda_1}{4(\lambda_1(k_o \lambda_L - \rho_1) - \alpha^2/4)} \end{cases} \quad (3.31)$$

e $\Delta\tau_i$ vale:

$$\Delta\tau_i = \mathbf{J}_i^T \left(\Gamma_i (\mathbf{G}^\dagger \mathbf{G}_i \mathbf{h}_i + \mathbf{G}^\dagger \boldsymbol{\theta}_i) + \mathbf{h}_{int,i}^d + k_f \mathbf{u}_{f,i} \right) + \kappa_i(t) \dot{\tilde{\mathbf{q}}}_{\sigma,i} \quad (3.32)$$

dove $\kappa_i(t)$ è un guadagno scalare adattativo tempo variante, che soddisfa la seguente relazione:

$$\kappa_i(t) > \frac{\|\Gamma_i \mathbf{J}_\sigma^\dagger \dot{\gamma} - \dot{\mathbf{x}}_i\| \|\tilde{\mathbf{h}}_{int,i} + k_f \mathbf{u}_{f,i}\|}{\|\dot{\tilde{\mathbf{q}}}_{\sigma,i}\|^2} \quad (3.33)$$

Inoltre se $\mathbf{J}_\sigma \mathbf{u}_f = \mathbf{0}_m$ anche $\tilde{\boldsymbol{\sigma}}$ converge all'origine.

La dimostrazione può essere trovata in [8].

Il contributo in coppia aggiuntivo $\Delta\tau_i$ è composto da tre contributi:

- $\Gamma_i (\mathbf{G}^\dagger \mathbf{G}_i \mathbf{h}_i + \mathbf{G}^\dagger \boldsymbol{\theta}_i)$ che rappresenta la compensazione del contributo dato dalle forze esterne generalizzate dovute all' i -esimo manipolatore, applicate sull'oggetto;
- $\mathbf{h}_{int,i}^d + k_f \mathbf{u}_{f,i}$ che rappresenta un feed-forward in forza che ha lo scopo di far convergere all'origine la quantità $\tilde{\mathbf{h}}_{int,i}$;
- $\kappa_i(t) \dot{\tilde{\mathbf{q}}}_{\sigma,i}$ che rappresenta un termine di robustezza.

Il termine $\|\dot{\tilde{\mathbf{q}}}_{\sigma,i}\|$ si suppone converga a zero e questo può causare la crescita del termine $\kappa_i(t)$ senza limiti. Quindi, nella pratica viene utilizzata la seguente approssimazione:

$$\begin{cases} \kappa_i(t) = \frac{\|\Gamma_i \mathbf{J}_\sigma^\dagger \dot{\gamma} - \dot{\mathbf{x}}_i\| \|\tilde{\mathbf{h}}_{int,i} + k_f \mathbf{u}_{f,i}\|}{\|\dot{\tilde{\mathbf{q}}}_{\sigma,i}\|^2} & \text{if } \|\dot{\tilde{\mathbf{q}}}_{\sigma,i}\|^2 > \epsilon \\ \kappa_i(t) = \frac{\|\Gamma_i \mathbf{J}_\sigma^\dagger \dot{\gamma} - \dot{\mathbf{x}}_i\| \|\tilde{\mathbf{h}}_{int,i} + k_f \mathbf{u}_{f,i}\|}{\epsilon} & \text{if } \|\dot{\tilde{\mathbf{q}}}_{\sigma,i}\|^2 \leq \epsilon \end{cases} \quad (3.34)$$

dove ϵ è uno scalare costante positivo.

L'architettura di controllo adottata può essere visualizzata complessivamente nello schema di Figura 3.2 [8].

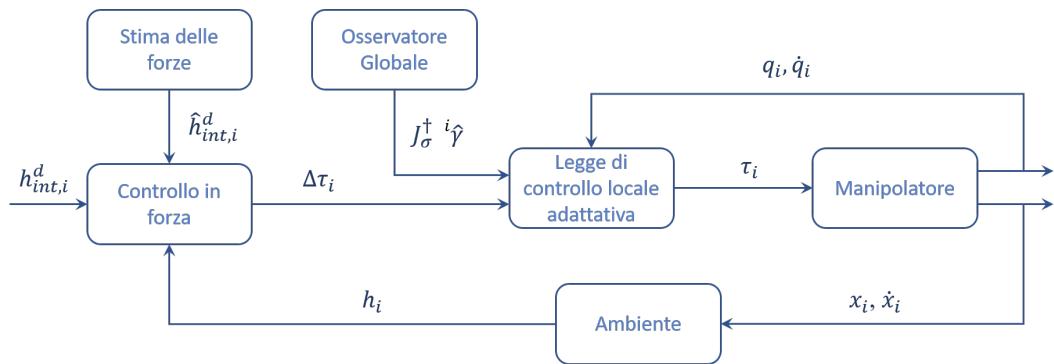


Figura 3.2: Schema di controllo che per il controllo di interazione, che mostra i principali componenti della soluzione progettata con riferimento all'*i*-esimo manipolatore

Capitolo 4

Simulazione e risultati

Nel seguente capitolo vengono riportati i risultati simulativi. Per le simulazioni è stato considerato un team di robot omogeneo, ovvero un team composto da quattro robot Comau Smart-Six la cui struttura cinematica sarà descritta nel seguito del capitolo.

Il software per la simulazione è stato sviluppato in Matlab e come ambiente simulativo, per la visualizzazione dei moti dei robot, è stato utilizzato il simulatore V-REP.

4.1 Comau Smart-Six

Come sudetto, per le simulazioni il team scelto è composto da quattro robot Comau Smart-Six (Figura 4.1). Questi robot sono stati montati su base mobile per avere due gradi di libertà aggiuntivi e dunque ottenere dei robot ridondanti (in totale si hanno 8 gradi di libertà).



Figura 4.1: Robot COMAU SMART-SIX utilizzato nella simulazione

La struttura del robot è del tipo antropomorfo ed è composta da 6 giunti rotoidali, avendo dunque 6 gradi di libertà. Nelle Figure 4.2 e 4.3 viene mostrato lo spazio di lavoro del robot Comau.

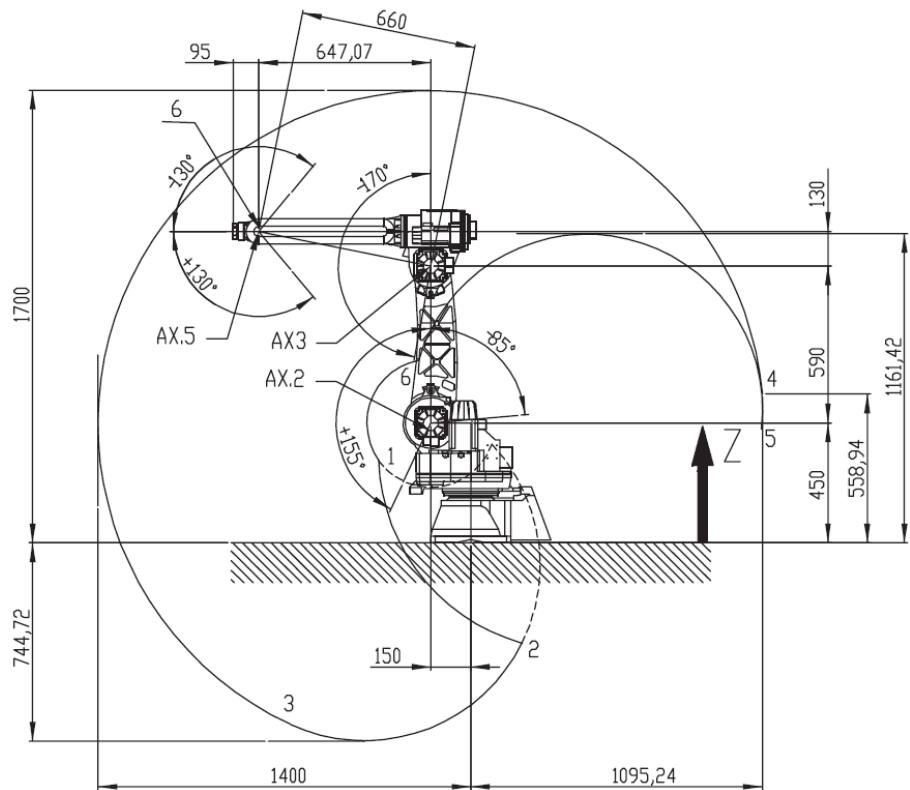


Figura 4.2: Spazio di lavoro del robot Comau Smart-Six

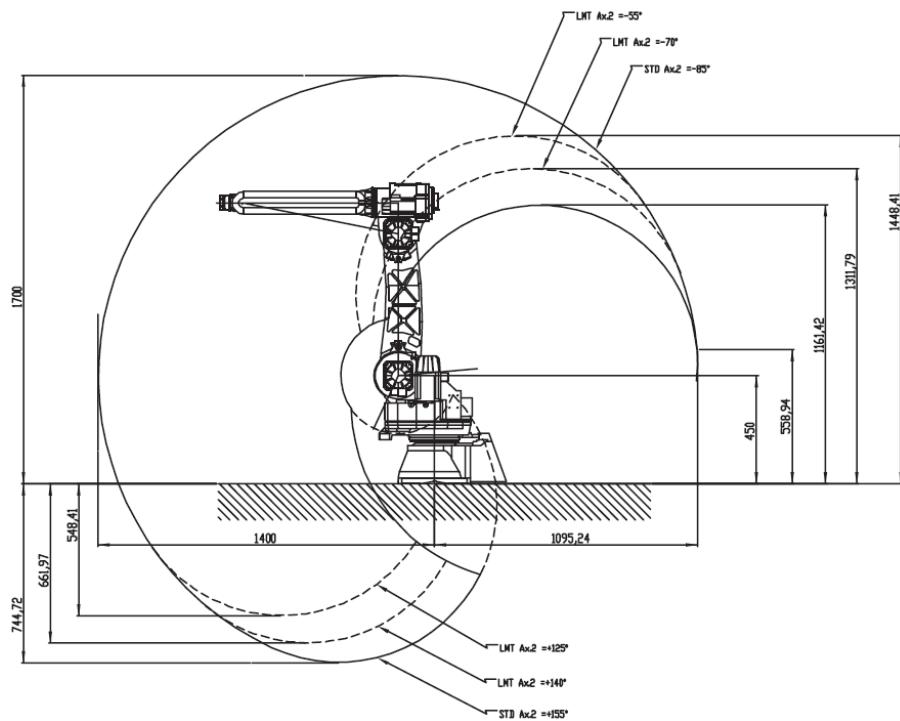


Figura 4.3: Spazio di lavoro del robot Comau Smart-Six

Lo studio della cinematica diretta del robot è di fondamentale importanza in quanto permette di esprimere le configurazioni dell'end-effector in funzione delle configurazioni dei giunti rispetto ad una terna di riferimento. Dunque lo scopo è trovare posizione e orientamento dell'end-effector, i quali sono ricavabili dalla matrice di roto-traslazione $\mathbf{A}_{e_i}^b$ tra l'end-effector e la terna di riferimento (chiamata anche terna base). La matrice di roto-traslazione è costruita nel modo seguente:

$$\mathbf{A}_{e_i}^b(\mathbf{q}_i) = \begin{bmatrix} \mathbf{R}_{e_i}^b(\mathbf{q}_i) & \mathbf{p}_{e_i}^b(\mathbf{q}_i) \\ \mathbf{0}_3^T & 1 \end{bmatrix} \in \mathbb{R}^{4 \times 4} \quad (4.1)$$

Tale matrice è calcolabile facilmente come prodotto di matrice di roto-traslazione tra giunti consecutivi. Le matrici intermedie sono calcolabili grazie alla conoscenza dei parametri di Denavit-Hartenberg (DH) [13] forniti dall'azienda costruttrice del robot, COMAU. In Figura 4.4 è mostrato come sono state fissate le terne per ciascun braccio del robot e in Tabella 5.1 invece sono mostrati i valori dei parametri di DH.

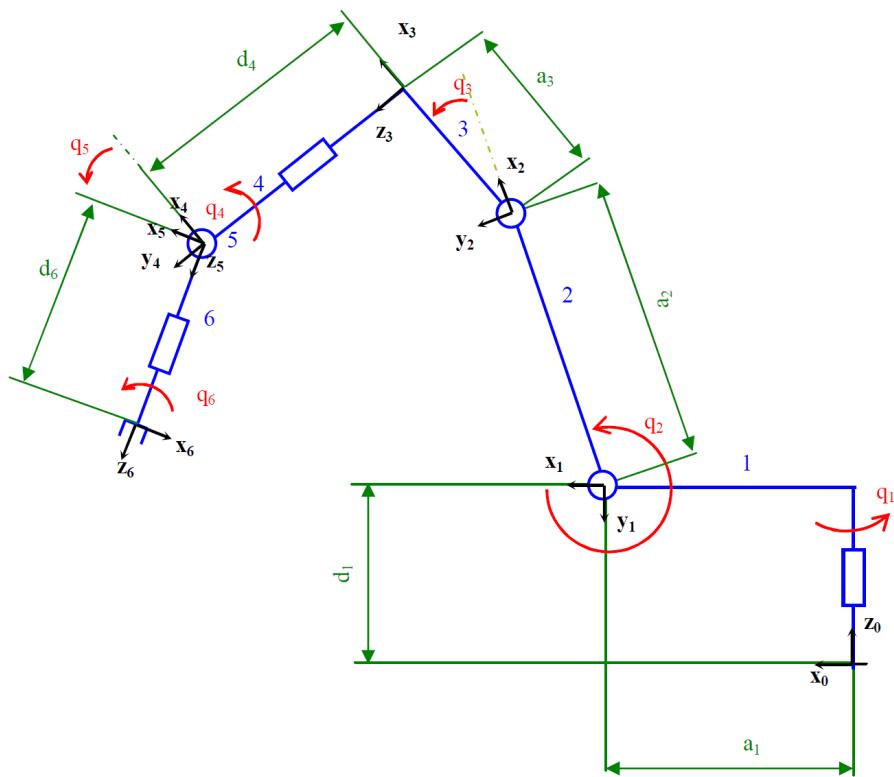


Figura 4.4: Schema del robot Comau Smart-Six con terne fissate sui bracci secondo la convenzione DH

Braccio	a_i (metri)	d_i (metri)	α_i (gradi)	θ_i
1	0.150	0.450	-90	q_1
2	0.590	0	0	q_2
3	0.130	0	-90	q_3
4	0	0.64707	90	q_4
5	0	0	-90	q_5
6	0	0.095	0	q_6

Tabella 4.1: Parametri di Denavit-Hartenberg forniti dall'azienda costruttrice

Tutte queste quantità sono riferite al robot i -esimo.

A partire dalla matrice di roto-traslazione $A_{n_i}^{0_i}(\mathbf{q}_i)$ si può ricavare la matrice di nostro interesse come:

$$\mathbf{A}_{e_i}^b(\mathbf{q}_i) = \mathbf{A}_{0_i}^b \mathbf{A}_{6_i}^{0_i} \mathbf{A}_{6_i}^{e_i} \quad (4.2)$$

con $\mathbf{A}_{0_i}^b$ matrice di roto-traslazione tra la terna 0 del robot i -esimo e la terna base e $\mathbf{A}_{6_i}^{e_i}$ matrice di roto-traslazione tra la terna dell'ultimo giunto del robot e la terna solidale all'end-effector.

È possibile definire lo Jacobiano geometrico del robot come $\mathbf{J}_{g,i} = \begin{bmatrix} \mathbf{c}_{i,1} & \mathbf{c}_{i,2} & \cdots & \mathbf{c}_{i,6} \end{bmatrix}$ con:

$$\mathbf{c}_{i,k} = \begin{bmatrix} \mathbf{z}_{i,k-1} \times (\mathbf{p}_i - \mathbf{o}_{i,k-1}) \\ \mathbf{z}_{i,k-1} \end{bmatrix}, \quad k = 1, 2, \dots, 6 \quad (4.3)$$

dove $\mathbf{z}_{i,k-1}$ è il versore z della terna $k-1$, $\mathbf{o}_{i,k-1}$) è il vettore posizione dell'origine della terna $k-1$ e \mathbf{p}_i è il vettore posizione dell'end-effector.

Dallo Jacobiano geometrico è possibile ottenere quello analitico utilizzando una matrice di trasformazione $\mathbf{T}_{A,i}(\phi_i)$, ovvero:

$$\mathbf{J}_i(\mathbf{q}_i) = \mathbf{T}_{A,i}(\phi_i) \mathbf{J}_{g,i}(\mathbf{q}_i). \quad (4.4)$$

Nel caso di set di angoli di Eulero ZYZ

$$\mathbf{T}_{A,i}(\phi_i) = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{T}_i^{-1}(\phi_i) \end{bmatrix} \in \mathbb{R}^{6 \times 6} \quad (4.5)$$

e

$$\mathbf{T}_i(\phi_i) = \begin{bmatrix} 0 & -\sin(\phi_i) & \cos(\phi_i)\sin(\theta_i) \\ 0 & \cos(\phi_i) & \sin(\phi_i)\sin(\theta_i) \\ 1 & 0 & \cos(\phi_i) \end{bmatrix} \in \mathbb{R}^{3 \times 3} \quad (4.6)$$

Per quanto riguarda, infine, il modello dinamico esso è stato fornito dal Laboratorio di Automatica dell'Università degli Studi di Salerno, sotto forma di regressore in forma ridotta e di vettore dei parametri dinamici, ovvero $\mathbf{Y}_i \in \mathbb{R}^{6 \times 52}$ e $\boldsymbol{\pi} \in \mathbb{R}^{52}$.

4.2 Simulazione in ambiente Matlab

Le simulazioni numeriche prevedono l'esecuzione del seguente task: il trasporto di un oggetto di grandi dimensioni mediante la cooperazione di 4 ($N = 4$) robot Comau Smart-Six montati su base mobile (8 Degrees of freedom). Lo scenario

utilizzato è mostrato in Figura 4.5.

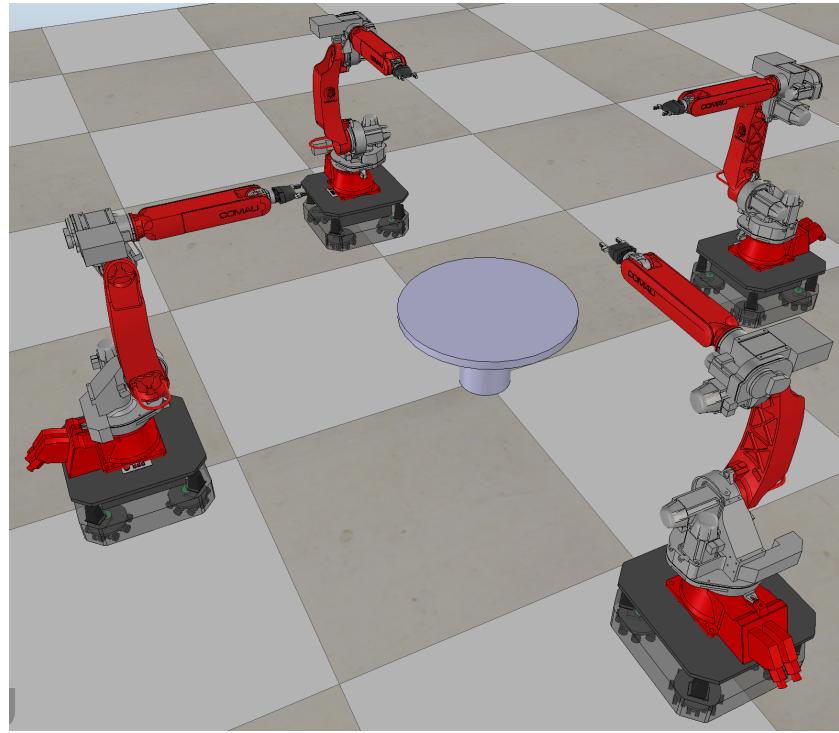


Figura 4.5: Scenario di simulazione

È stato detto che i robot sono montati su base mobile e questo permette di avere due gradi di libertà aggiuntivi e dunque ridondanza. La base mobile viene schematizzata con due giunti prismatici, i cui assi di traslazione sono, rispettivamente, x e y della terna base, i quali sono ortogonali fra loro. In Figura 4.6 sono mostrati i giunti costituenti la base mobile, con le rispettive terne assegnate.

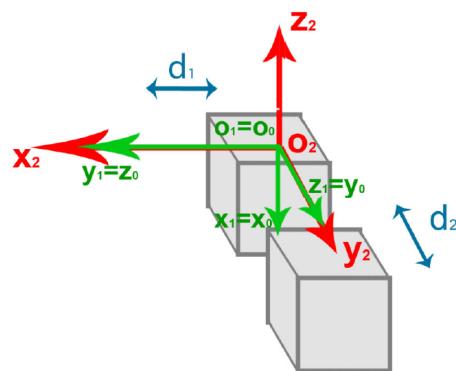


Figura 4.6: Schema della base mobile con relative terne definite in accordo alla convenzione DH

Nella tabella sottostante sono riportati i parametri DH che considerano anche la base mobile.

Braccio	a_i (metri)	d_i (metri)	α_i (gradi)	θ_i
1	0	0	-90	q_7
2	0	0.095	0	q_8
3	q_1	0.450	-90	0
4	q_2	0.450	90	-90
5	0.150	0.450	-90	q_3
6	0.590	0	0	q_4
7	0.130	0	-90	q_5
8	0	0.64707	90	q_6

Tabella 4.2: Parametri di Denavit-Hartenberg della struttura dell'intero robot compresa la base mobile

Mediante i parametri di DH si determina, quindi, la matrice $A_{n_i}^{0_i}$ la cui struttura è analoga per i quattro manipolatori mobili. Per determinare la matrice di interesse $A_{e_i}^b$ è necessario definire le matrici $A_{0_i}^b$ e $A_{e_i}^{8_i}$ per ciascun manipolatore. Considerando lo scenario mostrato in Figura 4.5, l'origine della terna base è stata fissata al centro dell'oggetto (del tavolino da trasportare), con altezza nulla. Le matrici $A_{0_i}^b$ sono quindi ricavate a partire dalla terna base, fissando la terna 0_i sulla base di ciascun robot, come mostrato in Figura 4.7

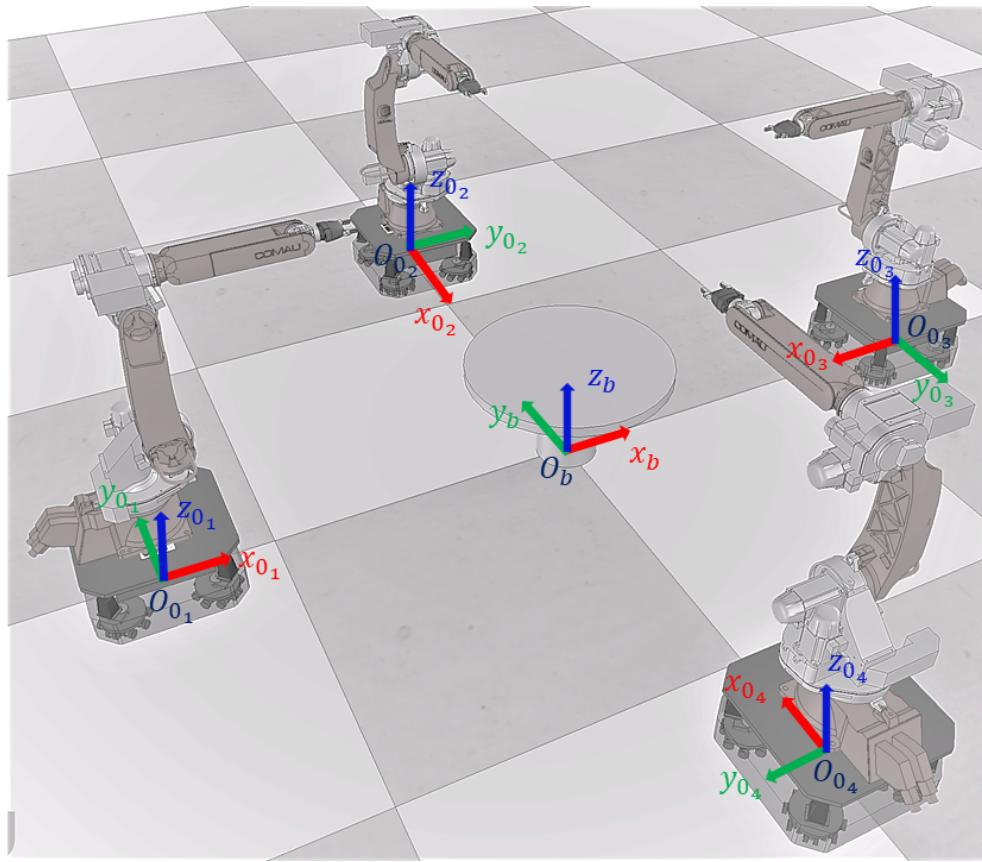


Figura 4.7: Scenario con terna base e terne 0_i dei robot fissate

A partire dalle terne in figura dunque si ha che le matrici \mathbf{A}_{0i}^b assumono i seguenti valori:

$$\mathbf{A}_{01}^b = \begin{bmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.2985 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{A}_{02}^b = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0.2985 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{A}_{03}^b = \begin{bmatrix} -1 & 0 & 0 & 2 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0.2985 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{A}_{04}^b = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & -2 \\ 0 & 0 & 1 & 0.2985 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Per quanto riguarda le terne end-effector, invece, su ciascuna flangia è montata una pinza a due dita. Essa è mostrata nel dettaglio in Figura 4.8 in cui è fissata anche la relativa terna end-effector.

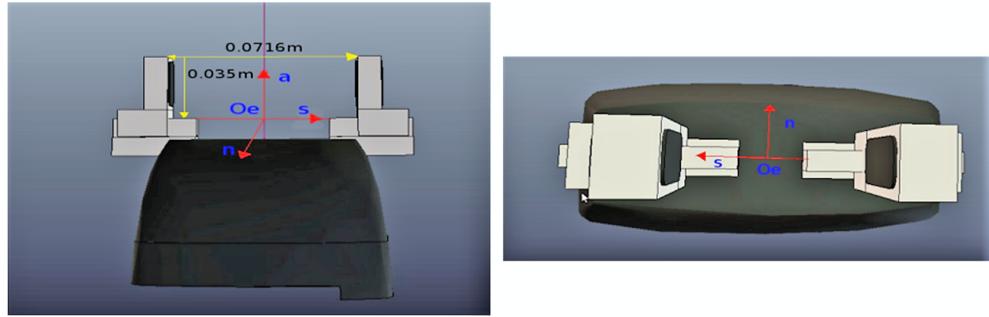


Figura 4.8: End-effector montato sull'ultimo giunto di ciascun robot e relativa terna assegnata

La matrice $\mathbf{A}_{e_i}^{6i}$ è uguale per tutti i robot e vale:

$$\mathbf{A}_{e_i}^{6i} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.0849 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.7)$$

La presenza della base mobile altera anche lo Jacobiano geometrico e di conseguenza quello analitico. Infatti lo Jacobiano geometrico presenta due ulteriori colonne relative ai giunti prismatici:

$$\mathbf{c}_{i,k} = \begin{cases} \begin{bmatrix} \mathbf{z}_{i,k-1} \\ 0 \end{bmatrix} & k = 1, 2 \\ \begin{bmatrix} \mathbf{z}_{i,k-1} \times (\mathbf{p}_i - \mathbf{o}_{i,k-1}) \\ \mathbf{z}_{i,k-1} \end{bmatrix} & k = 3, \dots, 8 \end{cases} \quad (4.8)$$

4.2.1 Task desiderato

Al fine di movimentare l'oggetto, la traiettoria desiderata nel task space può essere espressa in termini di baricentro e formazione. La traiettoria consiste in uno spostamento dell'oggetto di 1.5 metri lungo l'asse x della terna base. La traiettoria può essere decomposta in traiettorie più semplici ed elementari che consentono di raggiungere il comportamento complessivo desiderato. In Figura 4.9 è mostrata la

sequenza di traiettorie semplici di cui si compone quella complessiva. Le immagini mostrano le seguenti fasi del task:

1. configurazione iniziale dei robot; i robot si trovano all'altezza della superficie di presa dell'oggetto. Si suppone che la posizione di partenza sia già questa, altrimenti è possibile portare ciascun robot in questa configurazione eseguendo una traiettoria pianificata nello spazio giunti di ciascun robot, senza considerare, solo per questa parte di traiettoria, baricentro e formazione;
2. i robot si avvicinano al fine di prendere l'oggetto dunque la formazione si stringe e il baricentro resta costante;
3. i robot sollevano l'oggetto. Il baricentro viene alzato, la formazione resta costante;
4. trasporto dell'oggetto. Il baricentro varia lungo l'asse x, la formazione resta costante;
5. i robot riportano la base dell'oggetto sul pavimento. Il baricentro viene abbassato, la formazione resta costante.

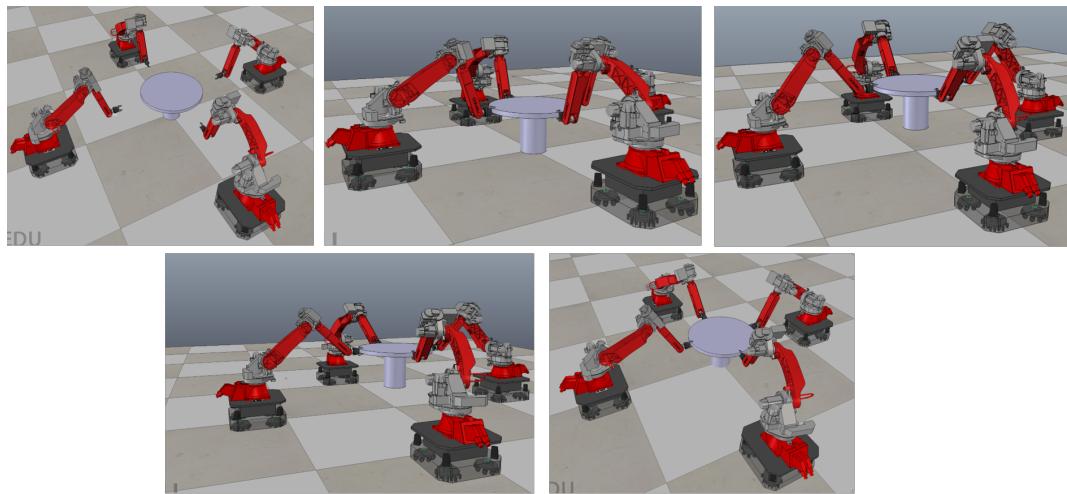


Figura 4.9: Traiettorie di cui si compone il task complessivo

Ciascuna traiettoria è stata pianificata nello spazio del task, mediante profilo trapezoidale, in modo da ottenere continuità in termini di velocità e imponendo la

durata dell'esecuzione di ciascuna di esse. La traiettoria ottenuta dalla pianificazione è mostrata in Figura 4.10.

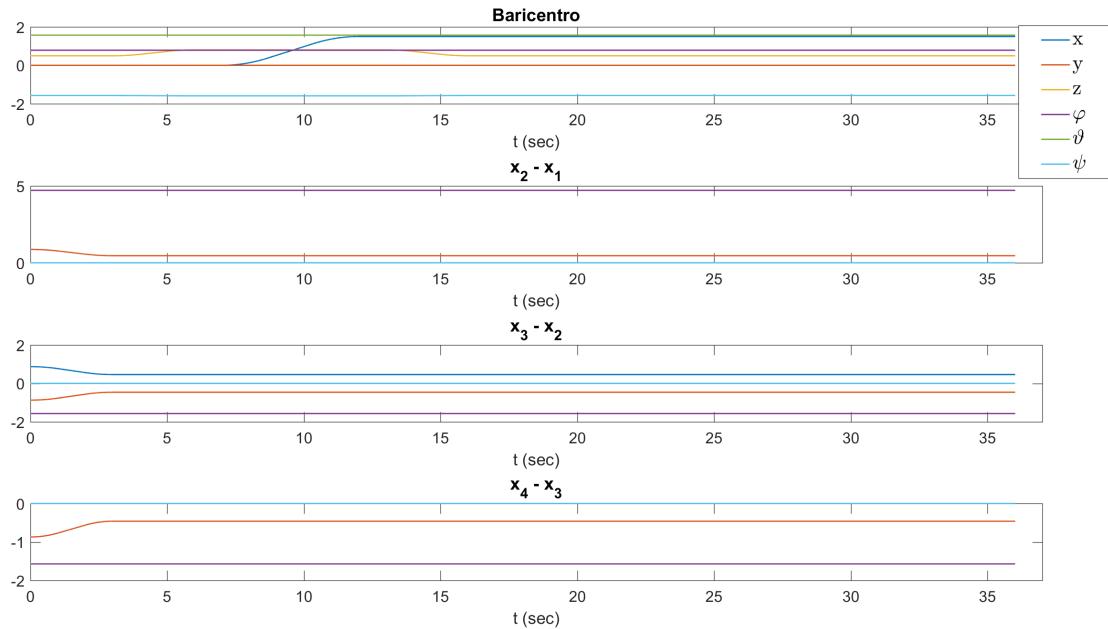


Figura 4.10: Traiettorie desiderate pianificate nello spazio del task

La traiettoria ha una durata totale di 16 secondi, dopo i quali è stata aggiunta una traiettoria costante per raggiungere la fase di regime e osservare il comportamento del sistema. Ciascuna sotto-traiettoria, in realtà, è intervallata con la successiva da una traiettoria costante, di regime, della durata di 1 secondo.

È stato detto nel capitolo 2 che la comunicazione, ovvero lo scambio di informazioni tra i robot, è modellata tramite grafo. La topologia qui considerata è rappresentata da un grafo non orientato e connesso, mostrato nella Figura 4.11.

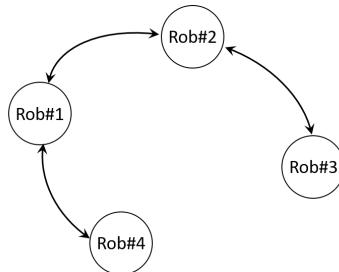


Figura 4.11: Topologia del grafo di comunicazione

4.2.2 Implementazione dell'algoritmo di controllo

La fase simulativa ha previsto la realizzazione di uno script Matlab mediante il quale sono coordinati i quattro manipolatori mobili. Lo script consta di una prima fase di inizializzazione del setup e di una successiva fase di evoluzione dell'algoritmo di controllo. In particolare, nella fase preliminare si procede alla definizione della topologia del grafo di comunicazione, alla generazione della traiettoria desiderata e all'inizializzazione dello stato di ciascun robot. L'algoritmo evolve con un tempo di campionamento T pari a 2 msec. Ciascun robot è rappresentato come un dato strutturato contenente le informazioni che lo caratterizzano, quali posizioni, velocità e accelerazioni correnti dei giunti, stima dello stato globale del sistema e vettore dei parametri dinamici aggiornato, forze calcolate e forze interne stimate; lo script gestisce, dunque, una collezione di tali dati strutturati e ne evolve lo stato ad ogni passo di campionamento. In particolare l'aggiornamento del robot i -esimo avviene secondo le seguenti fasi.

1. Aggiornamento della stima dello stato globale del sistema ${}^i\hat{\mathbf{x}}$.
2. Calcolo della legge di controllo globale ${}^i\hat{\mathbf{u}}$ in base alla corrente traiettoria desiderata nel task space.
3. Solo nel caso del controllo di interazione è presente la stima da parte del manipolatore i -esimo del proprio contributo agli stress interni all'oggetto trasportato.
4. Calcolo dell'ingresso di controllo locale $\boldsymbol{\tau}_i$ mediante legge di controllo adattativa. In tale legge, poiché i manipolatori sono ridondanti, si è utilizzato il contributo $\dot{\mathbf{q}}_{n,i}$ per soddisfare vincoli secondari. Più in dettaglio esso è utilizzato per massimizzare localmente la distanza dai limiti meccanici dei giunti e per minimizzare i moti interni. Da un punto di vista formale, ciò è espresso proiettando un vettore di velocità arbitrarie $\dot{\mathbf{q}}_{0,i}$, nel nullo dello Jacobiano \mathbf{J}_i ovvero:

$$\dot{\mathbf{q}}_{n,i} = (\mathbf{I}_{n \times n} - \mathbf{J}_i^\dagger \mathbf{J}_i) \dot{\mathbf{q}}_{0,i} \quad (4.9)$$

dove $\dot{\mathbf{q}}_{0,i}$ è progettato mediante una tecnica a gradiente ed è scelto in modo da minimizzare localmente una funzione obiettivo $\mathbf{w}(\dot{\mathbf{q}}_i)$:

$$\dot{\mathbf{q}}_{0,i} = -\mathbf{k}_n \frac{\partial \mathbf{w}_i(\mathbf{q})}{\partial \mathbf{q}_i} \quad (4.10)$$

con $\mathbf{k}_n \in \mathbb{R}^+$. La funzione $\mathbf{w}(\dot{\mathbf{q}}_i)$ è definita nel seguente modo:

$$w(\dot{\mathbf{q}}_i) = \sum_{j=1}^8 \left(\frac{q_{i,j} - q_{i,jmedio}}{q_{i,jMax} - q_{i,jmin}} \right)^2 \quad (4.11)$$

dove $q_{i,j}$, $q_{i,jmedio}$, $q_{i,jMax}$, $q_{i,jmin}$ rappresentano rispettivamente la posizione attuale, il centro corsa, il limite superiore e il limite inferiore del j -esimo giunto del robot i . Per il calcolo dell'ingresso di controllo locale nel caso di controllo di interazione viene calcolato anche il contributo aggiuntivo in coppia $\Delta \boldsymbol{\tau}_i$.

5. Aggiornamento del vettore dei parametri dinamici $\hat{\boldsymbol{\pi}}_i$ sulla base del corrente errore $\dot{\tilde{\mathbf{q}}}_{\sigma,i}$.
6. Evoluzione dello stato del robot in accordo all'ingresso di controllo locale. Le accelerazioni ai giunti sono dedotte mediante il modello di dinamica diretta ovvero:

$$\ddot{\mathbf{q}}_i = \mathbf{M}_i^{-1}(\mathbf{q}_i)(\boldsymbol{\tau}_i - \mathbf{C}_i(\mathbf{q}, \dot{\mathbf{q}}_i)\dot{\mathbf{q}}_i - \mathbf{F}_i\dot{\mathbf{q}}_i - \mathbf{g}_i(\mathbf{q}_i) - \mathbf{J}_i\mathbf{h}_i) \quad (4.12)$$

Per emulare la conoscenza approssimata del modello dinamico dei manipolatori e per poter testare quindi la bontà dell'adattamento dei parametri, il modello dinamico usato per calcolare l'ingresso di controllo locale (punto 5) differisce dal modello usato per evolvere lo stato del robot (punto 6). In particolare è stata considerata un'incertezza iniziale sui parametri dinamici generata in accordo ad una distribuzione uniforme, e pari massimo al 15% del valore reale dei parametri. Anche l'errore iniziale di stima dello stato globale del sistema è diverso da zero e in particolare l'incertezza è generata secondo una distribuzione uniforme e può essere pari massimo al 20% dei valori reali.

4.2.3 Risultati ottenuti

A partire da quanto descritto, sono state effettuate diverse simulazioni utilizzando le architetture progettate. Queste sono state utili sia per la taratura dei parametri, sia per verificare la bontà di quanto progettato e sia per evidenziare le differenze tra un controllo puramente cinematico e l'utilità del controllo di interazione. Sono stati considerati i seguenti parametri per effettuare gli studi:

- l'errore in norma sull'inseguimento del task in posizione $\|\tilde{\sigma}\|$ e velocità $\|\dot{\tilde{\sigma}}\|$;
- l'errore in norma sulla stima dello stato dell'intero sistema $\|\tilde{x}^*\|$;
- le forze interne relative a ciascun manipolatore $\mathbf{h}_{int,i}$;
- l'errore di stima delle forze interne $\tilde{\mathbf{h}}_{int,i}$;
- le coppie di attuazione generate $\boldsymbol{\tau}_i$;
- l'errore di stima delle forze impresse sull'oggetto dagli altri robot del team.

Una prima fase ha riguardato la taratura dei guadagni facendoli variare singolarmente e vedendo come la variazione di essi contribuisse al miglioramento o peggioramento delle variabili considerate.

4.2.4 Risultati sul controllo puramente cinematico

I guadagni da tarare sono i seguenti: k_o , k_q , k_n , \mathbf{K}_π , k_σ .

Consideriamo per primo il guadagno k_o che è il guadagno relativo alla legge di aggiornamento della stima dello stato del sistema complessivo effettuata dall'osservatore globale (3.9).

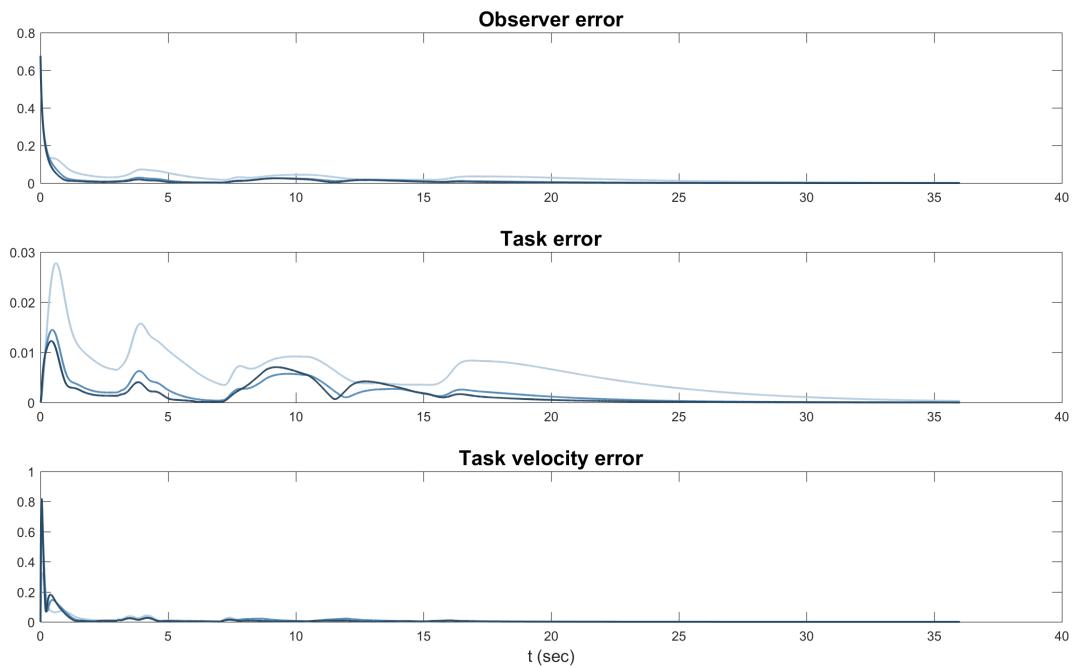


Figura 4.12: Errore dell'osservatore (sopra), errore sul task in posizione (in mezzo), errore sul task in velocità (sotto) al variare del guadagno k_o il quale cresce con lo scurirsi dei grafici. I valori utilizzati sono 2, 6, 10

Nella Figura 4.12 è possibile visualizzare gli andamenti degli errori dell'osservatore e dell'inseguimento del task in posizione e velocità al variare del guadagno k_o il quale assume i valori di 2, 6 e 8 che corrispondono rispettivamente ai grafici dal più chiaro al più scuro. Si può notare che all'aumentare di tale guadagno le prestazioni tendono a migliorare, dunque viene scelto un valore alto di tale guadagno precisamente $k_o = 10$. Nei primi istanti temporali in questi grafici e in quelli successivi sarà possibile notare che l'errore è più alto rispetto al resto della traiettoria e questo comportamento è dovuto al fatto che l'errore di stima dello stato complessivo del sistema e l'errore iniziale sui parametri dinamici, vengono recuperati dopo questi istanti di tempo, mentre sono particolarmente significativi nella parte iniziale del task eseguito.

Passiamo ora ad analizzare il comportamento del sistema nel caso in cui a variare è il guadagno k_q . Ricordiamo che k_q viene utilizzato nella legge di controllo per introdurre un'azione lineare stabilizzante di tipo PD sull'errore di inseguimento. Dalla Figura 4.13 si può notare che più è grande tale guadagno (dal grafico più chiaro al grafico più scuro k_q assume i valori di 60, 80 e 100), rispettando sempre

i limiti della stabilità, più il comportamento degli errori migliora, essendo più piccoli in norma per tutta la durata della traiettoria. Pertanto il valore scelto per tale guadagno è pari a 100.

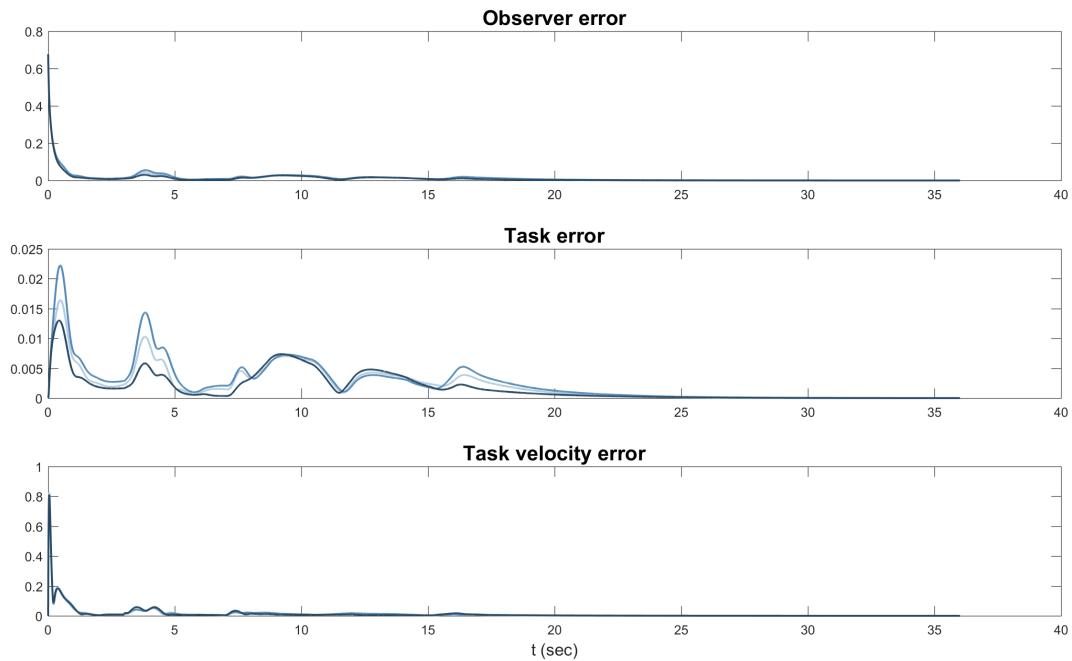


Figura 4.13: Errore dell'osservatore (sopra), errore sul task in posizione (in mezzo), errore sul task in velocità (sotto), al variare del guadagno k_q il quale cresce con lo scurirsi dei grafici. I valori utilizzati sono 60, 80, 100

Continuiamo ancora con la progettazione del guadagno k_σ , dal quale dipende la velocità di convergenza dell'errore di inseguimento del task desiderato (Figura 4.14).

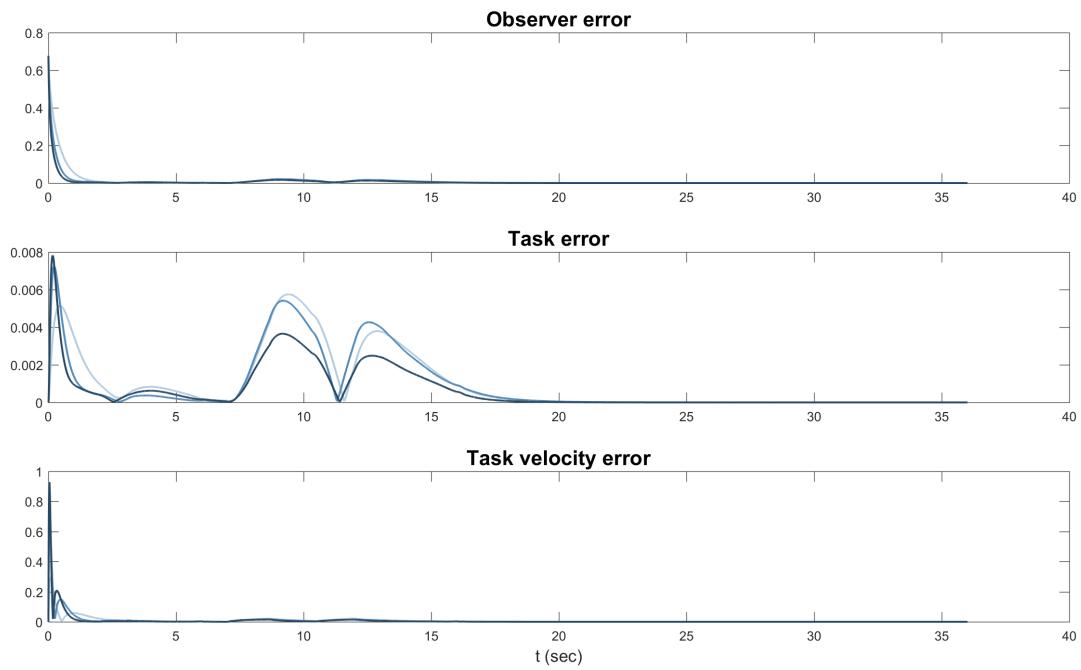


Figura 4.14: Errore dell'osservatore (sopra), errore sul task in posizione (in mezzo), errore sul task in velocità (sotto), al variare del guadagno k_σ il quale cresce con lo scurirsi dei grafici. I valori utilizzati sono 2, 6, 10

Anche in questo caso si può osservare, soprattutto per quanto riguarda l'errore di inseguimento del task in termini di posizione, che all'aumentare del guadagno considerato, il quale aumenta dal grafico più chiaro al grafico più scuro assumendo rispettivamente i valori di 2, 6, 10, le prestazioni migliorano. Quindi la scelta che è stata effettuata è $k_\sigma = 8$ (per ragioni di stabilità non si è scelto un valore maggiore).

Infine resta da progettare la matrice diagonale di guadagni \mathbf{K}_π scegliendo di utilizzare valori sulla diagonale uguali tra loro. La matrice \mathbf{K}_π determina la velocità di adattamento dei parametri dinamici utilizzati, a quelli reali del robot. È possibile notare il comportamento del sistema sulla base della variazione di questa matrice in Figura 4.15. I valori sulla diagonale della matrice, nei grafici mostrati, sono rispettivamente dal più chiaro al più scuro 1, $1/50$, $1/100$ e $1/150$.

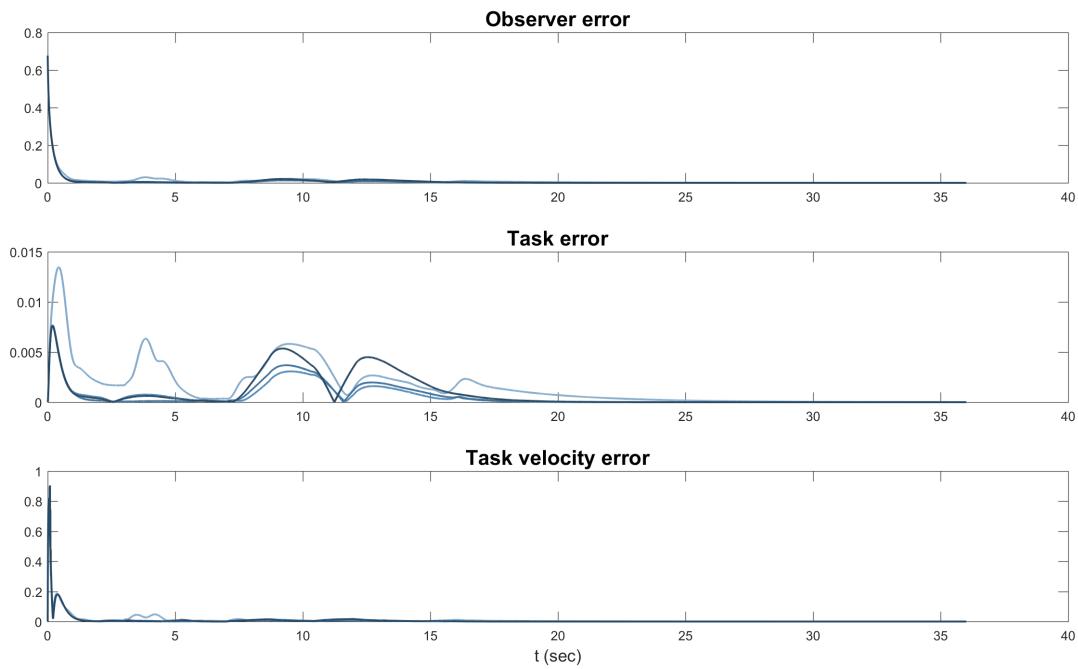


Figura 4.15: Errore dell’osservatore (sopra), errore sul task in posizione (in mezzo), errore sul task in velocità (sotto), al variare del guadagno k_π il quale cresce con lo scurirsi dei grafici. I valori utilizzati sono 1, 1/50, 1/100, 1/150

Al variare di questa matrice non si riesce a notare grande variazione di prestazione, in quanto l’adattamento avviene soprattutto nei primi istanti temporali e in quanto l’errore sui parametri non è molto grande. Sceglieremo di utilizzare $k_\pi = 1/50$.

Mentre tutti questi guadagni sono stati progettati tenendo conto delle prestazioni del sistema, il guadagno k_n è stato progettato in modo da tenere i giunti dei robot il più lontano possibile dai propri limiti, ma solo quando realmente si stanno avvicinando ad essi. Infatti è inutile avere un contributo in velocità diverso da zero che viene proiettato nel nullo dello jacobiano e quindi non produce velocità nello spazio operativo per tutta la durata della traiettoria se i giunti non si stanno allontanando dal proprio centro. Per questo motivo è stato deciso di utilizzare una legge parabolica dipendente dalla distanza dei giunti dal proprio centro.

In seguito alla taratura di tutti questi parametri, è utile osservare quali sono le coppie richieste per eseguire tale task (Figura 4.16).

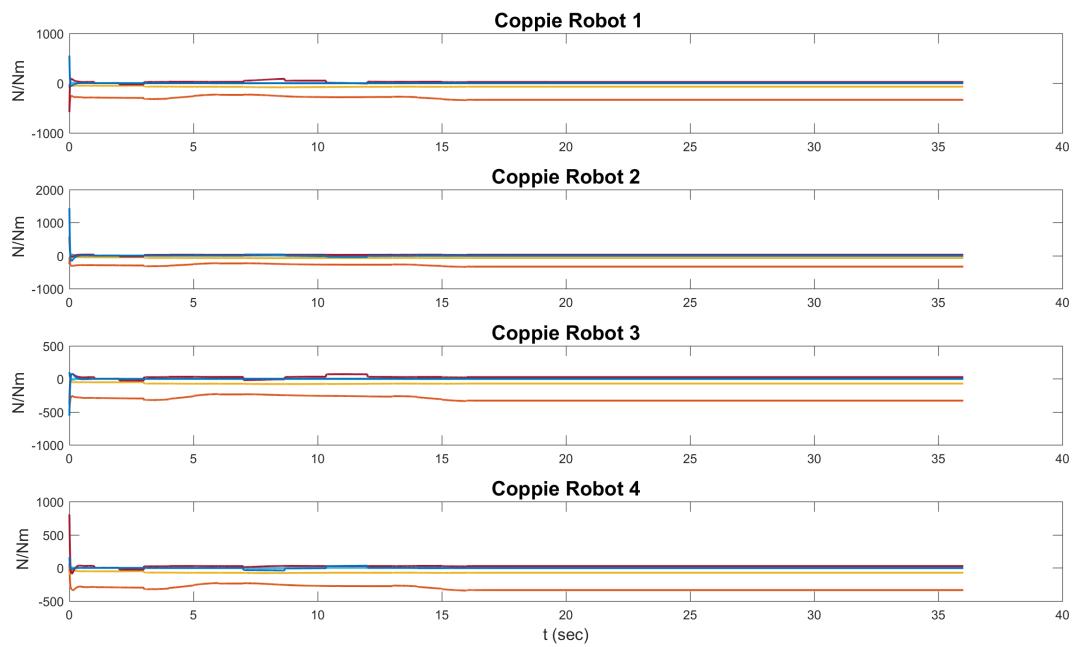


Figura 4.16: Coppie richieste ai giunti per ciascuno dei quattro manipolatori

Per quanto riguarda le prestazioni al variare della connettività del grafo, è possibile fare riferimento alla Figura 4.17. In particolare, andando dal celeste al blu, la connettività del grafo aumenta. Si nota come per grafi maggiormente connessi, l'errore di inseguimento del task viene recuperato più velocemente. Infatti considerando ad esempio il nodo 1 connesso ad altri 3 nodi, questo riceve ad ogni passo di campionamento la stima dello stato del sistema effettuata da tutti gli altri. In realtà questa stima è costituita dallo stato effettivo degli altri nodi, in quanto essendo il grafo fortemente connesso, ogni nodo comunica il proprio stato effettivo a tutti gli altri.

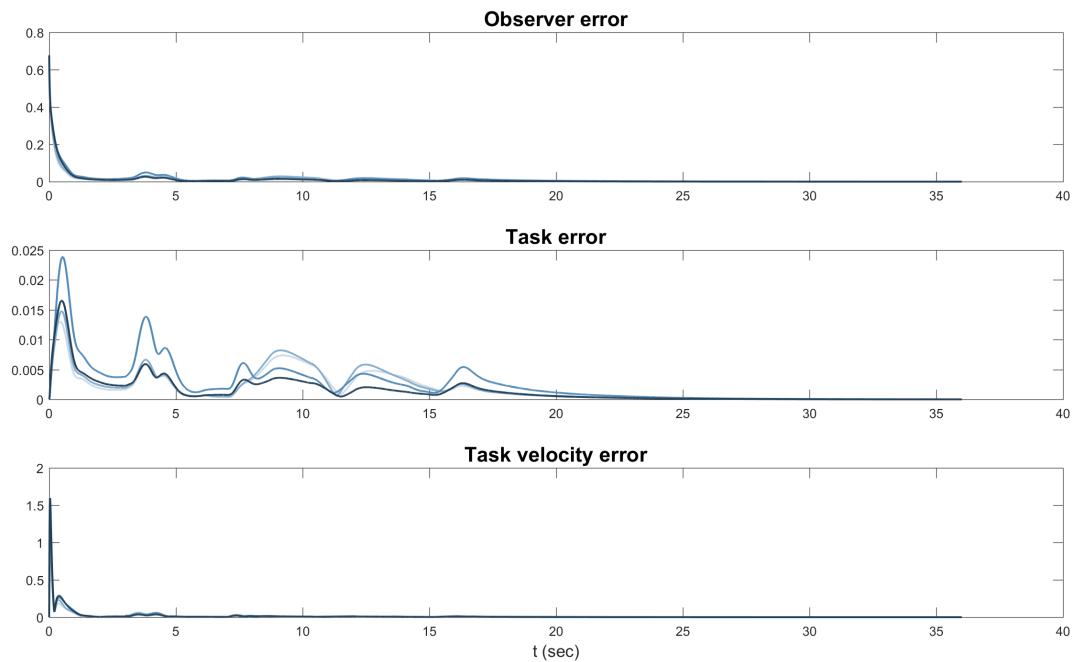


Figura 4.17: Errore dell’osservatore (sopra), errore sul task in posizione (in mezzo), errore sul task in velocità (sotto), al variare della connettività del grafo

4.2.5 Risultati sul controllo di interazione

Al fine di mostrare che il controllo puramente cinematico non è sufficiente nel caso di manipolazione di oggetti rigidi, mostriamo le forze interne che vengono generate durante la fase di trasporto dell’oggetto, notando che si ottengono quantità non accettabili e che porterebbero alla rottura di questo o dei robot.

Il contatto end-effector oggetto è stato modellato attraverso un sistema molla smorzatore. Per implementare ciò in termini matematici, si può scrivere:

$$\mathbf{h}_i = \begin{bmatrix} \mathbf{K}_P(\mathbf{p}_i - \mathbf{p}_o) + \mathbf{K}_D(\dot{\mathbf{p}}_i - \dot{\mathbf{p}}_o) \\ \mathbf{0}_3 \end{bmatrix} \quad (4.13)$$

dove \mathbf{K}_P rappresenta la costante elastica ed è stata posta pari a 1000, mentre \mathbf{K}_D rappresenta la costante di smorzamento ed è stata posta pari a $6\sqrt{\mathbf{K}_P \cdot \text{massa}}$. Le costanti \mathbf{K}_P e \mathbf{K}_D sono state tarate in modo da non avere un numero troppo elevato di oscillazioni nelle coppie generate, ma anche in modo da rendere abbastanza rigido il grasping.

Consideriamo in particolare un oggetto di massa 3 Kg da trasportare. Viene ipotizzata la conoscenza non perfetta della geometria dell'oggetto, con conseguente errore della pianificazione delle variabili di task relative. Pertanto per il controllo di interazione, dall'istante in cui l'oggetto viene afferrato, vengono considerate solo le variabili di task relative al baricentro ($\sigma_1(t)$). Inoltre in tutta la simulazione viene ignorato l'orientamento, in particolare i momenti che si generano e il controllo di essi.

Le forze interne che si generano nel caso in cui esse non vengono controllate sono mostrate in Figura 4.18 ed è chiaro che tali valori non possono essere accettati in quanto potrebbero causare la rottura dell'oggetto o dei manipolatori.

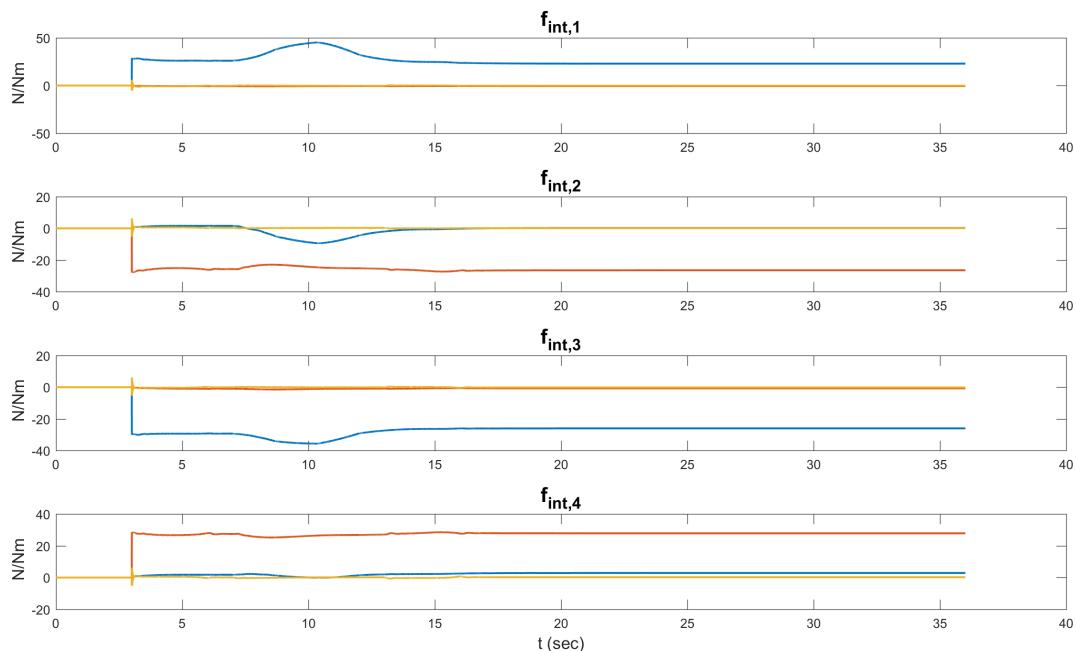


Figura 4.18: Forze interne nel caso in cui le forze non vengono controllate durante il trasporto di un oggetto rigido

Si evince dunque che è necessario controllare le forze per poter portare a termine un task di grasping rigido.

Aggiungendo il controllo delle forze interne (Paragrafo 3.3.4), infatti, si nota una diminuzione delle loro intensità, ma soprattutto si vede come tali forze convergano subito ad un valore desiderato che in questo caso è stato posto pari a zero

(Figura 4.19). A causa della conoscenza non perfetta della geometria dell'oggetto, dell'errore iniziale di stima dello stato del sistema e dell'errore sulla conoscenza dei parametri dinamici di ciascun robot, si ha che le forze interne generalizzate sono molto grandi nella fase iniziale del task. È stato necessario progettare nuovamente i guadagni da utilizzare valutando le prestazioni come mostrato nel paragrafo precedente. In particolare le scelte effettuate sono le seguenti: $k_\sigma = 8$, $k_q = 80$, $\mathbf{K}_\pi = 1/50$.

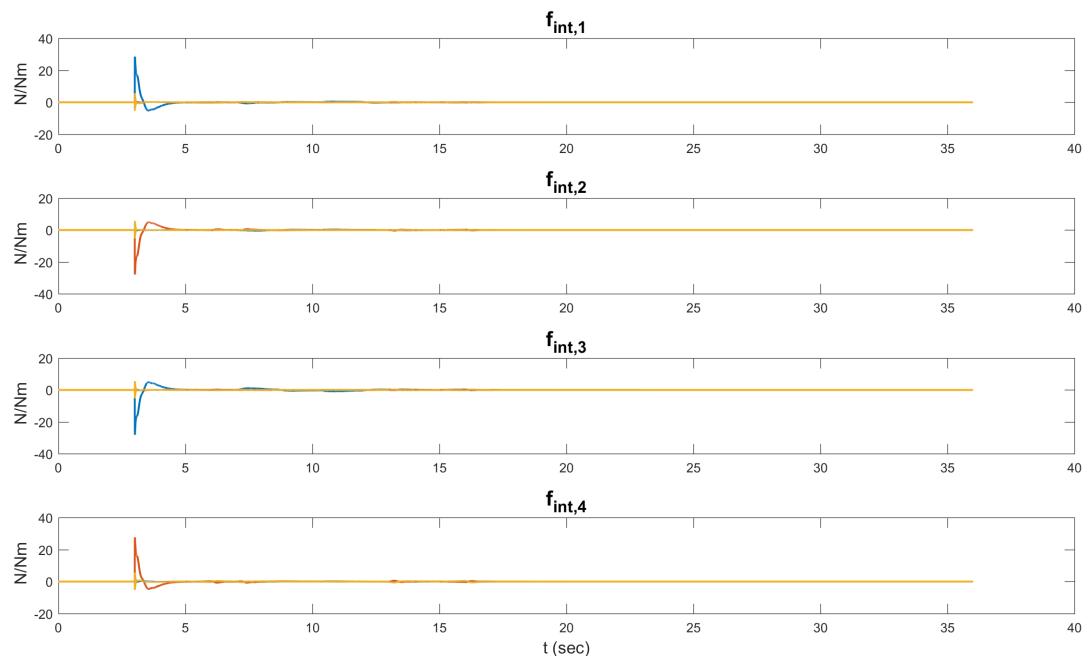
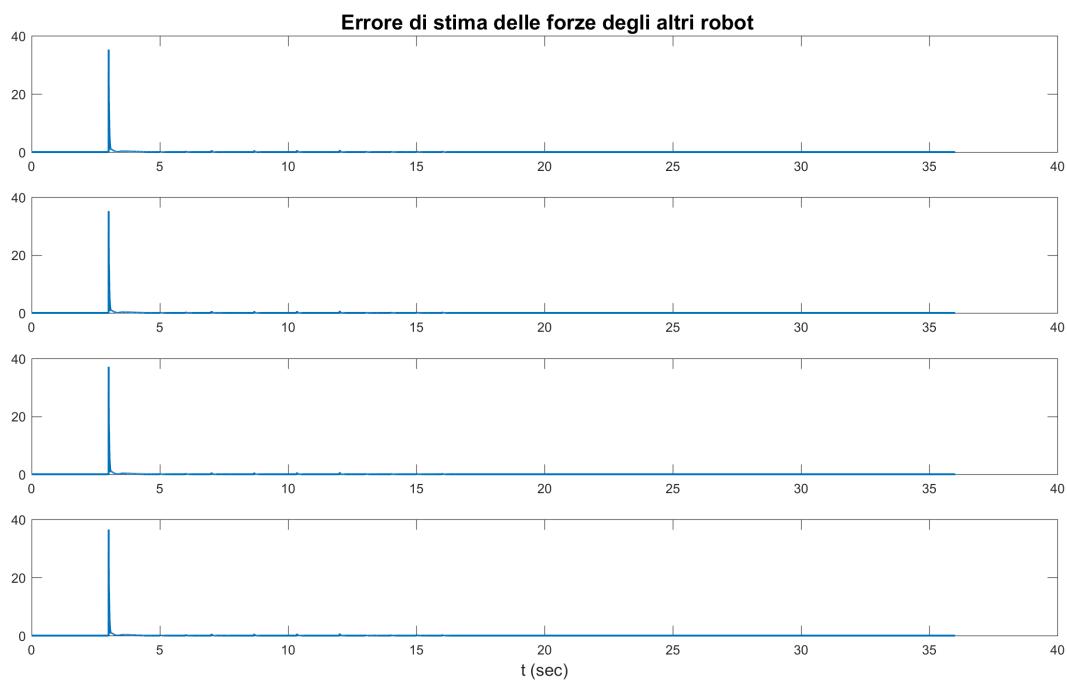


Figura 4.19: Forze interne (Equazione 3.26) nel caso in cui le forze vengono controllate durante il trasporto di un oggetto rigido

Valutiamo poi l'errore di stima delle forze impresse dagli altri robot sull'oggetto, dato che ciascun robot utilizza solo informazioni locali per stimare tali valori e non conosce le forze effettive che gli altri componenti del team stanno applicando. Dal grafico di Figura 4.20 si evince che l'errore di stima è alto all'inizio dell'interazione ma viene subito recuperato, compensando dunque nel modo giusto le forze interne e riducendo a zero gli stress interni all'oggetto trasportato.



*Figura 4.20: Errore di stima delle forze impresse dagli altri robot sull'oggetto
(Equazione 3.27)*

Le coppie di attuazione ai giunti richieste per l'intera durata del task sono mostrate in Figura 4.21.

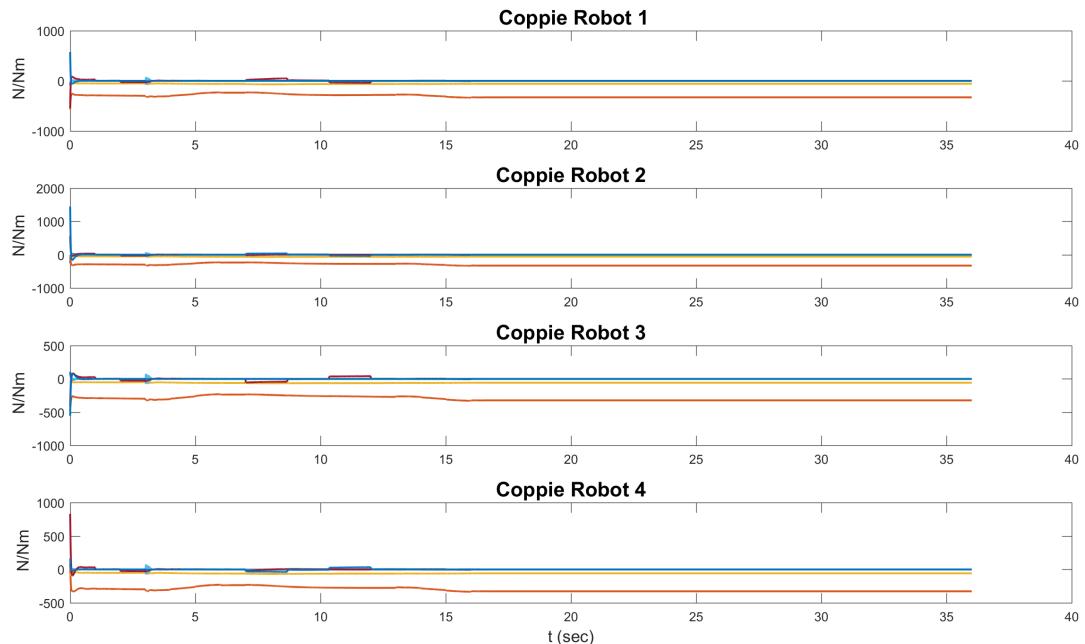


Figura 4.21: Coppie di attuazione per ciascuno dei quattro manipolatori necessarie per l'intera esecuzione del task

In seguito a tutte le scelte effettuate, l'errore sull'inseguimento del task e l'errore commesso dallo stimatore dello stato del sistema sono mostrati in Figura 4.22.

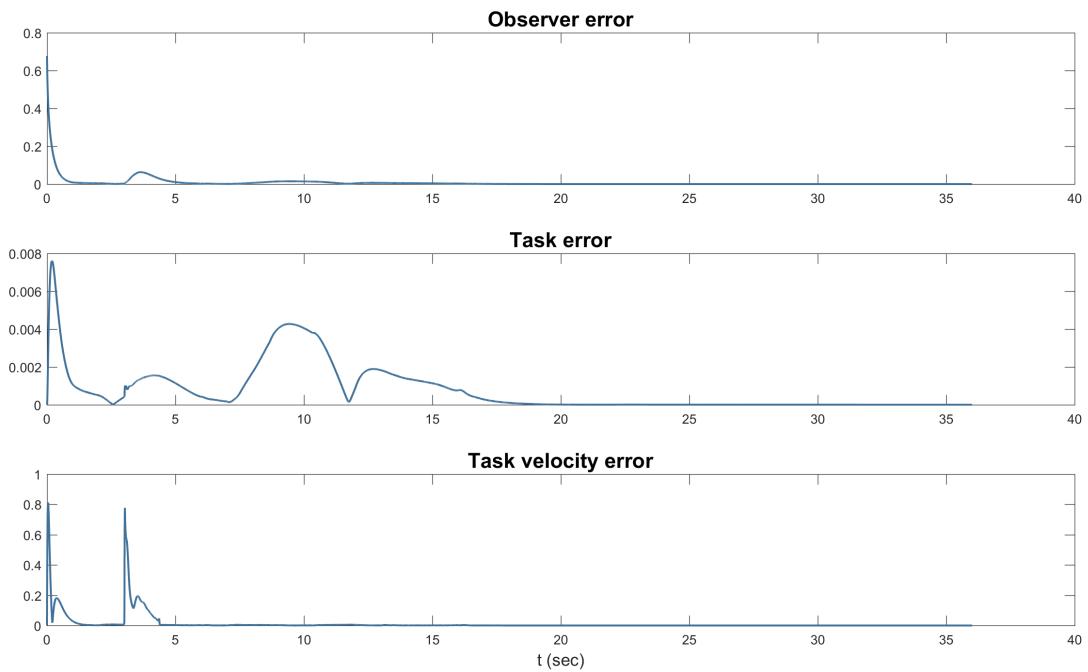


Figura 4.22: Errore dell'osservatore (sopra), errore sul task in posizione (in mezzo), errore sul task in velocità (sotto)

Capitolo 5

Esperimenti

L’obiettivo di questa parte sperimentale è progettare un’architettura per gestire la sicurezza di una persona all’interno di uno scenario multi-robot, in assenza di un’unità di controllo centrale. La presenza simultanea di persone e robot all’interno di stesse aree, permette di avere celle di lavoro estremamente flessibili che possono essere facilmente riconfigurate. In questi scenari non per forza l’uomo deve collaborare strettamente con i robot, infatti possono anche condividere solamente gli stessi spazi. Per garantire la sicurezza delle persone possono essere adottati ad esempio algoritmi di l’obstacle avoidance e si possono ad esempio desiderare comportamenti repulsivi dei robot nei confronti della persona. Bisogna notare che ogni robot rappresenta un’unità a sé stante nel senso che ognuno è dotato di un proprio controllo che stabilisce il comportamento desiderato al fine di ottenere un certo comportamento complessivo del sistema. Lo scenario analizzato in questo lavoro di tesi si compone di robot collaborativi che svolgono un determinato task, di un robot di supervisione e di un solo operatore umano che può invadere lo spazio di lavoro dei robot. Per quanto riguarda i robot cooperanti, essi hanno il compito di trasportare un oggetto, anche se in modo limitato, dato lo spazio ridotto presente all’interno della cella di lavoro. Il robot supervisore invece ha il ruolo di ispezionare in modo continuo tutte le aree della cella di lavoro e in caso di presenza di una persona, esso ha il ruolo di seguirle e tenerne sempre traccia. Questo è richiesto in quanto è esso a gestire la sicurezza della persona. Per questo motivo esso modifica, sulla base della distanza tra l’operatore e l’og-

getto trasportato dai robot cooperanti, il task di tali robot in modo da mantenere tra essi sempre una certa distanza di sicurezza. L'assenza di un'unità centrale e la non predittività del comportamento della persona fanno sì che sia necessaria una pianificazione online del moto di tutti i robot; del robot supervisore perché non si conosce in anticipo come si muoverà la persona per inseguirla, e dei robot cooperanti in quanto non si sa quanto la persona si avvicinerà all'oggetto che stanno trasportando e come il robot supervisore provvederà a modificare il task che stanno eseguendo.

Lo scenario complessivo è mostrato in Figura 5.1 e Figure 5.2, in cui si nota la presenza di due robot Comau Smart-Six che sono i robot cooperanti, un robot UR10 che è il robot supervisore ed una persona che invade lo spazio di lavoro (questo lavoro di tesi è stato sviluppato considerando la possibilità che una sola persona per volta possa entrare nella cella di lavoro).

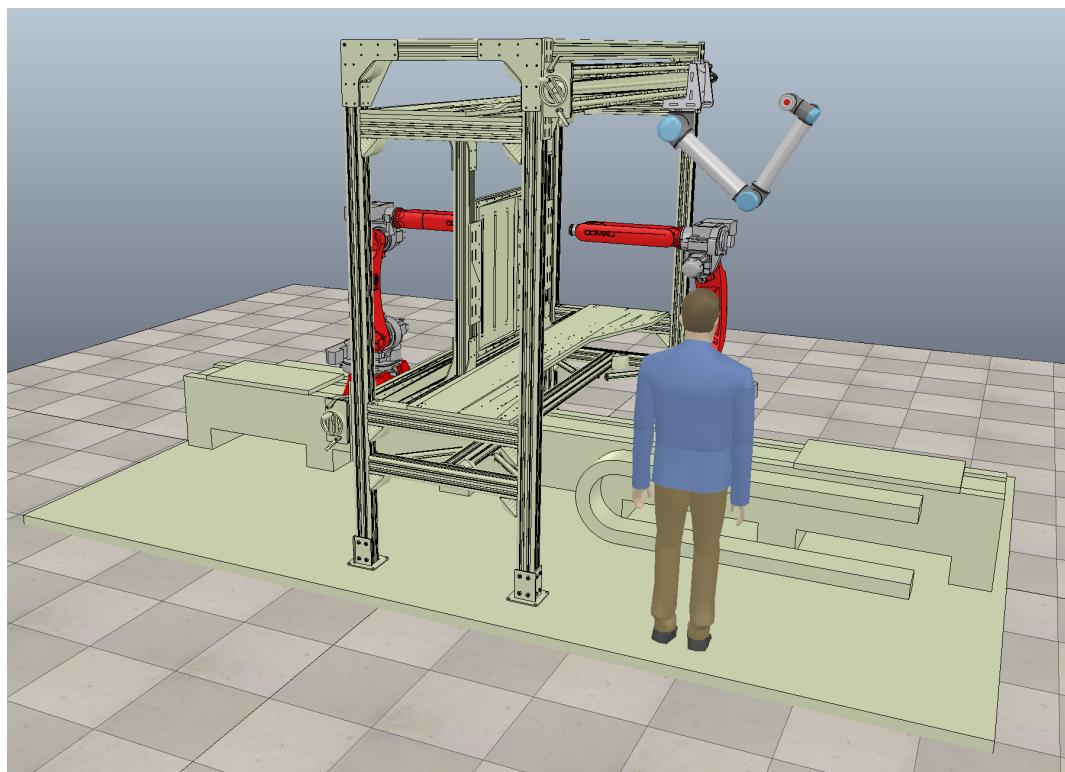


Figura 5.1: Cella robotica del Laboratorio di Automatica dell'Università degli Studi di Salerno ricostruita in V-REP rappresentante lo scenario considerato nel lavoro di tesi (robot cooperanti Comau Smart-Six, robot supervisore UR10 e operatore umano)



Figura 5.2: Cella robotica del Laboratorio di Automatica dell'Università degli Studi di Salerno rappresentante lo scenario considerato nel lavoro di tesi (robot cooperanti Comau Smart-Six, robot supervisore UR10 e operatore umano)

La parte sperimentale è stata pensata e sviluppata a partire da quella simulativa sfruttando un'implementazione del controllo puramente cinematico nella realtà relativa ad un precedente lavoro di tesi [7]. L'obiettivo è utilizzare i due robot Comau Smart-Six per trasportare un oggetto di grandi dimensioni che non può essere sollevato da un solo robot. L'oggetto viene movimentato nella parte centrale della struttura che divide i robot, la quale nella realtà è stata in parte smontata per permettere appunto lo svolgimento del task. Il robot UR10 nel frattempo supervisiona la cella e mette in sicurezza la persona che è all'interno, garantendo sempre che esso stesso e l'oggetto trasportato mantengano una distanza di sicurezza dall'operatore.

La soluzione proposta è stata testata prima in simulazione e poi nella realtà e alla fine del capitolo sono mostrati i risultati ottenuti.

5.1 Descrizione dello scenario

Come si può osservare in Figura 5.3, all'interno del Laboratorio di Automatica dell'Università degli Studi di Salerno è presente una cella di lavoro all'interno della quale sono collocati:

- due robot Comau Smart-Six montati su lati opposti di una slitta mobile;
- un robot UR10 montato su una struttura e quindi ad un'altezza di circa 2 metri.

A bordo del robot UR10, ovvero all'estremità del polso, è montata una Kinect (Paragrafo 5.3) attraverso la quale viene inquadrata la cella di lavoro, con lo scopo di monitorare l'ingresso di operatori umani. Dunque, il robot UR10 non collabora al task cooperativo ma ha solo il ruolo di supervisore, mentre la cooperazione avviene tra i robot Comau Smart Six con lo scopo di trasportare un oggetto difficilmente trasportabile da un robot soltanto. Lo spazio di lavoro dell'UR10 risulta essere alquanto limitato in quanto bisogna far attenzione alla presenza del robot Comau posto avanti ad esso e dei vari altri ostacoli presenti all'interno della cella.



Figura 5.3: Cella robotica del Laboratorio di Automatica dell'Università degli Studi di Salerno

5.1.1 Robot UR10

Il robot Comau Smart Six è già stato presentato in precedenza, quindi viene riportata solo una breve descrizione del robot UR10 (Figura 5.4). Esso è un robot industriale della famiglia Universal Robot, il più grande della gamma, progettato per svolgere attività faticose, ovvero il trasporto di oggetti pesanti, senza perdere in precisione e affidabilità. L'UR10, infatti, è pensato per processi di confezionamento, palletizzazione, assemblaggio, pick&place o comunque per tutte le fasi che richiedono una portata di 10Kg e uno sbraccio massimo di 1.3 metri [14].

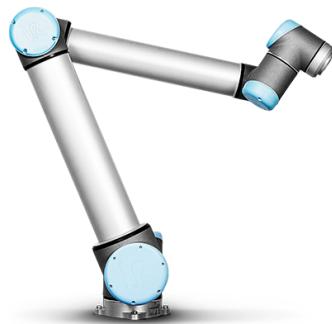


Figura 5.4: Robot UR10

I parametri di Denavit-Hartenberg (DH) relativi al robot in questione sono riportati nella tabella 5.1.

Braccio	a_i (metri)	d_i (metri)	α_i (gradi)	θ_i
1	0	0.128	-90	q_1
2	0.6127	0	0	q_2
3	0.5716	0	0	q_3
4	0	0.1639	-90	q_4
5	0	0.1157	90	q_5
6	0	0.0922	0	q_6

Tabella 5.1: Parametri di Denavit-Hartenberg relativi al robot UR10

5.2 Robot Operating System

Robot Operating System (ROS) è un framework per lo sviluppo e la programmazione di robot. ROS fornisce i servizi standard di un sistema operativo, come: astrazione dell'hardware, controllo dei dispositivi tramite driver, comunicazione tra processi, gestione delle applicazioni e altre funzioni di uso comune. Nonostante l'importanza della reattività e della bassa latenza nelle operazioni di controllo di un robot, ROS non è un sistema operativo real-time. Supporta la scrittura di software in due linguaggi di programmazione: Python e C++; quest'ultimo è stato utilizzato per l'implementazione del lavoro che verrà descritto [9].

5.3 Kinect

La Kinect è una telecamera sviluppata da Microsoft, esistente in due versioni. La prima versione, ovvero quella utilizzata in questo lavoro, è dotata di una telecamera RGB e di una telecamera a radiazione infrarossa con un risoluzione di 480p ciascuna, ed una capacità di lettura del movimento a 30 fps. Per il calcolo della lontananza degli oggetti la Kinect si avvale di uno scanner 3D a luce strutturata, in combinazione con gli infrarossi, che è dotato di una risoluzione di 320×240 pixel.

5.3.1 Calibrazione kinect

Per identificare la matrice di roto-traslazione tra le kinect e il giunto 6 (Figura 5.7) del robot UR10 e dunque identificare dove è posizionata e come è orientata la terna relativa ad essa, in riferimento alla terna mondo, è stato necessario dover eseguire la calibrazione di essa. La calibrazione della camera è quel processo volto a stimare parametri necessari per correlare i punti del mondo reale (sistema di coordinate mondo) con i punti dell'immagine catturata tramite una telecamera. I parametri si dividono in due categorie: intrinseci (dipendono dalla telecamera, come la lunghezza focale) ed estrinseci che sono quelli a cui siamo interessati perché consentono il cambio di coordinate tra due sistemi di riferimento. Sono stati sviluppati diversi metodi di calibrazione la maggior parte dei quali utilizzano un oggetto (target di calibrazione) sul quale sono tracciati i punti di calibrazione costituiti da N elementi che nell'immagine devono essere riconoscibili senza ambiguità ed avere coordinate note con accuratezza. Gli N elementi in genere sono quadrati disposti a scacchiera o dischi circolari, solitamente di colore nero su sfondo bianco. Per questo lavoro di tesi è stata utilizzata una scacchiera e sono stati acquisiti frame inquadrando da diverse distanze e diverse angolazioni. Si è deciso di utilizzare un algoritmo di calibrazione già sviluppato e fornito dall'ambiente di calcolo Matlab attraverso l'App *Camera Calibrator* la quale prende in input un certo numero di immagini della scacchiera acquisite attraverso la camera da calibrare e dà in output i parametri intrinseci ed estrinseci della camera. I frame

acquisiti sono mostrati in Figura 5.5.

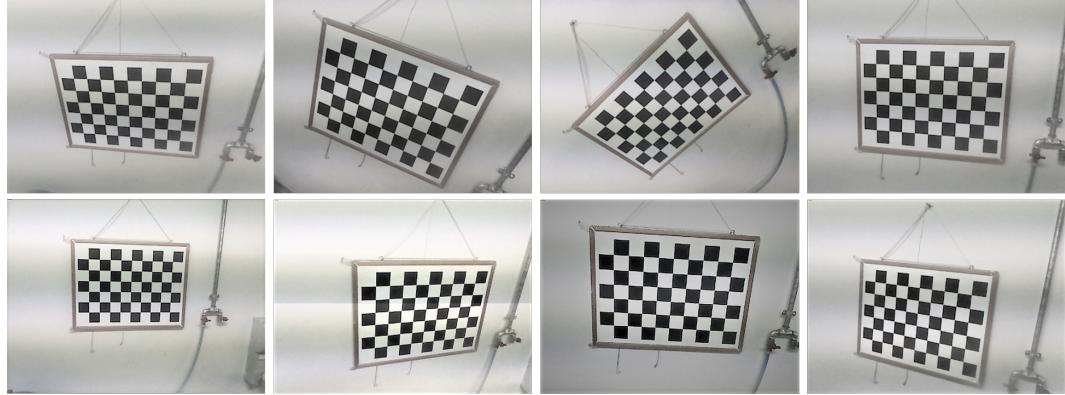


Figura 5.5: Frame acquisiti tramite kinect montata sul robot UR10, per effettuarne la calibrazione

Tra i parametri estrinseci dati in output dall'App di calibrazione troviamo le posizioni della scacchiera rispetto alla camera in riferimento all' i -esimo frame, le quali vengono denotate con $\mathbf{p}_{s,i} \in \mathbf{R}^3$. Inoltre sono fornite le matrici $\mathbf{R}_{c,i}^s \in \mathbf{R}^{3 \times 3}$ che esprimono la rotazione della camera rispetto alla scacchiera. Costruiamo a questo punto le matrici di roto-traslazione $\mathbf{A}_{c,i}^s \in \mathbf{R}^{4 \times 4}$ come:

$$\mathbf{A}_{c,i}^s = \begin{bmatrix} \mathbf{R}_{c,i}^{s \ T} & \mathbf{p}_{s,i} \\ \mathbf{0}_3^T & 1 \end{bmatrix} \quad (5.1)$$

dove $\mathbf{R}_{c,i}^{s \ T}$ esprime la rotazione della scacchiera rispetto alla camera.

Per ciascun frame acquisito sono state salvate le relative posizioni ai giunti del robot UR10. Tali configurazioni sono servite a calcolare la matrice di rotazione della terna associata all'end-effector dell'UR10 rispetto alla terna associata alla base del robot $\mathbf{R}_{ee,i}^0 \in \mathbf{R}^{3 \times 3}$ e le posizioni dell'end-effector spazio operativo rispetto alla terna di riferimento associata alla base del robot $\mathbf{p}_{ee,i} \in \mathbf{R}^3$.

A questo punto si può scrivere la seguente relazione:

$$\mathbf{A}_0^s \mathbf{A}_{ee,i}^0 \mathbf{A}_c^{ee} = \mathbf{A}_{c,i}^s \quad (5.2)$$

la quale può essere riscritta come:

$$\mathbf{A}_{ee,i}^0 \mathbf{A}_c^{ee} = \mathbf{A}_s^0 \mathbf{A}_{c,i}^s \quad (5.3)$$

È possibile ora applicare la risoluzione dell'equazione di Sylvester. Ovvero data un'equazione di Sylvester, cioè un'equazione della forma $\mathbf{AX} + \mathbf{XB} = \mathbf{C}$ dove, $\mathbf{A}, \mathbf{B}, \mathbf{C}$ sono matrici di dimensione $n \times n$, l'equazione può essere riscritta utilizzando il prodotto di Kronecker e l'operatore di vettorializzazione vec, come:

$$(\mathbf{I}_n \otimes \mathbf{A} + \mathbf{B}^T \otimes \mathbf{I}_n) \operatorname{vec}(\mathbf{X}) = \operatorname{vec}(\mathbf{C}) \quad (5.4)$$

Considerando come incognite \mathbf{A}_c^{ee} e \mathbf{A}_s^0 nel nostro caso scriviamo:

$$\mathbf{A}_{ee,i}^0 \mathbf{A}_c^{ee} = \mathbf{A}_s^0 \mathbf{A}_{c,i}^s \quad (5.5)$$

che può essere riscritta come:

$$(\mathbf{I}_4 \otimes \mathbf{A}_{ee,i}^0) * \operatorname{vec}(\mathbf{A}_c^{ee}) - (\mathbf{A}_{c,i}^{s,T} \otimes \mathbf{I}_4) * \operatorname{vec}(\mathbf{A}_s^0) = 0 \quad (5.6)$$

dove $\operatorname{vec}(\mathbf{A}_p^l) = [\mathbf{x}_{lp} \ \mathbf{y}_{lp} \ \mathbf{z}_{lp} \ \mathbf{O}_{lp} \ 1]^T \in \mathbf{R}^{16}$ ricordando che una generica matrice di roto-traslazione è costruita come

$$\mathbf{A}_p^l = \begin{bmatrix} \mathbf{x}_{lp} & \mathbf{y}_{lp} & \mathbf{z}_{lp} & \mathbf{O}_{lp} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.7)$$

Poniamo:

$$\Phi_i = \begin{bmatrix} \mathbf{I}_4 \otimes \mathbf{A}_{ee,i}^0 \\ -\mathbf{A}_{c,i}^{s,T} \otimes \mathbf{I}_4 \end{bmatrix}, \quad \zeta = \begin{bmatrix} \operatorname{vec}(\mathbf{A}_c^{ee}) \\ \operatorname{vec}(\mathbf{A}_s^0) \end{bmatrix} \quad (5.8)$$

L'equazione 5.6 può essere riscritta tenendo conto di tutti i frame:

$$\Phi \zeta = 0 \quad (5.9)$$

con $\Phi = [\Phi_1 \ \dots \ \Phi_8] \in \mathbf{R}^{128 \times 32}$. Affinchè questo prodotto si annulli è necessario che $\zeta \in \mathcal{N}(\Phi)$. Se inoltre il $\mathcal{N}(\Phi)$ ha dimensione unitaria e dunque la base

è formata da un solo autovettore, sarà semplice identificare ζ .

È importante valutare la bontà della calibrazione effettuata in quanto è possibile ottenere risultati differenti a seconda dei frame utilizzati.

Per prima cosa si decompone Φ in valori singolari, ovvero nelle matrici $\mathbf{U} \in \mathbf{R}^{128 \times 128}$, $\mathbf{S} \in \mathbf{R}^{128 \times 32}$ e $\mathbf{V} \in \mathbf{R}^{32 \times 32}$ tali che $\Phi = \mathbf{U}\mathbf{S}\mathbf{V}^T$ e si verifica che il più piccolo autovalore λ_1 di \mathbf{S} sia prossimo a 0 e molto minore del penultimo autovalore λ_2 , supponendo di aver ordinato gli autovalori nel seguente modo $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_{32}$. Se questo è verificato, si pone l'ultimo autovalore λ_2 pari a 0 e si ricalcola $\Phi = \mathbf{U}\mathbf{S}\mathbf{V}^T$, in modo che $\dim(\mathcal{N}(\Phi)) = 1$ ed è quindi possibile trovare un solo autovettore $\mathbf{x} \in \mathbf{R}^{32}$ costituente la base del nullo e scegliere ζ lungo tale autovettore, ovvero $\zeta : \zeta \in \mathcal{N}(\Phi)$. I risultati migliori ottenuti utilizzando i set acquisiti in questo lavoro di tesi presentano $\lambda_1 = 0.0109$ e $\lambda_2 = 0.1242$. ζ contiene le matrici cercate \mathbf{A}_c^{ee} e \mathbf{A}_s^0 espresse come un unico vettore colonna. Dalla conoscenza della forma delle matrici di roto-traslazione infatti l'elemento (4, 4) deve essere pari ad 1. Poniamo quindi $\zeta = \frac{\mathbf{x}}{\|\mathbf{x}\|}$ per avere il suo ultimo elemento pari ad 1 e in seguito è possibile ricostruire le matrici \mathbf{A}_c^{ee} e \mathbf{A}_s^0 a partire dal vettore ζ trovato. Bisogna porre attenzione al fatto che le matrici di rotazione contenute all'interno delle matrici di roto-traslazione trovate vanno normalizzate in quanto il loro determinante deve essere pari ad 1. La matrice di roto-traslazione tra la terna end-effector e la terna camera trovata è:

$$\mathbf{A}_c^{ee} = \begin{bmatrix} -0.9974 & -0.0375 & 0.0612 & -0.0069 \\ 0.0341 & -0.9979 & -0.0551 & 0.0560 \\ 0.0632 & -0.0528 & 0.9966 & 0.0932 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.10)$$

La terna camera che è stata identificata è visualizzabile nella Figura 5.6 in cui la terna $O_b - x_b y_b z_b$ rappresenta l'orientamento della terna mondo (le origini non sono coincidenti).

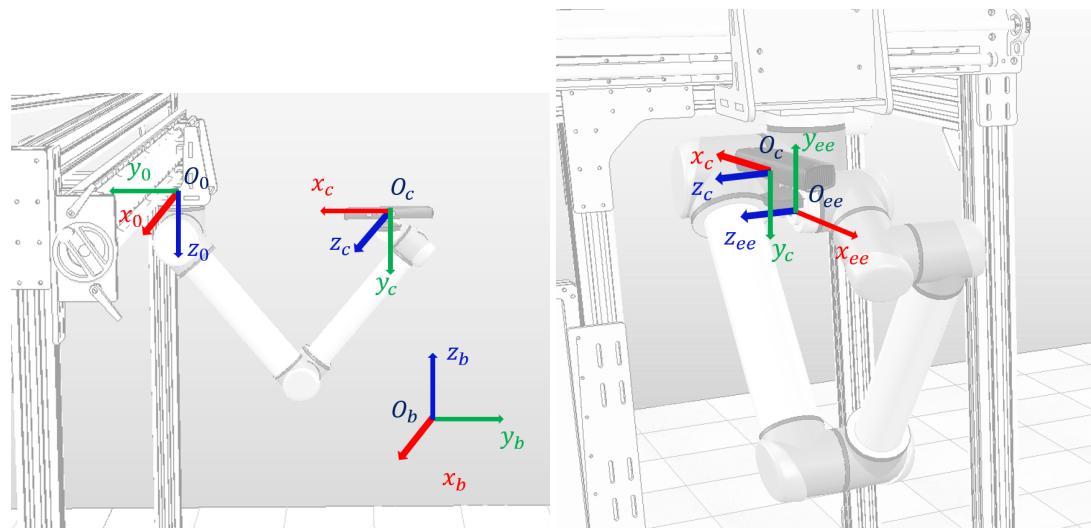


Figura 5.6: Terna associata alla kinect identificata attraverso il procedimento appena descritto

5.3.2 Detection e Tracking dello skeleton

Il software utilizzato per riconoscere una persona e farne il tracking è un software già sviluppato per Kinect e ROS e prende il nome di Openni_Tracker. Esso fornisce le posizioni, espresse in terna camera, di varie parti del corpo di una persona, come la testa, collo, spalle, torso, ecc. Queste informazioni vengono pubblicate sul topic "/tf". La terna in cui vengono lette le coordinate è mostrata in figura 5.7, la quale risulta essere ruotata rispetto alla terna della camera identificata, per cui è stato necessario individuare la matrice di rotazione tra le due che risulta essere:

$$\mathbf{R}_k^c = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix} \quad (5.11)$$

Con la lettera k ci si riferisce alla terna con cui il software Openni_Tracker restituisce le coordinate.

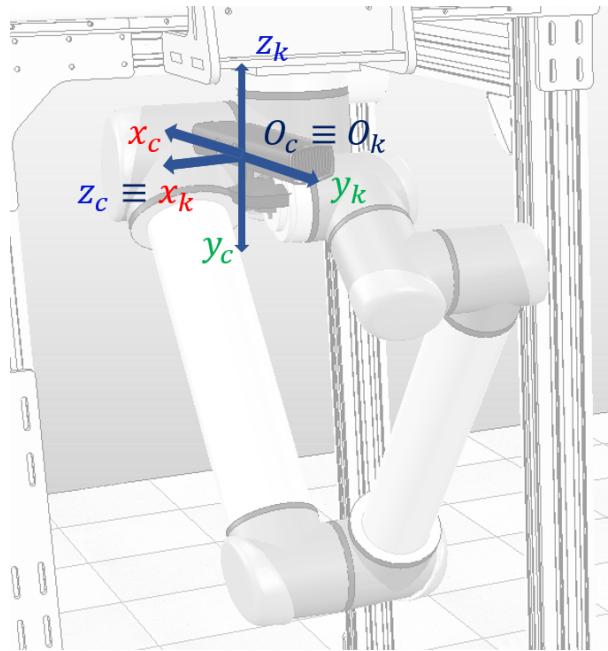


Figura 5.7: Terna c della camera, identificata attraverso il procedimento illustrato nel paragrafo precedente e terna con cui il software Openni_Tracker restituisce le coordinate

5.4 Architettura del sistema

L’architettura del sistema complessivo (Figura 5.8) è organizzata nel seguente modo: sono presenti due macro blocchi, uno riguardante i robot Comau Smart Six e l’altro riguardante il robot UR10 e la camera.

In particolare il blocco relativo ai Comau comprende un PC su cui gira il sistema operativo Ubuntu e su cui è presente un framework chiamato Orocó il quale fornisce l’infrastruttura e le funzionalità per la realizzazione di applicazioni real-time. Inoltre questo macro blocco comprende anche i Controller C4G dei due robot Comau i quali si interfacciano con il computer tramite un’infrastruttura di rete resa disponibile dal laboratorio di Automatica. Ciascun controller comunica al computer lo stato dei giunti del robot a cui è collegato, mentre il computer risponderà ad essi con i riferimenti in corrente da dare a ciascun giunto dei robot.

Il secondo macro blocco comprende un PC sul quale è installato il sistema operativo Ubuntu 14.04 e framework ROS e che comunica con il controller dell’UR10, appunto il controller stesso e la kinect. La kinect ha il compito di individuare

una persona analizzando i frame raccolti in real-time e di ricavare la posizione della persona rispetto ad una terna di riferimento associata ad essa. Per fare ciò, sul computer che gestisce tutta la logica di funzionamento del robot UR10 viene lanciato un nodo ROS che resterà sempre in esecuzione e che ha il compito di eseguire il tracking di una persona. La posizione di quest'ultima viene comunicata al nodo ROS che si occupa di gestire la logica di movimentazione del robot UR10, sempre in esecuzione sul medesimo computer, tramite messaggio pubblicato su topic. Il nodo ROS principale, leggerà la posizione della persona da topic e la utilizzerà per opportuni calcoli. Dopo aver fatto ciò, il nodo principale dovrà comunicare al controller del robot i riferimenti desiderati, ovvero posizioni e velocità ai giunti. Il controller invece risponderà al nodo principale comunicando le posizioni e le velocità effettive del robot, ovvero lo stato reale di esso. La comunicazione tra controller e computer avviene tramite un collegamento fisico punto punto attraverso una rete privata (scelta fatta per garantire che i pacchetti non subiscano ritardo avendo esigenze di real-time, infatti il robot UR10 si aspetta un riferimento ogni 8 millisecondi).

Infine tra i due macro blocchi esiste un collegamento fisico punto punto per scambiare informazioni e rendere le due applicazioni interattive. Infatti il computer con Orocosp comicherà al computer con ROS ad ogni passo di campionamento dei robot Comau (2 millisecondi) la posizione dell'oggetto che i robot stanno trasportando durante l'esecuzione del proprio task, mentre la comunicazione inversa serve per generare dei delta in posizione, velocità e accelerazione relativi al task eseguito dai robot Comau. I delta vengono calcolati sulla base della vicinanza della persona all'oggetto trasportato.

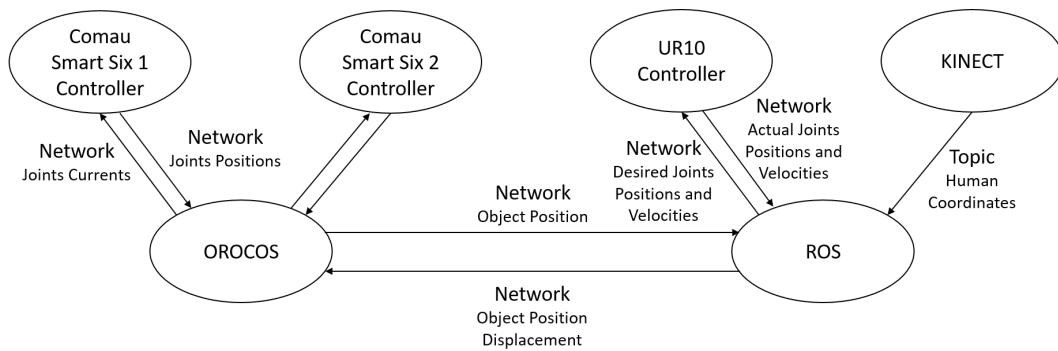


Figura 5.8: Architettura del sistema utilizzato nella fase sperimentale

Per quanto riguarda i dettagli architetturali e funzionali del blocco riguardante i robot Comau Smart Six si può fare riferimento alla tesi [7].

L'architettura invece relativa alla comunicazione tra il controller del robot UR10 e il nodo ROS principale è mostrata in Figura 5.9.

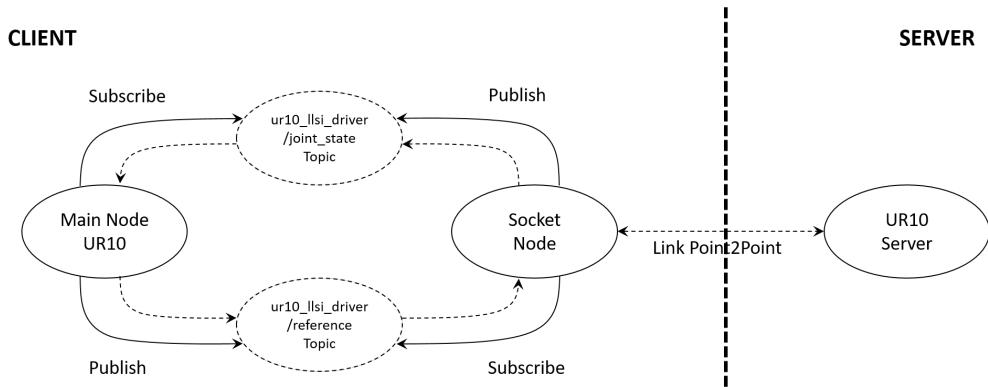


Figura 5.9: Architettura relativa alla comunicazione tra il controller del robot UR10 e il nodo ROS principale

Sono presenti 3 nodi principali, "Main Node UR10" che gestisce tutta la logica di funzionamento del robot e si trova sul PC con sistema operativo Ubuntu dotato di framework ROS citato in precedenza, "Socket Node" che è un nodo attivo sullo stesso PC del "Main Node UR10" e si occupa di far dialogare il nodo appena citato e il nodo Server dell'UR10, ovvero il controller dell'UR10, attraverso una socket TCP, infine è presente il nodo "UR10 Server" che si trova su un elaboratore separato connesso al PC tramite connessione di rete fisica punto-punto su cui sono installate le librerie di base del robot fornite dall'azienda costruttrice e su cui è

in esecuzione un programma, il server, che ha il ruolo di gestire la comunicazione col nodo Socket e di controllare il corretto funzionamento del robot.

5.5 Pianificazione del moto del robot UR10

Di seguito è spiegata la logica di funzionamento del software che gestisce la movimentazione del robot UR10. Non appena il programma viene avviato, la prima cosa che viene eseguita è una traiettoria nello spazio giunti che ha lo scopo di portare il robot in una configurazione di partenza consigliata per quelle che saranno le attività successive. Dunque campione per campione la traiettoria viene pianificata e le posizioni ai giunti restituite dalla pianificazione vengono date come riferimenti al robot.

Finita la traiettoria spazio giunti distinguiamo due modalità diverse di funzionamento, una in cui il robot ricerca una persona e alternando fasi di detection con fasi durante le quali esso si sposta in una posizione diversa per riuscire ad inquadrare tutte le aree della cella (parte evidenziata in verde nella Figura 5.11), e una modalità che prevede il tracking di una persona non appena questa viene individuata (parte evidenziata in blu).

- Prima modalità.

Il robot alterna due fasi. In una viene mantenuto fermo e viene fatta la detection della human. In particolare si esegue la lettura dal topic /tf per verificare se sono state pubblicate informazioni dal software Openni_Tracker in esecuzione su un altro nodo ROS. Questa verifica viene fatta per un determinato lasso di tempo, nel caso specifico degli esperimenti 10 secondi, dopodiché il robot viene spostato in un'altra posizione per poter inquadrare un'altra zona della cella di lavoro e per ricominciare la ricerca. Si passa dunque all'altra fase in cui avviene la pianificazione di una traiettoria nello spazio operativo, l'inversione cinematica delle variabili di spazio operativo restituite dalla pianificazione e infine la scrittura su topic dei riferimenti ai giunti del robot restituiti dall'inversione cinematica.

- Seconda modalità.

La seconda modalità entra in gioco non appena la persona viene identificato all'interno della cella e viene subito eseguita la lettura da topic /tf della sua posizione. A partire da questa vengono calcolati la posizione e l'orientamento desiderati per il robot per il prossimo campione. Siccome, la kinect riesce a rispondere in un tempo di campionamento superiore a quello a cui lavora il robot UR10, è necessario gestire la situazione in cui è attivo il tracking della persona, ma sul topic non sono presenti informazioni riguardo la sua posizione. In tal caso infatti la lettura da topic lancerà un'eccezione e come posizione e orientamento desiderati per il robot, vengono utilizzati quelli del precedente passo di campionamento del robot. Però, nel caso in cui non vengono trovate informazioni su topic si possono creare ambiguità. Infatti questo può avvenire sia per un ritardo di risposta del software che gira a bordo della kinect, sia perché la persona è stata persa e quindi non viene più pubblicato nulla riguardo essa. Dunque si utilizza un timer per tener conto di ciò e se per un tempo superiore alla durata del timer il software che stiamo descrivendo non trova informazioni, allora si effettua prima una traiettoria spazio giunti per riportarsi nella posizione in cui ci si porta quando il software viene avviato e in seguito si rientra nella prima modalità di funzionamento del robot partendo con la ricerca di una nuova persona.

La posizione e l'orientamento desiderati calcolati ad ogni passo di campionamento a partire dalla posizione della persona nella cella, vengono opportunamente filtrati, in modo da evitare che in input all'inversione cinematica vadano dei gradini e non una traiettoria dolce. Il filtro utilizzato è un filtro del secondo ordine che ha diversa velocità per le posizioni e per gli orientamenti (Figura 5.10). Per quello in posizione sono stati scelti $\zeta = 1$ e $\omega_n = 0.1$ e per quello in orientamento si è scelto di utilizzare $\zeta = 1$ e $\omega_n = 0.5$.

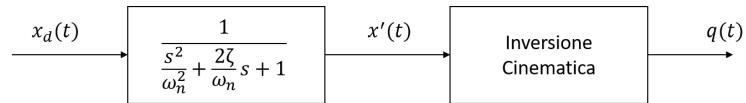


Figura 5.10: Filtro del secondo ordine per posizione e orientamento

Dato l'utilizzo del filtro, quando una persona viene persa, non è detto che il robot abbia già raggiunto l'ultima posizione desiderata o orientamento desiderato e quindi si hanno ancora velocità nello spazio operativo diverse da zero.

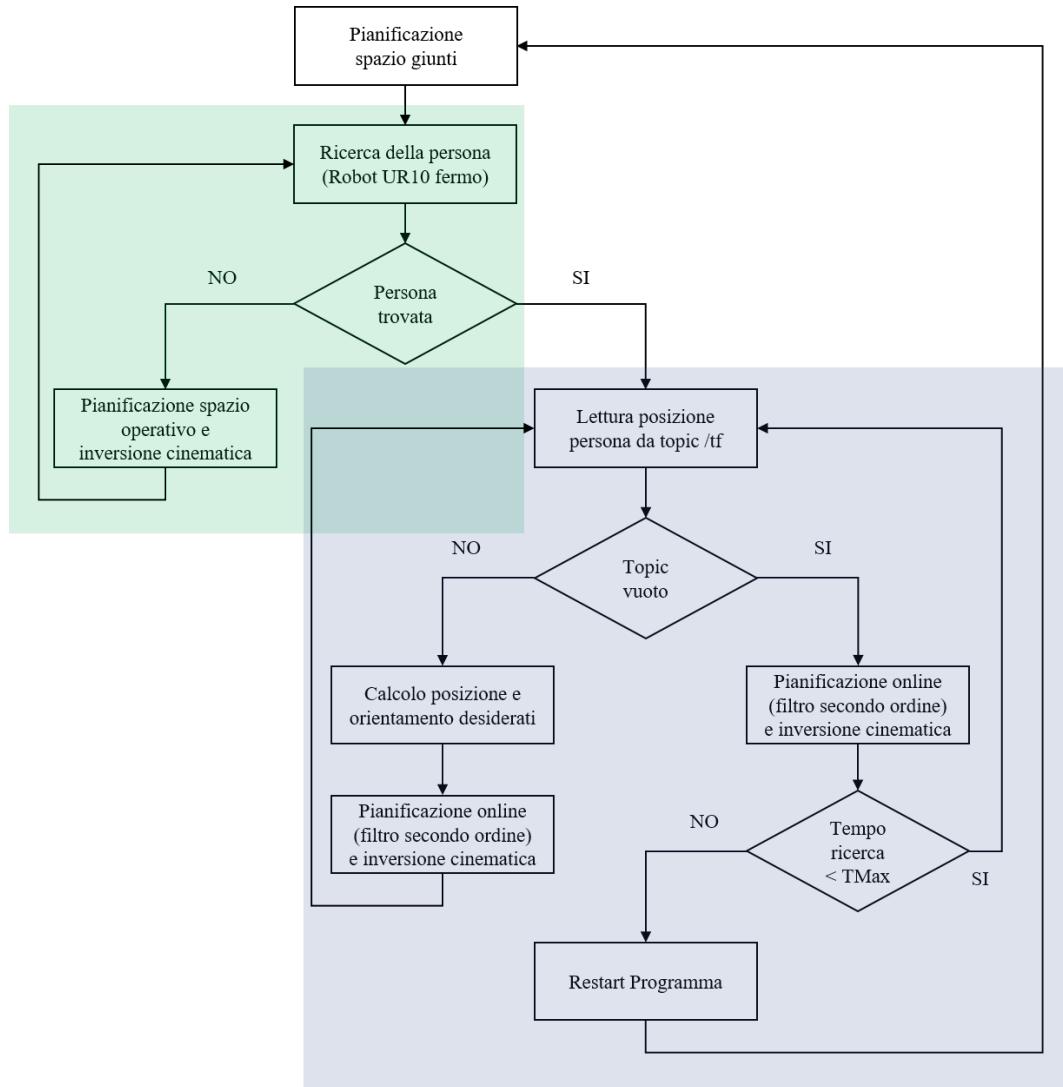


Figura 5.11: Diagramma di flusso del software che gestisce la movimentazione del robot UR10

Oltre alla logica di movimentazione del robot UR10, il software gestisce anche la logica di modifica del task di cooperazione eseguito dai robot Comau Smart-Six (Figura 5.12). In particolare il software legge da topic le informazioni target sull'oggetto trasportato dai robot Comau, ovvero la posizione, la velocità e l'accelerazione. La modifica del task consiste nel calcolo di un delta in posizione, velocità e accelerazione da sommare alle posizioni, velocità e accelerazioni desiderate per l'oggetto.

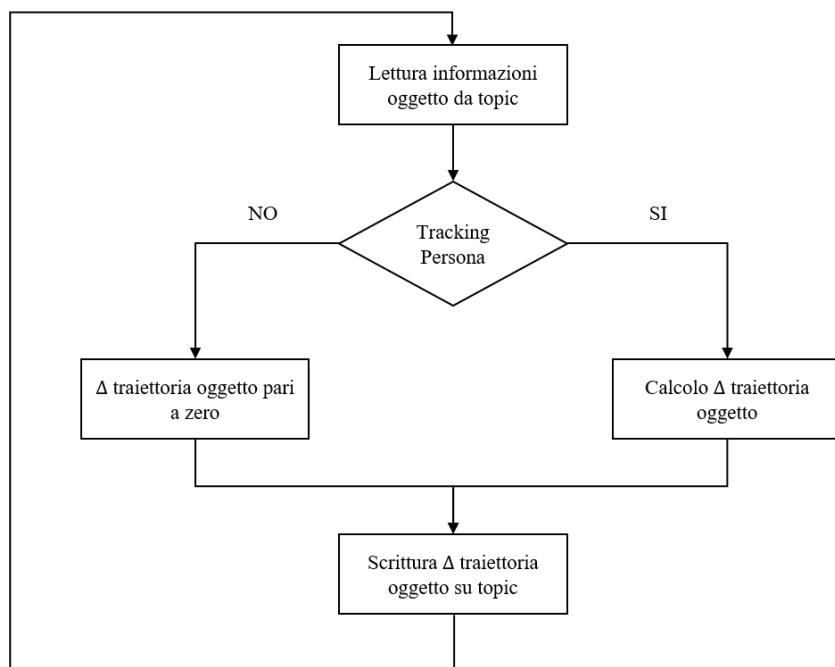


Figura 5.12: Diagramma di flusso del software che gestisce il calcolo del delta sul task eseguito dai robot Comau

Nel caso in cui ci troviamo nella prima modalità di funzionamento del robot UR10, spiegata precedentemente, il task eseguito dai robot Comau non viene modificato, in quanto questa modalità implica che nessuna persona ha invaso lo spazio del task. Nel caso in cui invece una persona è stata identificata, viene calcolata la distanza di questa dall'oggetto, per generare i delta sul task. La legge utilizzata corrisponde ad un controllo di impedenza ed è data da:

$$\Delta \ddot{\sigma} = f - K_D \Delta \dot{\sigma} - K_P \Delta \sigma \quad (5.12)$$

In questo modo il comportamento desiderato, ovvero il delta sul task generato quando la persona si avvicina all'oggetto, è specificato mediante un'impedenza dinamica, ossia un sistema massa-molla-smorzatore. La scelta della rigidezza nel modello regola l'ampiezza dei delta generati. Ovviamente si desidera che l'oggetto sia “cedevole”, ovvero si allontani, rispetto all'avvicinarsi della persona, quindi il \mathbf{K}_P viene scelto molto piccolo. Mentre, come appena detto, \mathbf{K}_P nel sistema rappresenta la rigidezza, \mathbf{K}_D lo smorzamento ed \mathbf{f} la massa. \mathbf{K}_D è scelto come $2\sqrt{\mathbf{K}_P}$. La norma di \mathbf{f} è una funzione della distanza tra la persona e l'oggetto trasportato e ha la seguente forma (Figura 5.13):

$$\|\mathbf{f}\| = \begin{cases} 0.887 K_P & 0 \leq d \leq 0.6 \\ \frac{-1.2 (d - 2.3) K_P}{2.3} & 0.6 \leq d \leq 2.3 \\ 0 & d \geq 2.3 \end{cases} \quad (5.13)$$

La direzione di \mathbf{f} è data invece dal seguente versore:

$$vers(\mathbf{f}) = \frac{\mathbf{o} - \mathbf{h}}{\|\mathbf{o} - \mathbf{h}\|} \quad (5.14)$$

dove \mathbf{o} indica la posizione dell'oggetto trasportato dai robot COMAU.

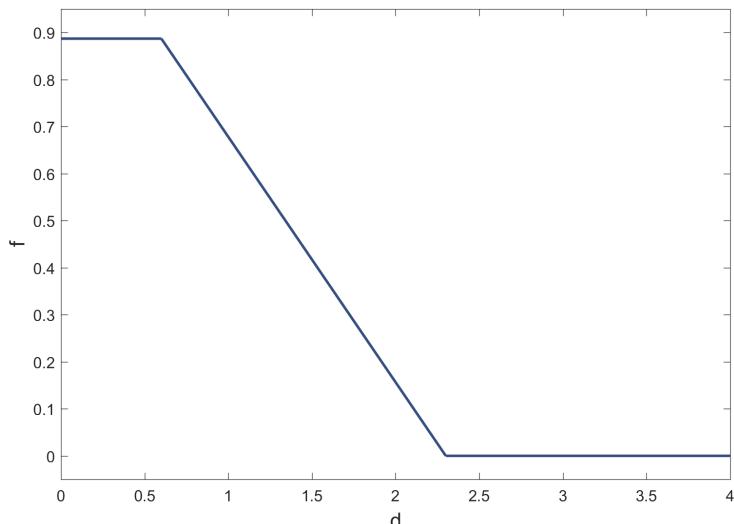


Figura 5.13: Forza generata per la legge di impedenza

5.5.1 Orientamento desiderato

Non appena inizia il tracking di una persona si desidera che quest'ultima sia inquadrata dalla camera sempre a centro immagine in modo da facilitarne il tracking e da non perderla mai mentre questa si muove all'interno della cella. Bisogna quindi calcolare per ogni passo di campionamento un orientamento desiderato per l'end-effector del robot affinché esso venga ruotato in modo da mantenere la camera parallela al piano $\pi_{x_b y_b}$ della terna base e in modo da mantenere la persona a centro immagine. La parte del corpo della persona considerata è la testa, dunque le coordinate sono espresse considerando il frame riferito ad essa

L'algoritmo prende in input la posizione della persona espressa in terna camera $\mathbf{h} = (h_x, h_y, h_z)$ e la matrice di rotazione \mathbf{R}_c^b tra la terna associata alla kinect e la terna mondo.

Analizziamo prima di tutto la rotazione necessaria intorno all'asse y della terna camera la quale serve ad allineare l'asse uscente dalla lente della camera e il vettore posizione relativo alla persona espresso in terna camera. Facendo riferimento alla Figura 5.14 l'angolo di rotazione intorno all'asse y della terna camera viene calcolato considerando la proiezione \mathbf{h}_{xz} di \mathbf{h} sul piano $\pi_{x_c z_c}$ e calcolando:

$$\varphi = \arcsin \frac{\mathbf{h}_{xz} \cdot \mathbf{x}_c}{\|\mathbf{h}_{xz}\| \|\mathbf{x}_c\|} \quad (5.15)$$

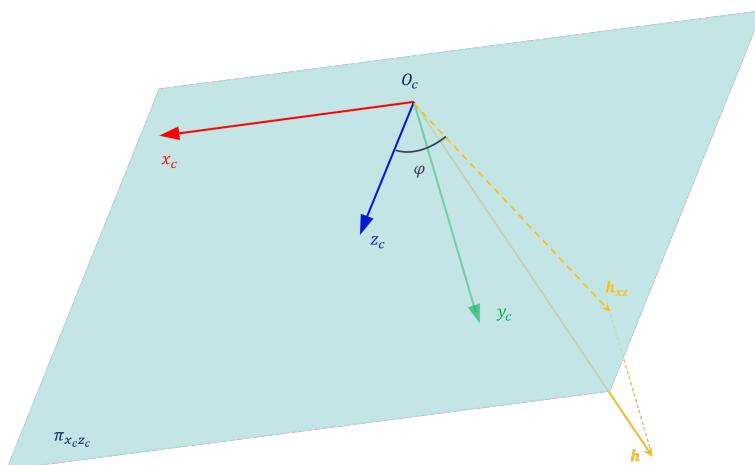


Figura 5.14: Calcolo dell'angolo di rotazione intorno all'asse y

A questo punto è possibile calcolare la matrice di rotazione per passare dalla terna camera corrente alla nuova terna effettuando una rotazione di un angolo φ intorno all'asse \mathbf{y}_c .

$$\mathbf{R}_y(\varphi) = \begin{bmatrix} \cos(\varphi) & 0 & \sin(\varphi) \\ 0 & 1 & 0 \\ -\sin(\varphi) & 0 & \cos(\varphi) \end{bmatrix} \quad (5.16)$$

Si può esprimere così la matrice di rotazione desiderata per passare dalla terna mondo alla terna camera desiderata come:

$$\mathbf{R}_{c,des}^b = \mathbf{R}_c^b \mathbf{R}_y(\varphi) \quad (5.17)$$

dove \mathbf{R}_c^b è data dalla cinematica diretta. La terna che si ottiene dopo la rotazione è mostrata in Figura 5.15. Come si può notare l'asse x della terna camera in cui ci si vuole portare non appartiene al piano $\pi_{x_b y_b}$ per cui è necessario ruotarla in modo da ottenere l'appartenenza al piano per far sì che la camera non stia mai in obliquo.

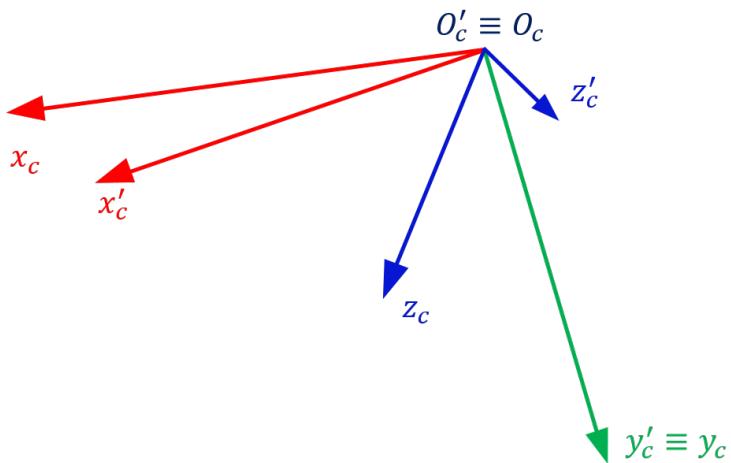


Figura 5.15: Terna ottenuta tramite rotazione dell'angolo φ intorno all'asse \mathbf{y}_c

Ricordiamo che le colonne di una qualsiasi matrice di rotazione costituiscono i versori della terna di arrivo, quindi a partire da $\mathbf{R}_{c,des}^b$ è possibile ricavare \mathbf{x}'_c , \mathbf{y}'_c e \mathbf{z}'_c . Consideriamo inoltre la terna $\mathbf{O}'_b - \mathbf{x}'_b \mathbf{y}'_b \mathbf{z}'_b$ che ha versori con stessa direzione e verso della terna mondo ma origine coincidente con \mathbf{O}'_c .

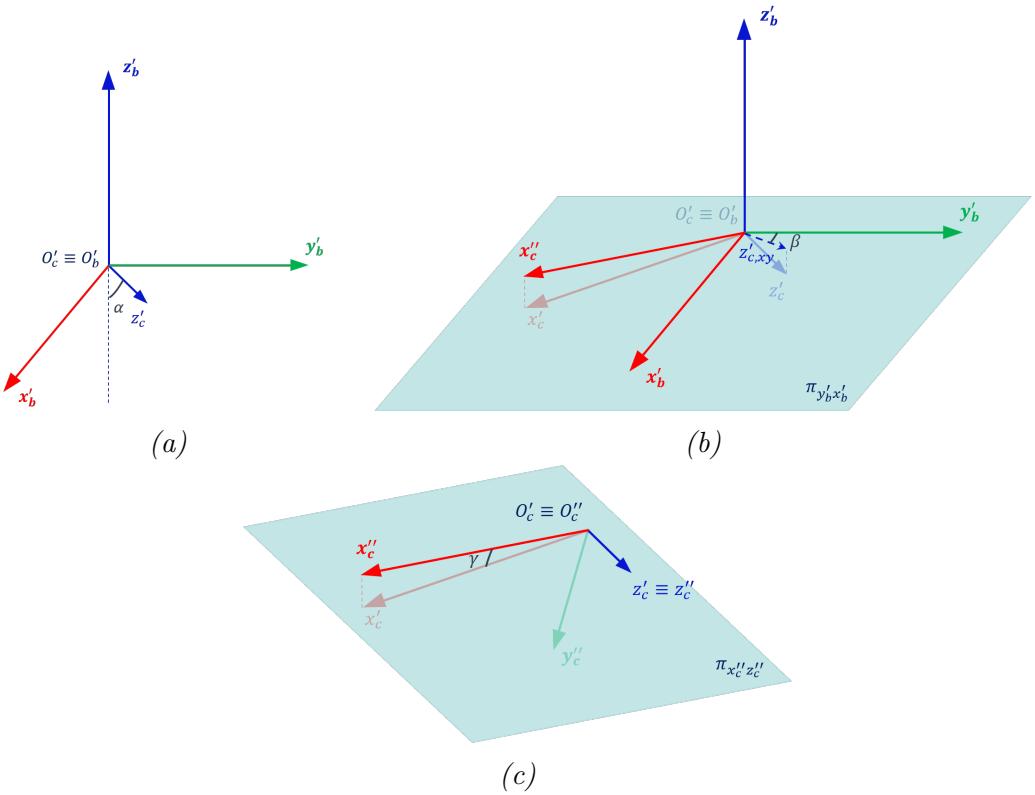


Figura 5.16: Rotazioni necessarie per allineare la kinect al piano $\pi_{x_b y_b}$

Calcoliamo ora l'angolo che si forma tra l'asse z'_c e l'asse z'_b (Figura 5.16a) come:

$$\alpha = \arccos \frac{\mathbf{z}'_b \cdot \mathbf{z}'_c}{\|\mathbf{z}'_b\| \|\mathbf{z}'_c\|} \quad (5.18)$$

Dalla conoscenza dell'angolo α è possibile calcolare la proiezione del vettore z'_c sul piano $\pi_{y'_b x'_b}$ come (Figura 5.16b):

$$\mathbf{z}'_{c,yx} = \frac{\mathbf{z}'_c}{\sin(\alpha)} \quad (5.19)$$

Calcoliamo ora l'angolo formato dal vettore $\mathbf{z}'_{c,yx}$ con l'asse y'_b in modo da poter ricavare l'asse x_c desiderato, ovvero appartenente al piano $\pi_{x_b y_b}$:

$$\beta = \arcsin \frac{\mathbf{z}'_{c,yx} \cdot \mathbf{x}'_b}{\|\mathbf{z}'_{c,yx}\| \|\mathbf{y}'_b\|} \quad (5.20)$$

Dunque si ha:

$$\mathbf{x}_c'' = \begin{bmatrix} \cos\left(\beta - \frac{\pi}{2}\right) \\ \sin\left(\beta - \frac{\pi}{2}\right) \\ 0 \end{bmatrix} \quad (5.21)$$

Resta da calcolare qual è l'angolo di rotazione intorno all'asse \mathbf{z}'_c necessario per portare il versore \mathbf{x}'_c sul \mathbf{x}''_c . Calcoliamo dunque la normale al piano $\pi_{\mathbf{x}''_c \mathbf{y}''_c}$ (Figura 5.16c) tramite il seguente prodotto vettoriale: $\mathbf{y}''_c = \mathbf{x}''_c \times \mathbf{z}''_c$ la quale deve essere normalizzata per rappresentare un versore:

$$\mathbf{y}''_c = \frac{\mathbf{y}''_c}{\|\mathbf{y}''_c\|} \quad (5.22)$$

Possiamo ora calcolare l'angolo γ di rotazione intorno all'asse \mathbf{z}'_c mostrato in Figura 5.16c come:

$$\gamma = \arcsin \frac{\mathbf{x}'_c \cdot \mathbf{y}''_c}{\|\mathbf{x}'_c\| \|\mathbf{y}''_c\|} \quad (5.23)$$

La rispettiva matrice di rotazione, per ottenere dunque una rotazione intorno all'asse \mathbf{x}'_c di un angolo γ è data da:

$$\mathbf{R}_{\mathbf{z}'_c}(\gamma) = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (5.24)$$

Infine bisogna effettuare una rotazione intorno all'asse x della terna corrente in modo da portare la testa della persona al centro dell'immagine. Calcoliamo l'angolo di cui bisogna ruotare considerando la terna di partenza e la posizione della persone espressa in quella terna. In particolare va calcolato il complemento dell'angolo che si forma tra la proiezione di \mathbf{h} sul piano $\pi_{\mathbf{y}_c \mathbf{z}_c}$, ovvero \mathbf{h}_{yz} e l'asse \mathbf{y}_c come mostrato in Figura 5.17:

$$\theta = \arcsin \frac{\mathbf{h}_{yz} \cdot \mathbf{y}_c}{\|\mathbf{h}_{yz}\| \|\mathbf{y}_c\|} \quad (5.25)$$

La rispettiva matrice di rotazione, per ottenere dunque una rotazione intorno

all'asse \mathbf{x}_c'' di un angolo θ è data da:

$$\mathbf{R}_{\mathbf{x}_c''}(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\gamma) & -\sin(\gamma) \\ 0 & \sin(\gamma) & \cos(\gamma) \end{bmatrix} \quad (5.26)$$

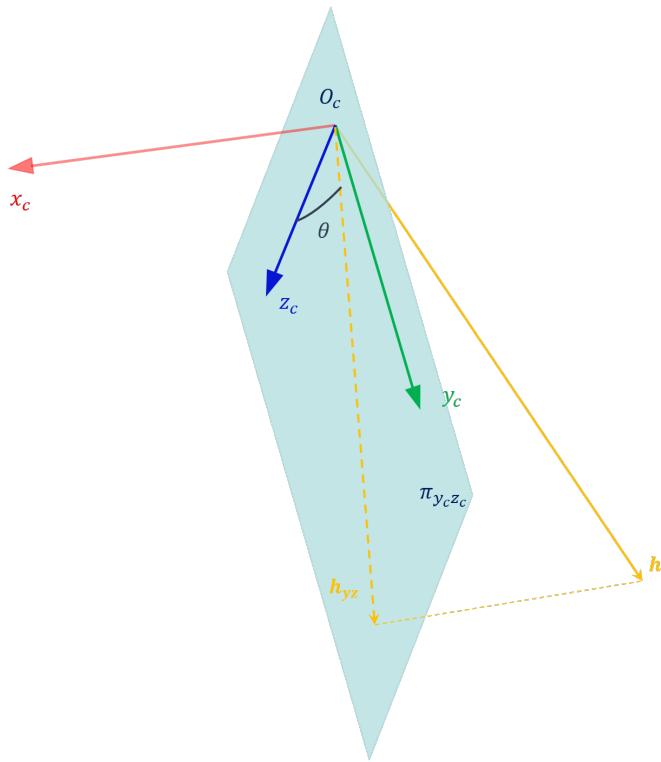


Figura 5.17: Calcolo dell'angolo di rotazione intorno all'asse x della nuova terna

La matrice di rotazione finale che esprime la rotazione per passare dalla terna mondo alla terna camera finale desiderata, illustrata in Figura 5.18, è data da:

$$\mathbf{R}_{c,finale}^b = \mathbf{R}_c^b \mathbf{R}_y(\varphi) \mathbf{R}_{z'_c}(\gamma) \mathbf{R}_{x''_c}(\theta) \quad (5.27)$$

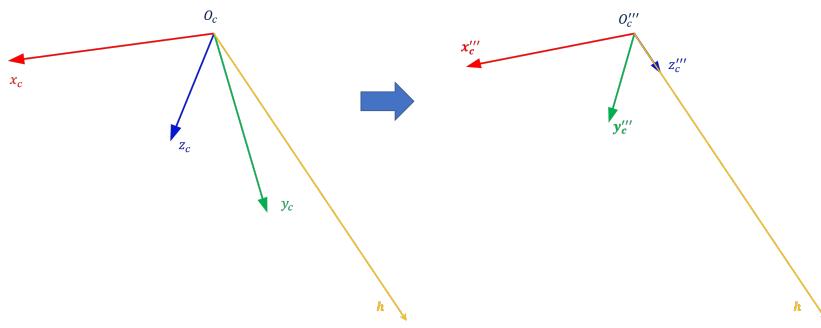


Figura 5.18: Terna camera corrente e terna camera desiderata, in modo che la kinect punti verso la persona mantenendolo al centro dell'immagine

5.5.2 Posizione desiderata

Nella fase di tracking della persona individuata all'interno della cella di lavoro viene calcolata per ogni passo di campionamento la posizione desiderata dell'end-effector del robot UR10 nello spazio operativo. In particolare lo scopo dell'algoritmo implementato è quello di mantenere sempre una certa distanza desiderata tra l'end-effector e la persona in modo da garantire sempre la sicurezza di quest'ultima e per poter inquadrare sempre in modo ottimo la zona della cella occupata da essa in modo da non perderne le tracce. Nel calcolare la nuova posizione desiderata per l'end-effector bisogna anche tener conto degli ostacoli presenti all'interno della cella i quali sono ben noti e del fatto che le posizioni calcolate appartengano allo spazio di lavoro del robot.

Gli input all'algoritmo per il calcolo della posizione desiderata dell'end-effector del robot UR10 per il prossimo passo di campionamento sono la posizione della persona \mathbf{h} espressa in terna mondo, la posizione della camera \mathbf{c} espressa anch'essa in terna mondo, \mathbf{b} la posizione della base del robot in terna mondo. Per comprendere l'algoritmo bisogna far riferimento alla Figura 5.19 e alla Figura 5.20. La camera nell'immagine rappresenta l'end-effector del robot.

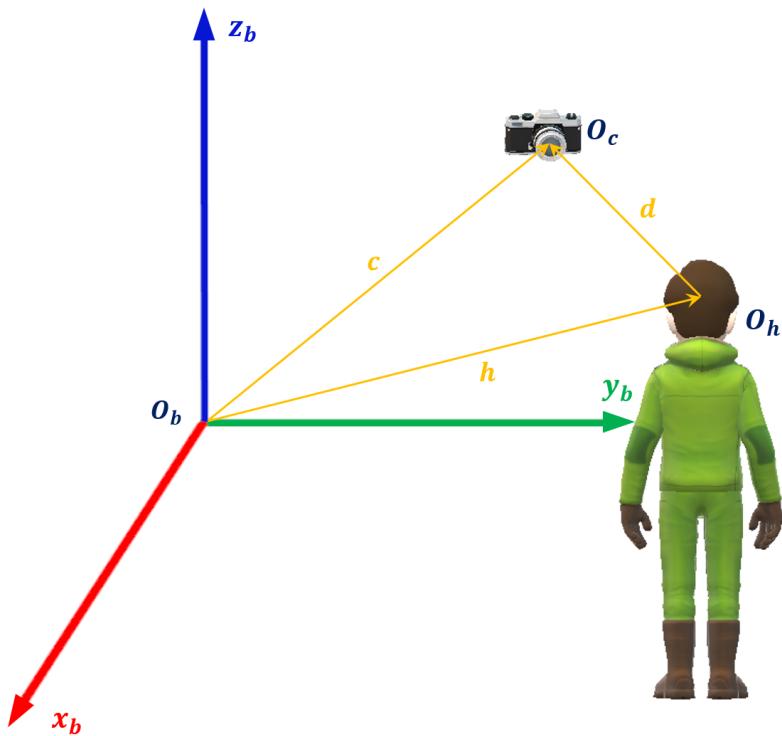


Figura 5.19: Scenario per la comprensione dell'algoritmo di calcolo della nuova posizione desiderata per l'end-effector del robot UR10

Vengono di seguito illustrati i passi dell'algoritmo.

1. Viene calcolata la distanza tra l'end-effector del robot UR10 e la persona da inseguire, come (Figura 5.19):

$$d = \|\mathbf{c} - \mathbf{h}\| \quad (5.28)$$

2. Se la distanza è maggiore o minore di quella desiderata allora bisogna scalare il vettore \mathbf{d} , ovvero bisogna calcolare un nuovo vettore che abbia stessa direzione del vettore \mathbf{d} ma norma diversa.

$$\mathbf{d}_{des} = (\mathbf{c} - \mathbf{h}) * \frac{d_{des}}{\|\mathbf{c} - \mathbf{h}\|} \quad (5.29)$$

e bisogna quindi calcolare la nuova posizione desiderata per l'end-effector:

$$\mathbf{c}_{des} = \mathbf{d}_{des} + \mathbf{h} \quad (5.30)$$

3. Si controlla che la posizione desiderata per l'end-effector per il prossimo campione \mathbf{c}_{des} sia contenuta nello spazio compreso tra la sfera di raggio inferiore e la sfera di raggio superiore della Figura 5.20 in cui \mathbf{b} è il vettore posizione della base del robot UR10 e \mathbf{c} è il vettore posizione desiderata per l'end-effector dell'UR10. Siccome le sfere sono entrambe centralate nell'origine del frame associato alla base del robot UR10 allora viene calcolato:

$$r = \|\mathbf{c}_{des} - \mathbf{b}\| \quad (5.31)$$

se il raggio è maggiore del raggio della sfera più grande si pone $r_{des} = r_{Max}$ se invece è minore del raggio della sfera più piccola si pone $r_{des} = r_{min}$. Se uno di questi due casi si verifica allora il vettore \mathbf{r} deve essere scalato ottenendo un vettore che ha stessa direzione di \mathbf{r} ma norma diversa:

$$\mathbf{r}_{des} = \mathbf{r} * \frac{r_{des}}{\|\mathbf{r}\|} \quad (5.32)$$

e di conseguenza la nuova posizione desiderata per l'end-effector diventa:

$$\mathbf{c}_{des} = \mathbf{r}_{des} + \mathbf{b} \quad (5.33)$$

4. La limitazione del punto precedente serve solo a non far uscire il robot UR10 dal proprio spazio di lavoro, ma non tiene conto degli ostacoli presenti all'interno della cella. Per evitare dunque collisioni le coordinate x, y e z costituenti la posizione desiderata dell'end-effector per il prossimo campione devono essere contenute all'interno di opportuni intervalli. Quindi si verifica l'appartenenza e in caso contrario la si impone.

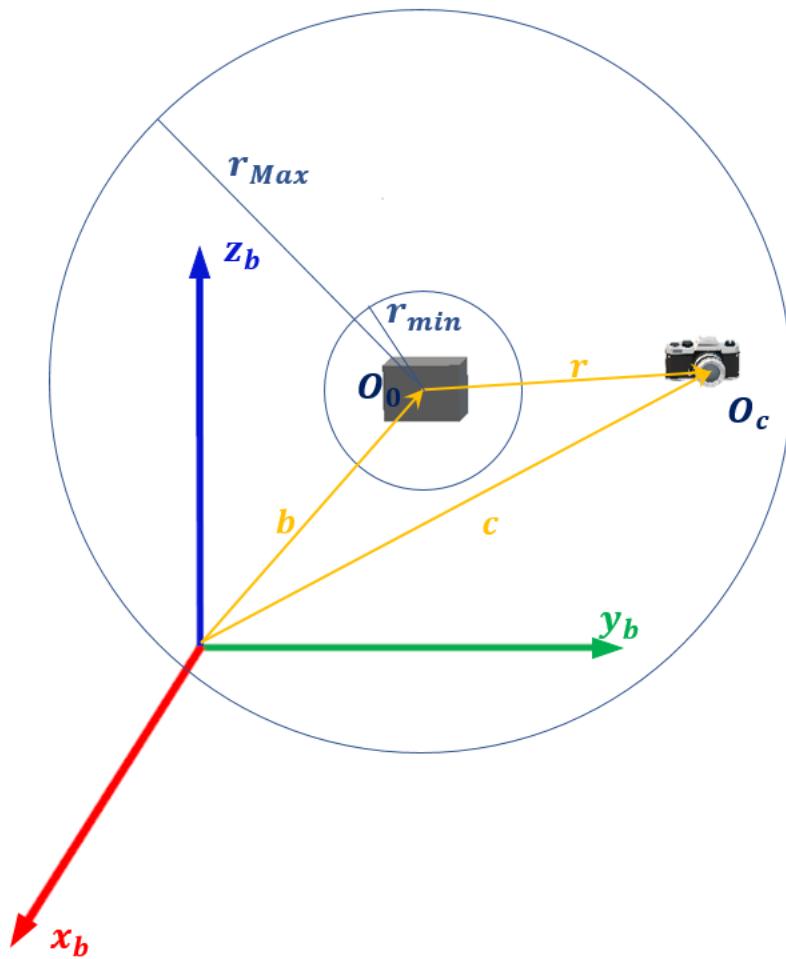


Figura 5.20: Spazio accessibile all'end-effector del robot UR10

5.6 Task cooperativo

I robot Comau Smart-Six assumono un funzionamento simile a quello presentato nel capitolo 4. In particolare l'architettura utilizzata è quella mostrata in Figura 3.1. Essi cooperano per il trasporto di un oggetto, in particolare uno scatolo (oggetto non particolarmente rigido e questo rende possibile evitare l'utilizzo del controllo in forza).

La comunicazione con il robot UR10 avviene con uno solo dei robot Comau, dunque i due robot Comau assumono i ruoli di leader e follower, dove è il leader a conoscere direttamente il delta sul task generato dall'UR10.

Sono stati effettuati due esperimenti, uno in cui il task è costante, dunque il ba-

ricentro non viene modificato per tutta la durata del task e si modifica solo a causa della generazione di un $\Delta\sigma$ diverso da zero e uno in cui l'oggetto non è solo afferrato e mantenuto fermo dai robot ma viene anche trasportato in altre posizioni all'interno della cella, quindi il baricentro è variabile.

Di seguito mostriamo i task, costante (Figura 5.21) e variabile (Figura 5.22), desiderati, senza però considerare i delta generati per la presenza della persona.

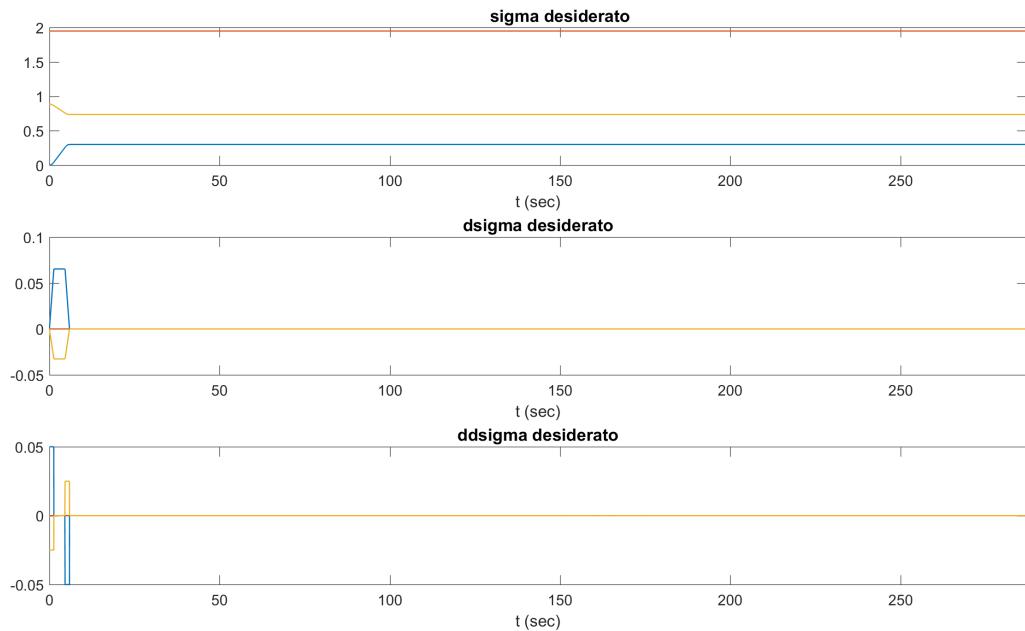


Figura 5.21: Task costante desiderato

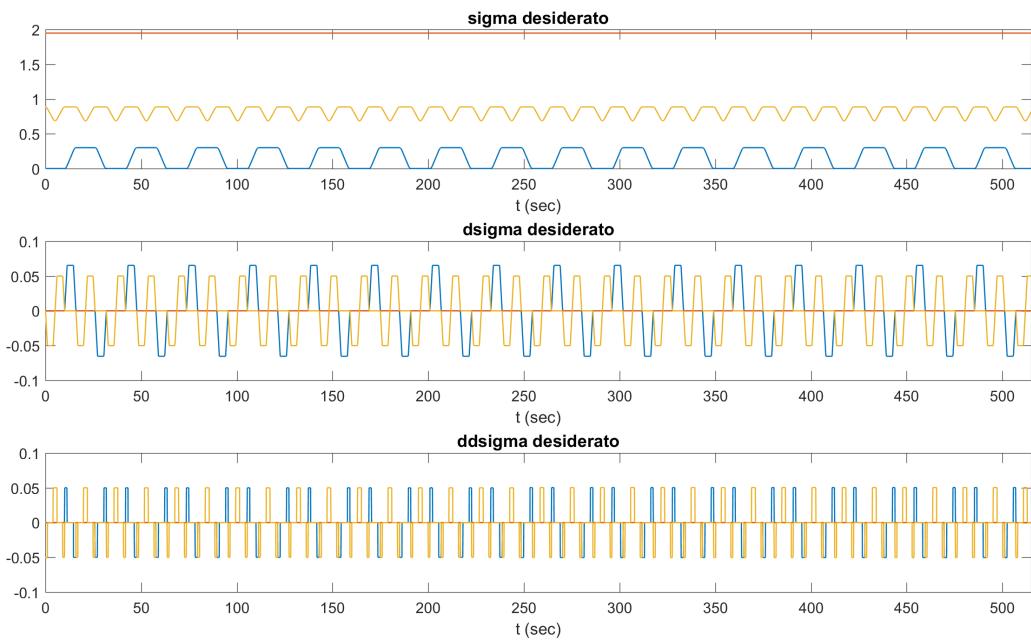


Figura 5.22: Task per il trasporto di un oggetto da parte dei robot Comau, da un punto ad un altro della cella, in modo periodico

Ciascun robot esegue dapprima una fase di inizializzazione mediante la quale si porta in una configurazione iniziale desiderata, successivamente attende che l’altro robot sia disponibile alla comunicazione e, infine, quando entrambi sono pronti per comunicare, viene eseguito l’algoritmo vero e proprio per la realizzazione del comportamento cooperante desiderato.

5.7 Risultati ottenuti

Il task che viene eseguito dai robot Comau, come detto in precedenza è modificato in base alla presenza o meno di una persona nella cella di lavoro e da quanto questa si avvicina all’oggetto trasportato. Per questo motivo il task effettivamente eseguito è il seguente:

$$\begin{cases} \boldsymbol{\sigma}_d(t) = \boldsymbol{\sigma}_d(t) + \Delta\boldsymbol{\sigma}(t) \\ \dot{\boldsymbol{\sigma}}_d(t) = \dot{\boldsymbol{\sigma}}_d(t) + \Delta\dot{\boldsymbol{\sigma}}(t) \\ \ddot{\boldsymbol{\sigma}}_d(t) = \ddot{\boldsymbol{\sigma}}_d(t) + \Delta\ddot{\boldsymbol{\sigma}}(t) \end{cases} \quad (5.34)$$

Dunque il task desiderato si modifica come nelle Figure 5.23 e 5.24.

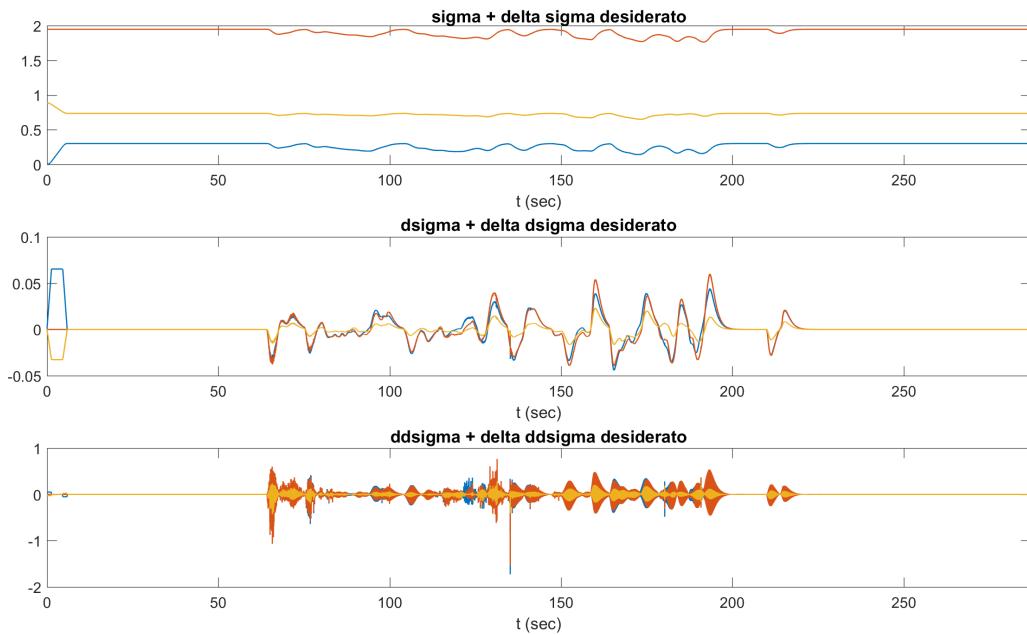


Figura 5.23: Task costante desiderato modificato dai delta generati per la presenza della persona all'interno della cella di lavoro

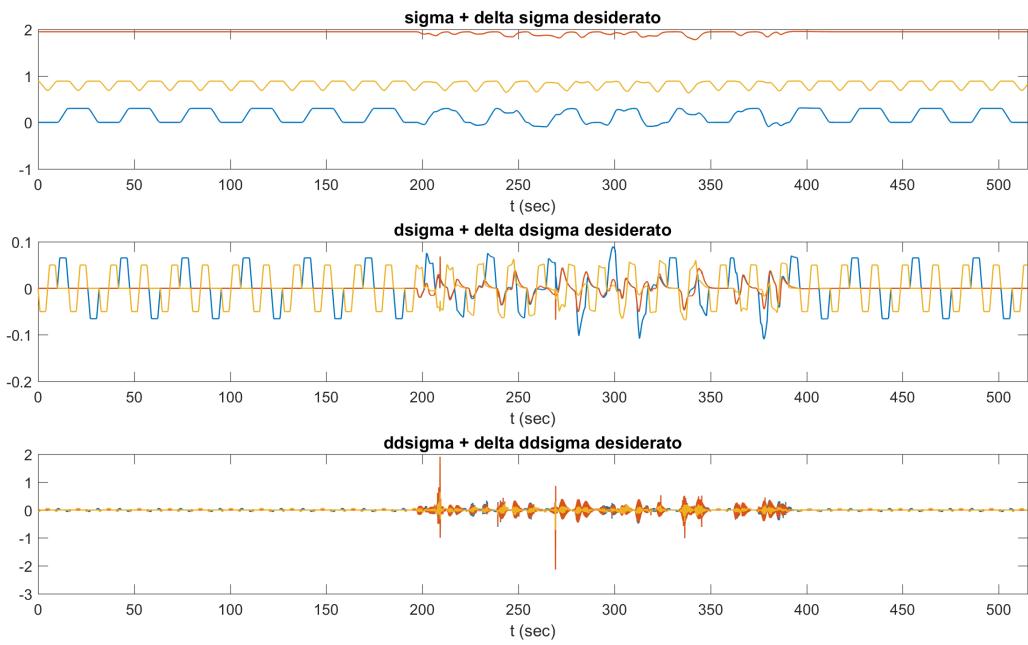


Figura 5.24: Task variabile desiderato modificato dai delta generati per la presenza della persona all'interno della cella di lavoro

Si nota in entrambe le figure che in corrispondenza dell'avvicinamento di una persona si generano delle accelerazioni molto più elevate e questo dipende da come

è stata definita la legge di impedenza. Da questa osservazione risulta facilmente individuabile dai grafici l'istante in cui una persona viene identificata perché entra nella cella di lavoro e l'istante in cui questa esce.

Nelle Figure 5.25 e 5.26 sono mostrati i grafici relativi agli errori sull'inseguimento del task e sulla stima dello stato globale sia nel caso di task costante che di task variabile.

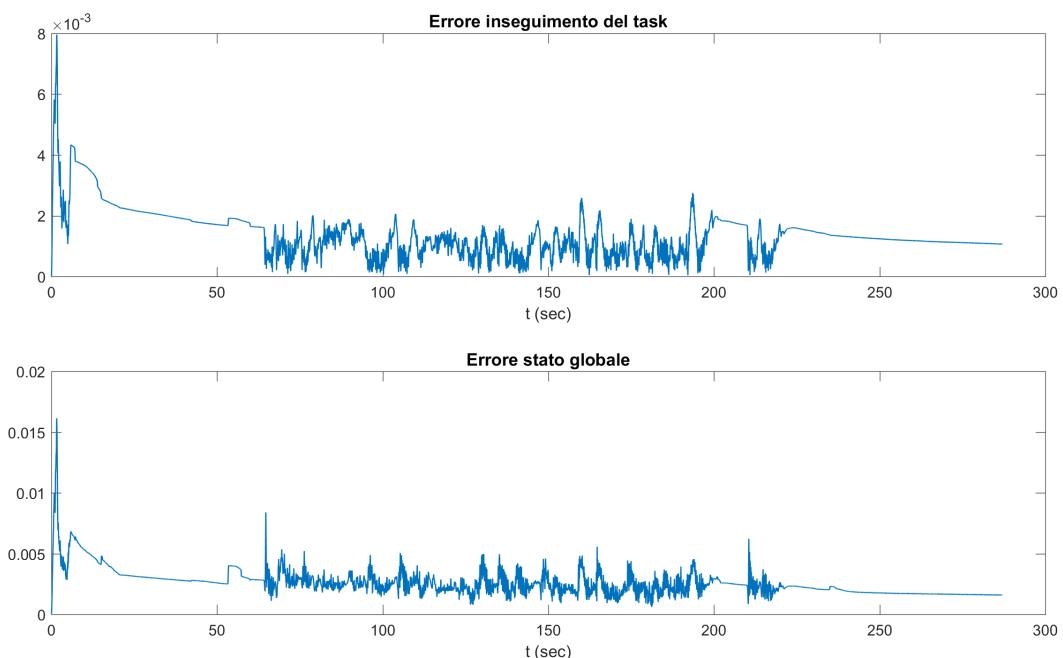


Figura 5.25: Errori sull'inseguimento del task e sulla stima dello stato globale nel caso di task costante

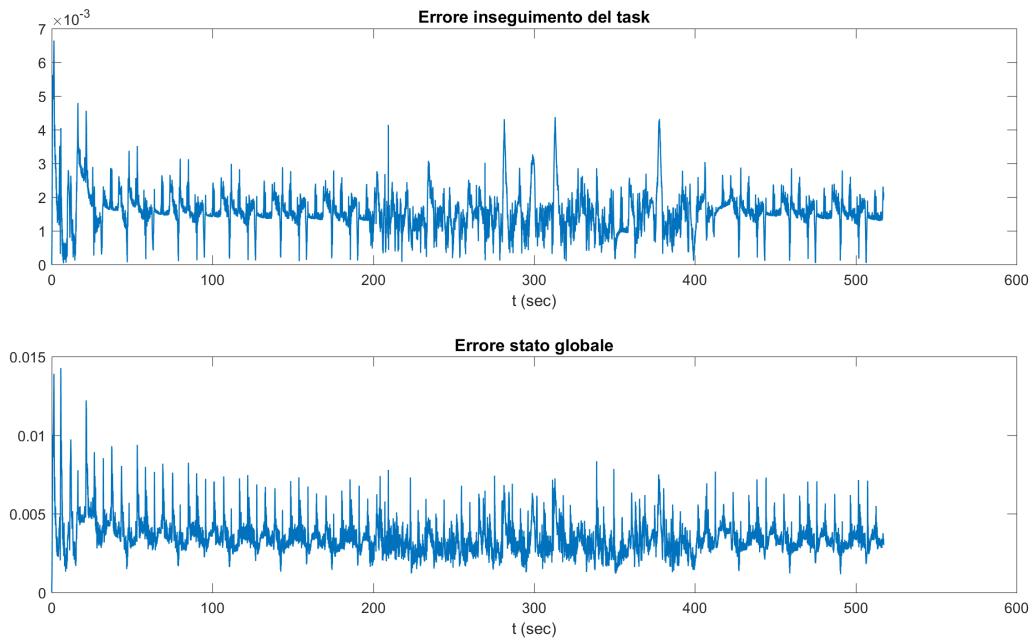


Figura 5.26: Errori sull'inseguimento del task e sulla stima dello stato globale nel caso di task variabile

I grafici evidenziano gli stessi comportamenti descritti nella parte simulativa, infatti essi convergono verso l'origine. Si osservano delle differenze con gli errori ottenuti in ambiente simulativo, infatti si notano maggiori oscillazioni e una convergenza dell'errore molto più lenta. Ciò è legato alla presenza di attrito statico su entrambi i robot il quale non è modellato nella legge di controllo. Se sono presenti errori di controllo a basso livello allora la stima dello stato non risulta coerente con lo stato effettivo del sistema, per cui la legge di controllo globale che fa uso della stima dello stato globale non permetterà di inseguire il task perfettamente ottenendo dunque un errore sull'inseguimento diverso da zero.

Le coppie richieste per eseguire i task sono mostrate nelle Figure 5.27 e 5.28.

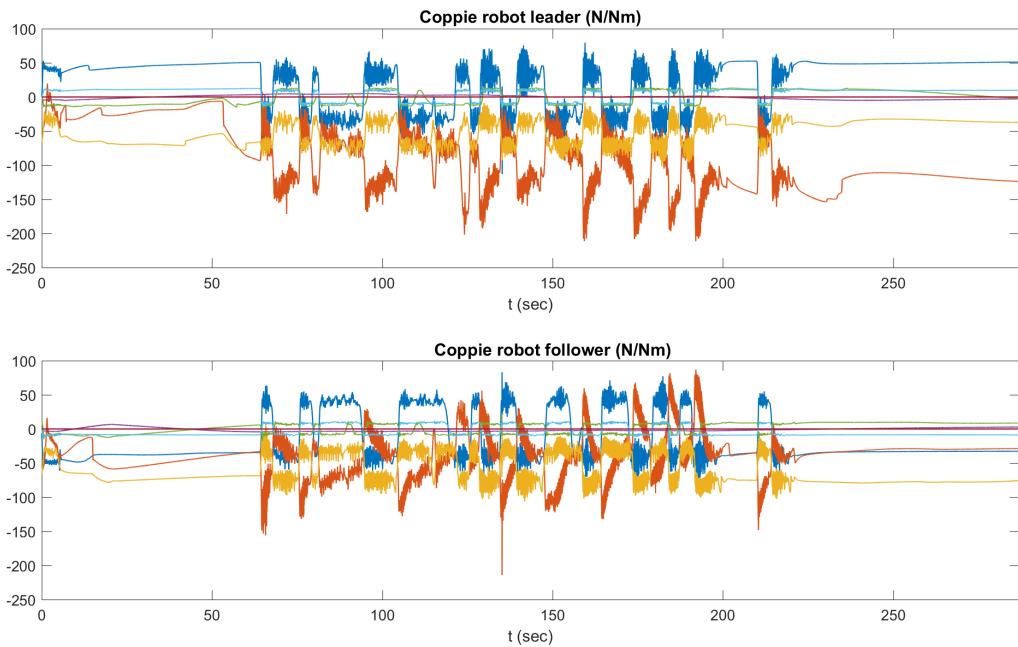


Figura 5.27: Coppie richieste nel caso di task costante

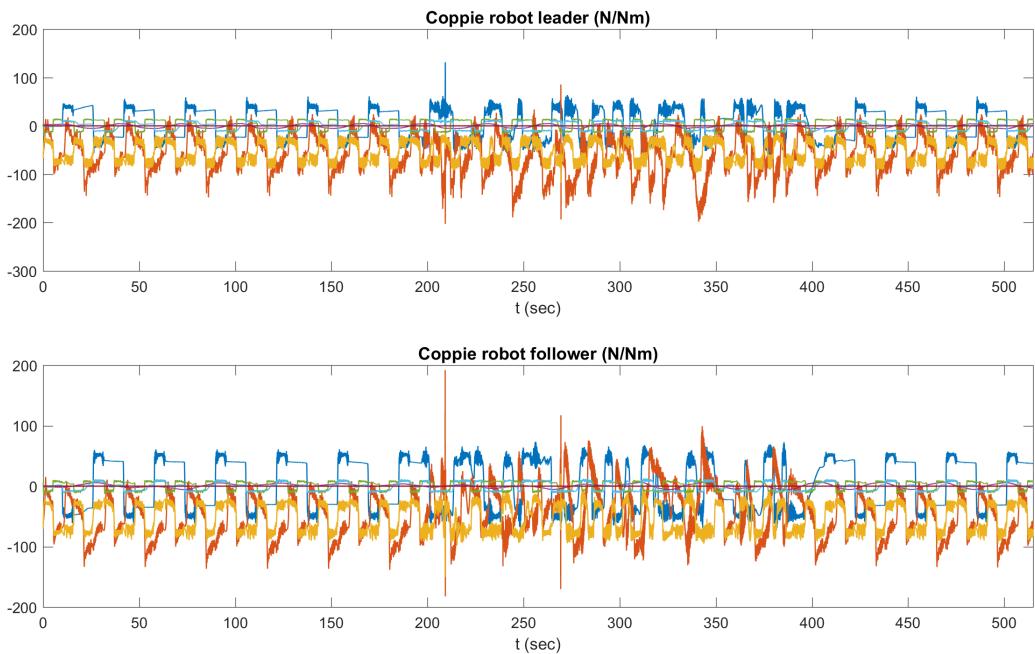


Figura 5.28: Coppie richieste nel caso di task variabile

Siccome i robot Comau lavorano ad un tempo di campionamento di 0.002 secondi e il robot UR10 ad un tempo di campionamento di 0.008 secondi, c'è un disallineamento tra le due esecuzioni. In particolare si verifica che i robot Comau ricevono una risposta dal robot UR10 solo ogni 4 cicli. Per risolvere questa pro-

blematica per gli istanti in cui non viene ricevuto nulla viene considerato sempre l'ultimo $\Delta\sigma(t)$ letto da socket e viene utilizzato un filtro del secondo ordine in modo da non avere un $\Delta\sigma(t)$ composto da tanti gradini ma una traiettoria che varia dolcemente.

Di seguito (Figura 5.29) viene mostrato il movimento della persona nella cella lungo i tre assi. Le posizioni mostrate sono relative alla testa in quanto il robot UR10 si muove rispetto alla persona considerando la terna associata alla testa, e alla mano destra in quanto è stato scelto di utilizzare il frame associato ad essa per calcolare la distanza dall'oggetto trasportato. Questa scelta discende dalla possibilità di avere spostamenti più ampi senza avvicinarsi troppo con tutto il corpo ai robot Comau, dunque la ragione è la sicurezza della persona. L'ampiezza maggiore dei movimenti della mano rispetto alla testa sono evidenziati dai grafici in Figura 5.29.

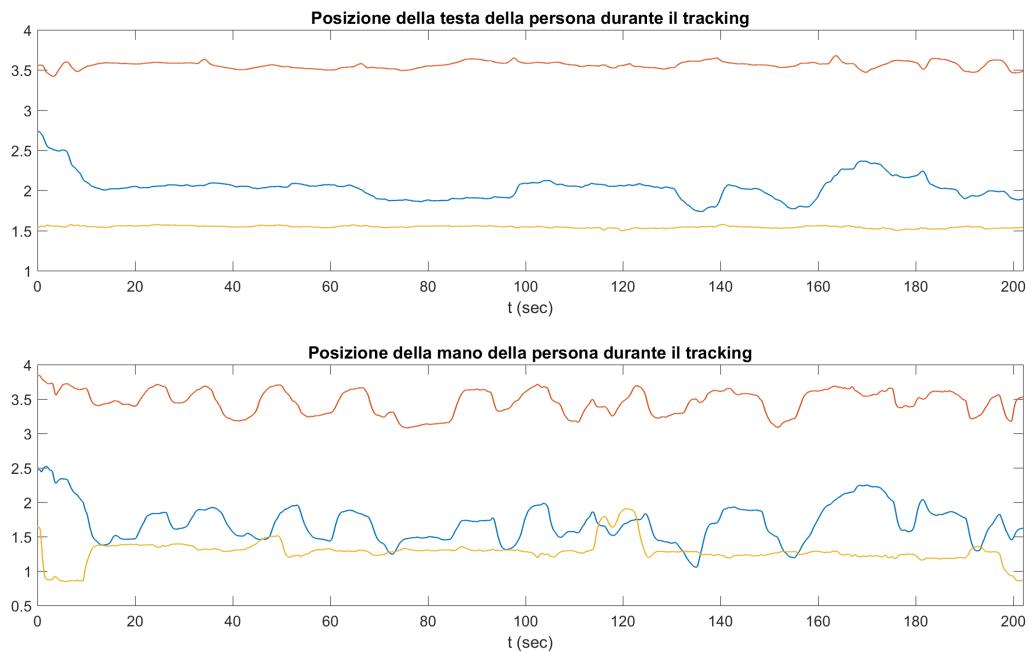


Figura 5.29: Posizioni della persona durante il tracking

Come è visibile dalla Figura 5.13, per la legge di impedenza viene generata una forza diversa da zero a partire da una distanza della persona dall'oggetto di 2.3 metri e questo è verificabile guardando i grafici della Figura 5.30.

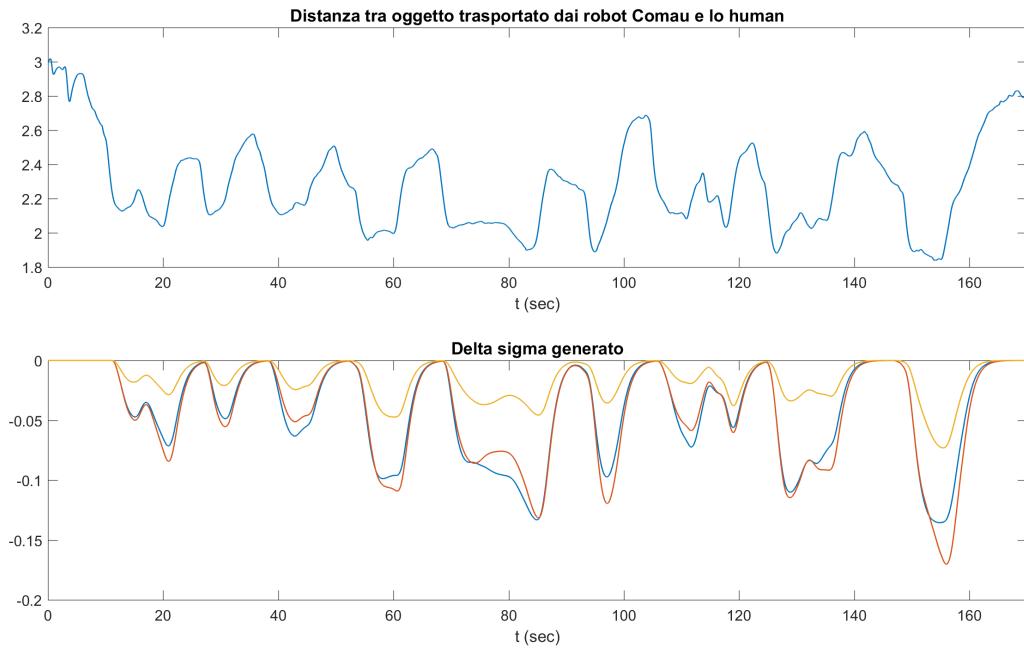


Figura 5.30: Distanza della persona dall'oggetto trasportato e delta sigma generato sulla base della distanza

Mostriamo ora l'errore generato dall'inversione cinematica implementata con il CLIK (Closed Loop Inverse Kinematics) del primo ordine per il robot UR10 (Figura 5.31) sia nella fase di ricerca di una persona nella cella (inversione cinematica di traiettoria pianificata offline) sia nella fase di tracking di una persona (inversione cinematica di traiettoria pianificata online). Le traiettorie pianificate sono mostrate invece in Figura 5.32.

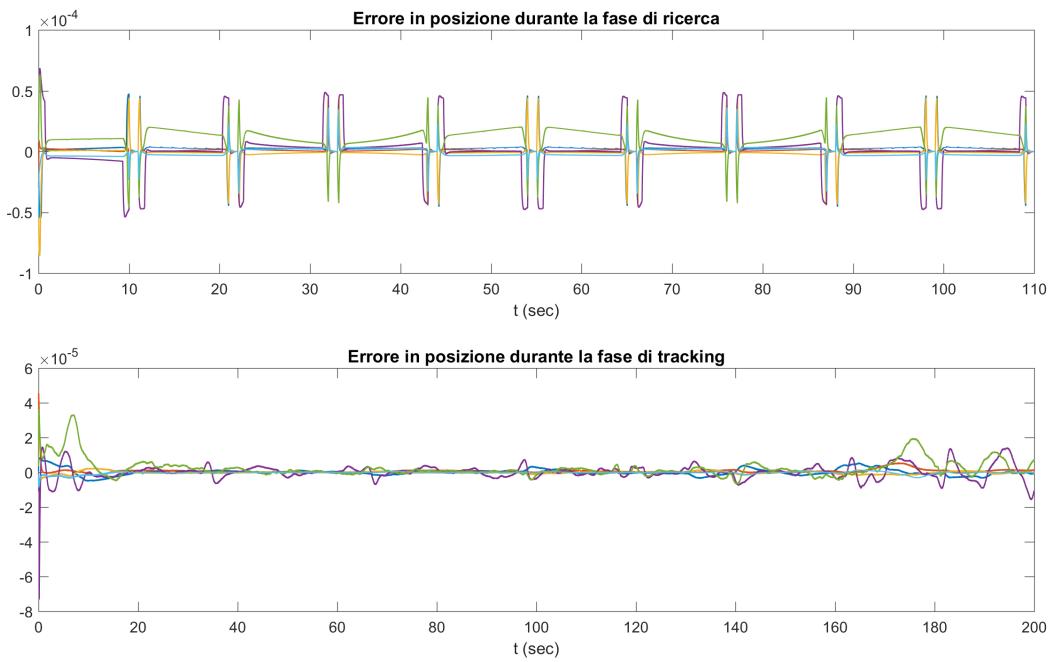


Figura 5.31: Errore di inversione cinematica

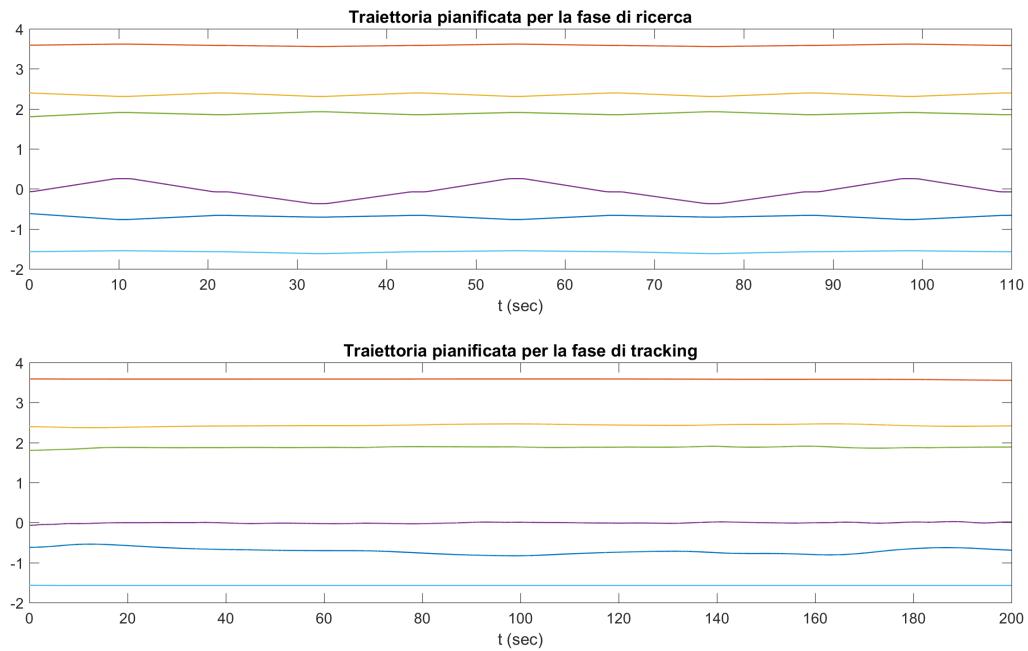


Figura 5.32: Traiettoria pianificata per la fase di ricerca e per la fase di tracking per il robot UR10

La distanza tra il robot UR10 e la persona durante tutta la durata della fase di tracking è mostrata in Figura 5.33.

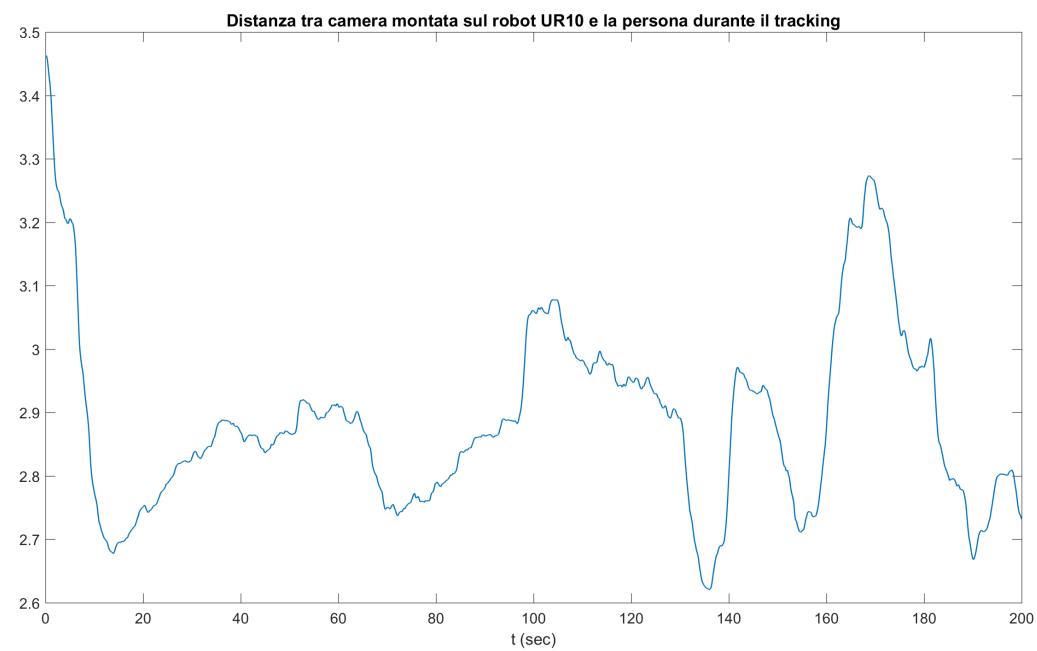


Figura 5.33: Distanza tra il robot UR10 e la persona durante la fase di tracking

Conclusioni e sviluppi futuri

Questo lavoro di tesi ha avuto lo scopo di progettare e sviluppare un'architettura che garantisca la sicurezza di operatori umani in scenari multi-robot decentralizzati. Questi scenari al giorno d'oggi sono sempre più frequenti in quanto si è nel pieno di una rivoluzione industriale chiamata Industry 4.0, la quale prevede che i robot non siano più costruiti per svolgere solo ruoli ripetitivi ma che siano flessibili di fronte ai task da svolgere. La rivoluzione prevede anche di ritrovarsi sempre più frequentemente in ambienti condivisi dall'uomo e dai robot e per questo motivo la sicurezza dell'uomo diventa un fattore molto importante.

Lo scenario analizzato in questo lavoro di tesi prevede pertanto la coesistenza di robot Comau Smart-Six che svolgono task cooperativi sulla base di un'architettura decentralizzata e un robot UR10 di supervisione che ha il ruolo di visionare la cella per rilevare la presenza di operatori umani e dunque garantire la loro sicurezza modificando il proprio comportamento e quello dei robot cooperanti.

Nell'algoritmo proposto, in accordo al paradigma di controllo decentralizzato, ciascun robot è un'entità attiva del sistema ed esegue autonomamente un algoritmo di controllo.

Per quanto riguarda i robot cooperanti, ogni robot implementa un'architettura multi-livello composta da: un primo livello che ha il ruolo di osservatore globale ed ha come scopo quello di stimare lo stato globale del sistema ed un secondo livello che sulla base della stima del livello superiore determina la legge di controllo locale per portare a termine il task globale desiderato. In questo modo, il secondo livello si comporta come se l'architettura fosse centralizzata perché ha a disposizione tutte le informazioni riguardanti il resto del sistema di cui fa parte

in quanto esse sono state stimate. Inoltre, si pone il problema che nell'utilizzo di robot reali il modello dinamico di essi non è mai perfettamente noto, quindi viene introdotta una componente di adattamento per compensare l'incertezza. La soluzione proposta è valida per due tipologie di applicazione: compiti di puro coordinamento tra robot e manipolazione di oggetti flessibili.

Per la manipolazione invece di oggetti rigidi è stato necessario aggiungere un'ulteriore componente alla soluzione finora descritta e riguarda il controllo delle forze di interazione per la compensazione delle forze che vanno a stressare l'oggetto trasportato.

Il robot UR10 come detto precedentemente, deve invece supervisionare la cella di lavoro in cui sono collocati i robot Comau che svolgono il proprio task cooperativo. Per fare questo esso assume due diversi comportamenti. In primo luogo esegue una fase di ricerca finché una persona non viene identificata. Questa fase prevede che il robot si sposti in vari punti dello spazio per inquadrare ogni parte della cella attraverso pianificazioni di traiettorie spazio operativo e attraverso la loro inversione cinematica. Non appena una persona viene individuata il comportamento del robot cambia, in quanto inizia ad inseguirla in posizione ed orientamento. Tutto questo avviene attraverso una pianificazione online del moto spazio operativo e la sua inversione. Lo scopo è mantenere la persona sempre a centro immagine (immagine catturata dalla kinect montata sull'ultimo giunto del robot) e mantenere il robot sempre ad una certa distanza dalla persona in modo da garantirne la sicurezza. In questa fase l'UR10 ha un ulteriore compito, che è quello di modificare il task che stanno svolgendo i robot cooperanti nel caso in cui la persona si avvicini troppo all'oggetto che essi stanno trasportando. Questo fa sì che anche tra l'oggetto trasportato e la persona siano mantenuti sempre ad una certa distanza di sicurezza.

Il lavoro presentato può essere ulteriormente ampliato sia per quanto riguarda l'architettura di controllo che lo scenario di interazione uomo-robot. Infatti si può pensare di testare più applicazioni e di diverso genere, di considerare all'interno del team, per la cooperazione, robot eterogenei. Diventa importante inoltre

testare tutte queste applicazioni non solo in ambiente simulativo ma anche reale. Si può pensare anche di implementare meccanismi per garantire la robustezza ai guasti.

Sarebbe interessante poi per l'interazione uomo-robot gestire situazioni differenti da quella analizzata. Ad esempio si può considerare una collaborazione tra operatori e robot per permettere ad essi di lavorare mantenendo una distanza anche nulla. Si può pensare ancora di considerare scenari più ampi con più robot supervisori e più operatori umani.

Ancora un'altra possibile applicazione potrebbe essere quella di considerare l'uomo appartenente al team cooperante per portare ad esempio a termine un task di formazione considerando un'architettura di tipo leader-follower in cui l'uomo assumerebbe il ruolo di leader. Con questo scenario si avrebbe una variazione in tempo reale del task in quanto i comportamenti dell'uomo non sono pianificabili e dunque diventa necessario considerare algoritmi per la stima del task da parte dei restanti componenti del team, ovvero i follower.

Bibliografia

- [1] Fabrizio Caccavale e Masaru Uchiyama. «Cooperative Manipulators». In: *Springer Handbook of Robotics*. A cura di Bruno Siciliano e Oussama Khatib. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 701–718.
- [2] *Collaborazione uomo-robot*. URL: <https://www.kuka.com/it-it/tecnologie/collaborazione-uomo-robot>.
- [3] Alessandro De Luca e Raffaella Mattone. «Sensorless robot collision detection and hybrid force/motion control». In: *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE. 2005, pp. 999–1004.
- [4] M Bernardine Dias e Anthony Stentz. «A free market architecture for distributed control of a multirobot system». In: *6th International Conference on Intelligent Autonomous Systems (IAS-6)*. 2000, pp. 115–122.
- [5] Alessandro Farinelli, Luca Iocchi e Daniele Nardi. «Multirobot systems: a classification focused on coordination». In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 34.5 (2004), pp. 2015–2028.
- [6] Frank L Lewis et al. «Algebraic Graph Theory and Cooperative Control Consensus». In: *Cooperative Control of Multi-Agent Systems*. Springer, 2014, pp. 23–71.
- [7] Martina Lippi. «Controllo decentralizzato di manipolatori mobili cooperanti: sintesi e sperimentazione». Tesi di laurea. Università degli Studi di Salerno, 2015/2016.

- [8] Alessandro Marino. «Distributed adaptive control of networked cooperative mobile manipulators». In: *IEEE Transactions on Control Systems Technology* (2017).
- [9] *Robot Operating System ROS*. URL: <http://wiki.ros.org/it>.
- [10] Jürgen Rossmann. «eRobotics Meets the Internet of Things: Modern Tools for Today's Challenges in Robotics and Automation». In: *Developments of E-Systems Engineering (DeSE), 2015 International Conference on*. IEEE. 2015, pp. 318–323.
- [11] Michael Rüßmann et al. «Industry 4.0: The future of productivity and growth in manufacturing industries». In: *Boston Consulting Group* 9 (2015).
- [12] R Olfati Saber e Richard M Murray. «Agreement problems in networks with directed graphs and switching topology». In: *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*. Vol. 4. IEEE. 2003, pp. 4126–4132.
- [13] Bruno Siciliano et al. *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.
- [14] *Universal Robot UR10*. URL: <https://www.universal-robots.com/it/prodotti/robot-ur10/>.