POLITECNICO DI MILANO

Scuola di Ingegneria Industriale e dell'Informazione
Corso di Laurea Magistrale in Ingegneria dell'Automazione

POLITECNICO
MILANO 1863

Assemblaggio robotizzato e bimanuale con introduzione di un sistema di controllo visivo

Relatore: Prof. Paolo Rocco
Correlatore: Ing. Matteo Parigi Polverini

Tesi di Laurea Magistrale di:
Alessandro Abbondanza Matr. 851729

Anno Accademico 2016-2017

A tutte le persone che mi vogliono bene...

Ringraziamenti

In primis ringrazio il Prof. Paolo Rocco per avermi assegnato questa tesi riguardo la robotica industriale. Lo ringrazio specialmente per avermi assegnato un lavoro non puramente teorico, come da me richiesto, e che ha soddisfatto molto più delle mie aspettative iniziali. Lo ringrazio anche per le conoscenze teoriche necessarie e i consigli per lavorare su questo argomento. Ringrazio l'Ing. Matteo Parigi Polverini per avermi accompagnato durante questo mio percorso, per avermi fornito le conoscenze specifiche sull'argomento e per tutta la disponibilità e comprensione fornitami. Ringrazio l'Ing. Andrea Maria Zanchettin per avermi aiutato nella parte di realizzazione dell'esperimento, nella quale le mie conoscenze di base erano nulle. Lo ringrazio specialmente per avermi aiutato e sopportato anche laddove i problemi da risolvere potessero risultare banali. Ringrazio Renzo Villa, Davide Nicolis e Andrea Casalino per il supporto fornitomi durante le ore di laboratorio. Ringrazio tutti i colleghi di laboratorio per i momenti felici passati insieme durante questo percorso intrapreso insieme fin dal primo anno di università. Ringrazio i miei genitori e i miei nonni per avermi sempre supportato durante tutta la mia vita e in tutte le mie scelte. Ringrazio tutti i miei amici, che considero come fratelli, con i quali sono cresciuto e che ancora oggi mi vogliono bene. Infine ringrazio Silvia, mia ex ragazza e compagna di percorso universitario, con la quale ho condiviso i miei ultimi quattro anni. La ringrazio per avermi capito e rincuorato nei momenti più difficili, per avermi reso felice, migliore, per avermi arricchito molto come persona.

Abstract

The increasing request for implementation of automatic methodologies aimed at problem solving, has favoured a great increase in the usage of robots, or in general of automatic devices, in industrial plants of the most diverse types. Starting from totally automated assembly lines, composed only of industrial robots, we arrive to hybrid assembly lines, in which a human-robot collaboration happens.
This new robot integration with human work has been possible thanks to a new kind of manipulator. Lightweight Dual Arm robots, such as ABB YuMi or Rethink Baxter, are now a commercial product with relatively low costs.
This new kind of robot is useful mainly for a reason: it has a great adaptability to the environment. Thanks to their really low weight, these robots can be easily moved around, and thanks to their structure, they can do assembly operation, autonomously or in collaboration with an operator, without using fixtures.
However, the set of problems that can be solved using only the manipulator is limited. The vision can help to increase the number of solvable problems. In assembly operation, the greatest part of the body to be assembled has a particular shape, or structure, that must fit with the shape, or structure, of a corresponding object. A robot without sensors doesn't have a way to recognize the technical features of the manipulated objects, and vision can give it that knowledge.
This thesis proposes the assembly of a lab pipe with the introduction of vision control. The problem will be addressed from a geometric point of view and a control algorithm for its solution will be implemented.
Experimental results on a dual arm robot manipulator will be shown.

Key words: Industrial Robotics, Dual Arm robot, Constrained Optimization, Vision Control.

Sommario

La crescente richiesta di metodologie automatiche per la risoluzione di svariati problemi ha favorito un grande incremento dell'utilizzo di robot, o in generale dispositivi automatizzati, nei settori industriali di tutte le tipologie. A partire da linee di produzione totalmente automatizzate costituite da soli robot industriali, si arriva a linee di assemblaggio ibride, in cui avviene una collaborazione tra uomo e robot.
Questa nuova integrazione di robot con l'uomo è stata possibile grazie ad una nuova tipologia di manipolatore. I robot leggeri a doppio braccio, come ad esempio YuMi di ABB e Baxter di Rethink, sono ora un prodotto commerciale, facilmente reperibile e con costi relativamente bassi.
Questa tipologia di robot è utile principalmente per un motivo: ha una grandissima adattabilità all'ambiente di lavoro. Grazie al peso bassissimo, questi robot sono facilmente trasportabili e, grazie alla loro struttura, riescono a svolgere operazioni di assemblaggio, autonomamente o in collaborazione con un operatore, senza servirsi di punti fissi di appoggio nell'ambiente di lavoro.
L'insieme di problemi risolvibili dal solo manipolatore è però limitato. A tal proposito, per aumentare la dimensione di questo insieme, la visione può essere di aiuto. Nelle operazioni di assemblaggio, infatti, la maggior parte dei corpi assemblabili ha una particolare forma, o struttura, che deve combaciare con la forma, o struttura, corrispondente di un altro oggetto. Un robot senza sensori non ha modo di riconoscere le caratteristiche tecniche negli oggetti che manipola e la visione può fornire queste conoscenze.
A tal riguardo, questa tesi propone l'assemblaggio di una pipetta da laboratorio con necessaria introduzione di un controllo di visione. Si discuteranno il problema dal punto di vista geometrico e l'algoritmo di controllo implementato per la sua soluzione.
Verranno mostrati anche risultati sperimentali su un manipolatore robotico a doppio braccio.

Parole chiave: Robotica Industriale, Robot Bimanuale, Ottimizzazione vincolata, Controllo di Visione.

Indice

Elenco delle figure

Capitolo 1

Introduzione

Lo sviluppo dell'industria automatizzata è stato esponenziale negli ultimi anni ed è stato accompagnato da una forte accelerazione della ricerca sulla robotica.
Inizialmente l'attenzione era stata rivolta all'utilizzo di robot a singolo braccio. L'utilizzo di questa tipologia di manipolatore era funzionale, ma molto limitante dal punto di vista delle applicazioni. L'assemblaggio di parti multiple o la manipolazione di oggetti flessibili, ad esempio, riscontravano migliori risultati se risolte con una configurazione a doppio braccio. Le prime ricerche in questo campo sono state effettuate al fine di consolidare la collaborazione tra bracci robotici convenzionali. Successivamente sono comparsi sul mercato robot collaborativi a doppio braccio.
Con robot collaborativo si intende una tipologia di manipolatore che possa aiutare un operatore umano in compiti che richiedano il sollevamento di carichi eccessivi, la realizzazione di operazioni ripetitive o l'esecuzione di compiti che richiedano alta precisione.

Figura 1.1: Esempio di collaborazione human-robot.

L'introduzione di questa nuova tipologia di robot, inoltre, ha favorito lo sviluppo e l'appli-

cazione della robotica in nuovi contesti. Uno di questi è il campo farmaceutico nel quale manipolatori collaborativi a doppio braccio possono essere utilizzati per l'assemblaggio di componenti, l'estrazioni di liquidi o altre operazioni nelle quali è richiesta molta precisione. L'utilizzo di questi manipolatori ha permesso la riduzione dei tempi di lavoro e dei rischi per operatori umani.
Per permettere l'applicazione di robot in nuovi ambienti di lavoro si sono studiate nuove metodologie di controllo basate su sensori esterni.
Particolare attenzione merita l'utilizzo della visione artificiale per la soluzione di problemi in maniera autonoma da parte di un robot. Senza visione un robot può risolvere un compito autonomamente, ad esempio, sfruttando degli algoritmi basati sullo studio di forze esterne percepite sull'end-effector. Questa tipologia di approccio, però, non sempre è ottima sia dal punto di vista dei tempi, sia dal punto di vista della metodologia. Infatti spesso il solo approccio di controllo basato sulla percezione di forze non riesce a garantire un quadro completo dell'azione che si vuole completare. Con l'introduzione della visione si dà la possibilità di migliorare gli algoritmi precedenti combinandoli con nuove soluzioni basate sullo studio di immagini. Ad esempio in [1] viene affrontato un problema di Peg-in-Hole con due differenti approcci, uno basato sulle forze percepite e uno basato sulla visione. I risultati mostrano come il controllo di visione possa migliorare i tempi di soluzione e la correttezza del problema.
Parallelamente alla crescita del campo della robotica, è avvenuto uno sviluppo nel campo della visione, con una grandissima evoluzione dei dispositivi visivi. Esempi di questi possono essere il Kinect, che permette oltre all'acquisizione di immagine, il riconoscimento dei punti scheletali di un eventuale essere umano presente nel campo visivo, o Xtion, il quale permette il riconoscimento dei movimenti. Questo ha permesso, oltre ad ottenere una più grande varietà e completezza di dati ricavabili, una sempre più diffusa applicazione, in campi differenti, di sistemi di controllo basati sulla visione grazie ai costi decrescenti dei dispositivi per la raccolta di immagini.

1.1 Obiettivo della tesi

L'obiettivo di questa tesi è l'assemblaggio robotizzato e bimanuale. In particolare l'attenzione verrà concentrata sulla possibilità di implementare nuovo algoritmi di controllo basati sulla visione, applicati a robot cinematicamente ridondanti Dual Arm.
In questa tesi si cercherà di compiere, come principale operazione, il montaggio completo di una pipetta (Fig 1.2), suddivisa in due sotto azioni:

- Inserimento dello stantuffo nel corpo;
- Avvitamento del cappuccio.

Nello specifico, verranno usati i componenti mostrati in figura (1.2) per eseguire il montaggio: un cappuccio, uno stantuffo e un corpo di una pipetta da laboratorio.

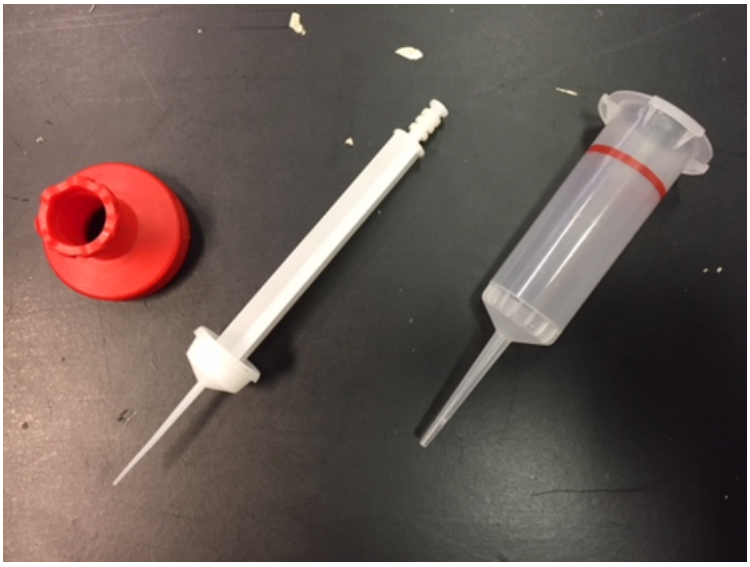

Figura 1.2: Da sinistra: cappuccio, stantuffo, corpo.

Per l'esecuzione di questi compiti viene utilizzato un robot a doppio braccio (Dual Arm) cinematicamente ridondante. L'operazione da eseguire è facilmente risolvibile da un uomo e la natura di questo robot è ideale per descrivere la struttura delle braccia umane.
Per la prima operazione verrà utilizzato un controllo di impedenza [2]. Il controllo di impedenza assicura un corretto posizionamento delle braccia per l'esecuzione di un task in caso di errori nel posizionamento iniziale.
Per la seconda operazione, invece, verranno applicate due strategie di controllo differenti:

1. Una prima basata sulla visione per un corretto posizionamento relativo tra le braccia del robot. Questo è il principale contributo per una corretta soluzione del problema affrontato;

2. Una seconda basata su un controllo di impedenza, simile a quella descritta per la prima operazione, congiunto ad un controllo di forza, per la corretta esecuzione dell'avvitamento.

La strategia del controllo di forza nella seconda fase deve assicurare che in ogni istante la forza sia compresa tra un valore minimo e un valore massimo, tenendo in considerazione eventuali incertezze nell'ambiente [3].
Il controllo basato sulla visione permette la corretta esecuzione del processo, correggendo eventuali errori nella disposizione iniziale degli oggetti manipolati. La forma degli oggetti manipolati è particolare e senza il contributo di un controllo basato sulla visione, il controllo di impedenza e il controllo di forza non sarebbero sufficienti alla corretta esecuzione dell'operazione di avvitamento. Il cappuccio della pipetta presenta tre fessure nelle quali dovranno andare ad incastrarsi le sporgenze presenti sul corpo della pipetta. L'algoritmo di visione permette il riconoscimento di queste forme e riesce a garantire il corretto inserimento. L'unica alternativa al controllo di visione vedrebbe come soluzione

il posizionare gli oggetti da manipolare in una determinata posizione iniziale prefissata ad ogni esperimento.
Tutte le strategie di controllo vengono implementate senza l'uso di sensori di forza. Questi ultimi, infatti, sono troppo pesanti e ingombranti per poter essere utilizzati sul robot, il quale può sollevare un carico massimo di 0.5Kg.
Dal momento in cui il robot non è equipaggiato con un sensore di forza, è stato adottato uno stimatore delle forze/momenti esterne basato sul modello dinamico del robot [4].
Come dimostrazione del lavoro effettuato, verrà eseguita un'implementazione a livello sperimentale utilizzando il robot Dual Arm ABB FRIDA.

Figura 1.3: Robot ABB FRIDA.

1.1.1 Risultati conseguiti

La definizione dell'operazione di assemblaggio viene effettuata tramite l'analisi delle possibili operazioni effettuabili con approccio a doppio braccio. Questa formalizzazione è stata prima utilizzata in ambito di simulazione e poi in ambito sperimentale. Un algoritmo di controllo basato sulla visione è stato introdotto con successo all'interno dello schema di controllo completo del robot. Questo controllo è stato necessario per fornire realizzabilità all'esperimento senza prefissare le condizioni iniziali sulla posizione degli oggetti manipolati. Sono stati riportati i comportamenti desiderati per tutte le operazioni necessarie al completamento del processo.
L'utilizzo del controllo basato sulla visione ha permesso di effettuare una corretta correzione degli errori riportati nell'orientamento degli oggetti manipolati. Il filtro di impedenza permette la correzione di errori nella posizione; il controllo di forza garantisce il contatto tra gli oggetti.
I risultati mostrano come l'algoritmo di controllo di visione vada a modificare l'orienta-

mento degli oggetti manipolati, in particolare del corpo e del cappuccio della pipetta, nel momento in cui essi si trovino in una posizione errata rispetto a quella desiderata. Questa strategia è utile quando si vuol fornire casualità alle condizioni iniziali del problema, non fornendo delle posizioni predefinite agli oggetti da manipolare. Da notare quanto questa strategia sia necessaria al fine di evitare rotture o malfunzionamenti durante il processo.

1.2 Struttura della tesi

La tesi sarà sviluppata su otto capitoli, che andranno a ripercorrere il percorso lavorativo effettuato su questo progetto.

- Nel primo capitolo verrà introdotta la collaborazione uomo-robot. Nello stato dell'arte verranno trattate la collaborazione tra manipolatori multipli, l'approccio all'assemblaggio bimanuale e la risoluzione dei problemi tramite Computer Vision.

- Nel secondo capitolo verranno introdotte le formalizzazioni delle operazioni Dual Arm. A partire da una descrizione generale delle operazioni previste, di inserimento e avvitamento, verranno analizzate geometricamente nello specifico del nostro problema le medesime operazioni. Verrà prestata particolare attenzione ai parametri scelti e alle problematiche affrontate. Come ultimo passaggio, verrà descritto lo Jacobiano relativo, necessario per la definizione dei vincoli cinematici nell'algoritmo di ottimizzazione.

- Il terzo capitolo sarà dedicato agli algoritmi di ottimizzazione vincolata, implementati per l'esecuzione di operazioni bi manuali. Verrà descritto l'algoritmo di controllo, con i suoi principali componenti. Verrà prestata attenzione alla creazione delle funzioni di costo. Successivamente verrà descritta l'implementazione attraverso due solutori.

- Il quarto capitolo sarà destinato alla Computer Vision. Come primo paragrafo di questo capitolo verranno descritte le principali funzioni utilizzate in questa tesi. Queste verranno trattate prima dal punto di vista teorico, poi dal punto di vista applicativo. Successivamente verrà descritto un approccio al controllo basato sull'utilizzo di telecamera. Infine verrà prestata attenzione anche alle problematiche che possono sorgere durante l'applicazione.

- Il quinto capitolo verrà dedicato alla descrizione della stima delle forze e del modello di attrito. Questo passaggio risulta necessario per eseguire una corretta implementazione di algoritmi senza l'ausilio di sensori.

- Il sesto capitolo sarà dedicato alla descrizione del controllo di impedenza. Come primo passo, verrà descritta la teoria alla base della creazione di questo tipo di

controllo. Successivamente ne verranno descritti l'implementazione e i parametri utilizzati.

- Il settimo capitolo sarà dedicato alla descrizione dell'ambito sperimentale della tesi. Dopo aver descritto tutti i componenti dell'apparato sperimentale, verrà mostrata la creazione dei gripper per l'esecuzione del processo, con descrizione delle scelte effettuate a seguito delle problematiche riscontrate. Il capitolo proseguirà con la descrizione della macchina a stati realizzata per l'esecuzione del processo completo, soffermandosi sulle scelte effettuate. Per concludere il capitolo, verrà mostrato un esempio di applicazione del controllo di visione.

- Nell'ottavo e ultimo capitolo verranno presentate le conclusioni del lavoro della tesi, con principali problematiche affrontate. Verranno anche proposti possibili sviluppi successivi.

1.3 Stato dell'arte

In questa sezione verrà riportato lo stato dell'arte relativo alla manipolazione Dual Arm e verrà presentato lo stato dell'arte anche sull'approccio di soluzione ai problemi tramite Computer Vision.
La collaborazione tra robot è diventato un argomento molto discusso negli ultimi anni, vista la sempre più crescente domanda di soluzioni automatizzate per la risoluzione di problemi. In una classica catena di montaggio, diversi robot collaborano al fine di realizzare uno stesso prodotto. La dinamica di ogni manipolatore permette di conoscere l'interazione e lo scambio di forze tra robot e l'ambiente. In [5] vengono discusse la cinematica e la statica di un sistema composto da più robot, nello specifico M manipolatori a N giunti. A partire dalla posizione e dall'orientamento attuali del robot, viene definita la cinematica diretta di ogni manipolatore. Successivamente vengono introdotti lo Jacobiano analitico e lo Jacobiano geometrico di ogni singolo manipolatore, al fine di ottenere la cinematica differenziale.

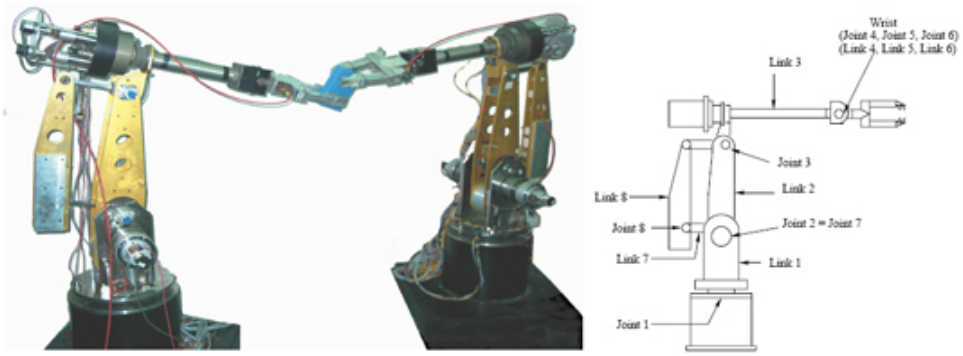

Figura 1.4: Esempio di robot collaboranti, con schema di struttura.

In letteratura sono proposti diversi approcci al problema di formalizzazione di operazioni per robot cooperanti. Degni di nota sono:

- Cooperative task space, finalizzato alla collaborazione contemporanea di robot per l'afferraggio di un medesimo oggetto [5]. Il cooperative task space viene descritto attraverso 4 variabili rappresentanti posizione e orientamento assoluti dell'oggetto afferrato, e posizione e rotazione relative delle braccia che lo manipolano. Questo approccio richiede l'utilizzo di 6 x M variabili. Nel caso in cui si voglia controllare il moto dell'oggetto manipolato basta agire su posizione e orientamento assoluti dello stesso, sfruttando posizione e orientamento relativi tra le braccia per il controllo di altre features. Questo approccio, inoltre, risulta molto utile se l'oggetto o l'end effector non risultino rigidi. Questa formalizzazione risulta, però, inutile nel caso in cui non si voglia manipolare il medesimo oggetto.

- Virtual Sticks [6]. Questo metodo si basa sulla creazione di vettori costanti che individuino la posizione di un punto qualsiasi dell'oggetto da manipolare rispetto ai punti di contatto con l'end effector. In questo metodo è fondamentale l'ipotesi che l'oggetto manipolato e l'end effector siano rigidi. Scegliendo opportunamente un insieme di angoli di Eulero, è possibile ricavare la posizione e l'orientamento dell'oggetto rispetto agli end effector. Allo stesso modo è possibile ricavare la velocità di un end effector a partire dalla velocità dell'oggetto manipolato, tramite il principio dei lavori virtuali attraverso forze generalizzate. In questo modo alle forze e velocità legate all'end effector, corrispondono forze e velocità legate al virtual stick.

- Multi fingers manipulation [7]. Formalizzazione finalizzata a ricondurre l'approccio multi braccio ad un approccio multi dita. L'end effector di ogni braccio è caratterizzato dalla presenza di più dita, di conseguenza solo alcune componenti del moto vengono trasmesse al resto del sistema. Per ottenere una corretta manipolazione deve avvenire uno scambio di forze e coppie nel punto in cui l'oggetto viene afferrato. Le forze interne garantiscono un'ottima presa anche nel caso in cui vengano applicate forze esterne.

La manipolazione a doppio braccio può essere vista come un caso particolare di collaborazione robotica, in cui due bracci robotici a N giunti collaborano per la soluzione del medesimo problema.
Le soluzioni per l'assemblaggio presentate in letteratura riguardano per la maggior parte l'approccio a singolo braccio, dove il braccio robotico manipola un componente da montare su un altro componente fissato nell'ambiente di lavoro. Il componente che si trova nell'ambiente di lavoro si trova in un punto operativo raggiungibile dal robot e la posizione e l'orientamento nominale sono noti a priori. [8] affronta la soluzione del problema di Peg-in-Hole ed è uno dei pochi articoli in cui l'apparato sperimentale è costituito da un

robot a doppio braccio. Ogni braccio è costituito da VIA (Variable Impedance Actuators) a quattro gradi di libertà. Entrambe le braccia risultano necessarie poichè il numero di gradi di libertà richiesto per la soluzione del problema è cinque. La tecnica di soluzione utilizzata si basa sulle particolari proprietà del VIA piuttosto che sulla geometria del problema: viene precalcolato un cammino da far compere ad entrambe le braccia e vengono determinati istanti di tempo in cui viene tentato l'inserimento. L'eventuale inserimento viene verificato dall'algoritmo che, qualora non fosse effettuato, fa riprendere la ricerca fino ad un nuovo tentativo casuale.

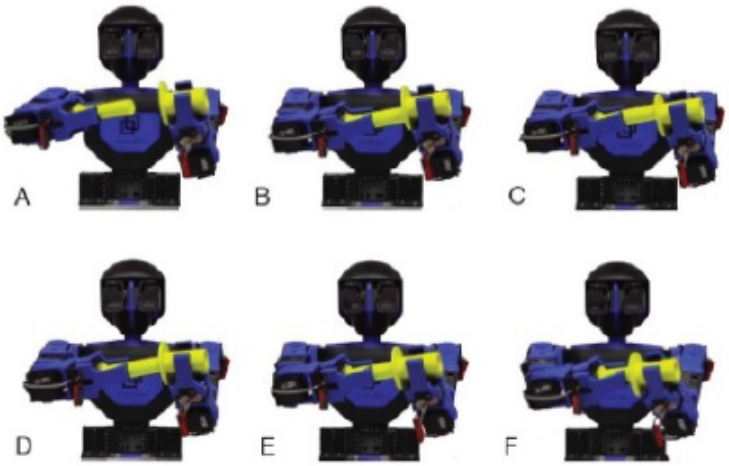

Figura 1.5: Fasi assemblaggio con VIA.

In Fig. (1.5) vengono mostrate le seguenti fasi: preinserimento (A), inizio fase di ricerca (B), ricerca (C), tentativo di inserimento (D), vibrazioni e riduzione errori di allineamento (E), inserimento completato (F).

Con la necessità di introdurre nuove tipologie di soluzione per la creazione di sistemi totalmente autonomi, la visione è stata fortemente studiata e sviluppata in modo tale da creare queste opportunità. Ad esempio in [1] viene trattata la soluzione del problema di Peg-in-Hole tramite l'utilizzo della telecamera.
Con Computer Vision si intende un campo interdisciplinare che si occupa di come i computer percepiscono immagini o video digitali.
Come trattato in molteplici articoli e libri sull'argomento, come ad esempio [9], [10], [11], [12], [13], la Computer Vision si occupa di acquisire le immagini, capirle, analizzarle e processarle.
L'utilizzo della Computer Vision dipende fortemente dall'applicazione nella quale si vuole operare: molto spesso vengono costruite funzioni uniche per la risoluzione di un problema specifico. Per questo motivo non si può generalizzare la risoluzione dei problemi sfruttando la visione, ma si può costruire uno schema logico da seguire per applicare un approccio

corretto alla soluzione. I 6 passi fondamentali da seguire per affrontare un problema con la visione sono:

1. Image acquisition: sfruttando un sensore d'immagine produrre un'immagine digitale. I sensori possono essere molteplici e, in base a quale si sceglie, possono fornire immagini 2D, immagini 3D, sequenze di immagini o insiemi di dati contenenti molteplici informazioni sulla stessa immagine (esempio: luminosità, profondità, calore, ecc.).

2. Pre-processing: processare un'immagine precedentemente la sua analisi, in modo tale da poterla sfruttare al meglio. Un esempio di pre-processing può riguardare l'aggiustamento della nitidezza o la focalizzazione su una specifica regione dell'immagine.

3. Feature extraction: estrarre delle specifiche da un'immagine. Ad esempio vengono estratti punti, linee, bordi o punti di interesse come angoli.

4. Detection/segmentation: scelta di quali punti o regioni dell'immagine sono rilevanti per un successivo processing. Esempi possono essere: scelta di un determinato insieme di punti di interesse, segmentazione di una o più regioni dell'immagine contenenti oggetti di interesse.

5. High-level processing: avendo come input una piccola parte dell'immagine, applicare un processing di alto livello come ad esempio il riconoscimento di oggetti o altre task analizzate in precedenza.

6. Decision making: basandosi sui dati raccolti in precedenza, decidere se le domande poste come obiettivo sono state soddisfatte o no. Ad esempio se è avvenuto un riconoscimento facciale corretto o se è stato passata/fallita un'ispezione automatica.

In questa tesi verrà seguita questa sequenza logica di operazioni per costruire un algoritmo di controllo basato sulla visione.

Capitolo 2

Formalizzazione di un task dual arm

In questo paragrafo verrà definita formalmente un'operazione Dual Arm, analizzando le principali differenze con task risolvibili con approccio Single Arm. In seguito verrà data una formalizzazione dal punto di vista geometrico, verranno descritti i parametri scelti per la realizzazione dei nostri esperimenti ed, infine, verrà spiegata la relazione tra la velocità di giunto e la velocità nello spazio operativo.

2.1 Introduzione alle azioni Dual Arm

Con lo sviluppo della robotica collaborativa, si è cercato sempre più di innovare e rafforzare l'interazione tra uomo e macchina. Il processo di integrazione dei manipolatori con il lavoro umano è stato necessario al fine di agevolare gli operatori. Le applicazioni più comuni di robot collaborativi possono essere ritrovate in azioni riguardanti lo spostamento di carichi elevati o nelle operazioni nelle quali è richiesta molta precisione di esecuzione. Per permettere una corretta collaborazione, si cerca di rendere sempre più simile il comportamento di un robot al comportamento di un umano.
Come esempio si può riportare un caso di avvitamento di un tappo su una bottiglia. Nel caso di un essere umano, il metodo più intuitivo prevede un movimento rotatorio del polso. Un robot antropomorfo ridondante ha infinite possibilità di compiere una determinata azione, ma solamente un insieme di queste può essere associabile alle azioni che può compiere un essere umano. A tal fine si cerca di ottimizzare i movimenti di un robot per ritrovare questa somiglianza.
Questo ultimo esempio però non deve confondere il lettore sulla definizione di task risolvibile con approccio Dual Arm. Con approccio Dual Arm si intende un metodo di risoluzione di un problema che prevede l'essenziale utilizzo di due braccia. Diverso è l'approccio Single Arm, nel quale un'azione potrebbe essere risolta con un singolo braccio, ma con tempi maggiori. L'esempio di avvitamento, infatti, viene risolto con approccio Dual Arm, ma potrebbe essere risolto anche con l'utilizzo di un solo braccio.
Per comprendere appieno un'operazione risolvibile solamente con approccio Dual Arm, è

necessario un ulteriore esempio. Si prenda in considerazione lo spostamento di un carico ingombrante da mantenere in una determinata posizione, come ad esempio una pentola piena d'acqua afferrandola da una maniglia laterale. Affrontando il problema con un singolo braccio, non è detto che si riescano a soddisfare tutti i vincoli imposti. Per non versare il contenuto della pentola è necessario soddisfare il vincolo imposto dalla gravità e il vincolo imposto dalla rotazione attorno al punto di presa. A meno di particolari situazioni di lavoro, in cui si operi con forti semplificazioni o con forze elevatissime, questo task risulta irrisolvibile con un singolo braccio. Affrontando il problema con due braccia, invece, molti dei vincoli vengono semplificati automaticamente e l'esecuzione risulta elementare.

2.2 Inserimento e avvitamento

Dopo questa prima introduzione sulla definizione generale di task Dual Arm, verranno descritte in maniera formale le principali operazioni effettuate per la soluzione di questo processo. Nello specifico, verranno descritte l'operazione di inserimento, o Peg-in-Hole, e l'operazione di avvitamento, o Capping.
Le due operazioni, come descritto nel paragrafo precedente, potrebbero essere eseguite anche con un approccio Single Arm. Questo potrebbe avvenire nel caso in cui si operasse con opportuni supporti, in modo tale da mantenere un oggetto in posizione fissa e stabile. Nel caso di risoluzione con approccio Dual Arm, questi supporti non sono richiesti.

2.2.1 Inserimento

Con inserimento, o Peg-in-Hole, si intende un'operazione nella quale si vuole inserire un corpo di una determinata forma in una fessura che lo può contenere. In [14] viene descritto in modo molto ampio e dettagliato tutto ciò che riguarda questa operazione, dai primi approcci di soluzione, alle problematiche riscontrate. In questo capitolo verranno riportati solamente gli aspetti principali dell'operazione.
Nel nostro caso specifico vogliamo inserire lo stantuffo nel corpo della pipetta. Come prima ipotesi semplificativa assumiamo che non ci siano attriti e inceppamenti.
Per eseguire questa operazione sono necessari due passaggi:

1. allineamento dello stantuffo con il corpo;
2. inserimento dello stantuffo nel corpo.

Sotto l'ipotesi semplificativa presentata precedentemente, per eseguire questo compito è sufficiente definire delle terne opportune per gli end effector di ogni braccio. Dopo averle definite, è necessario compiere dei movimenti in modo tale da ottenerne l'allineamento.

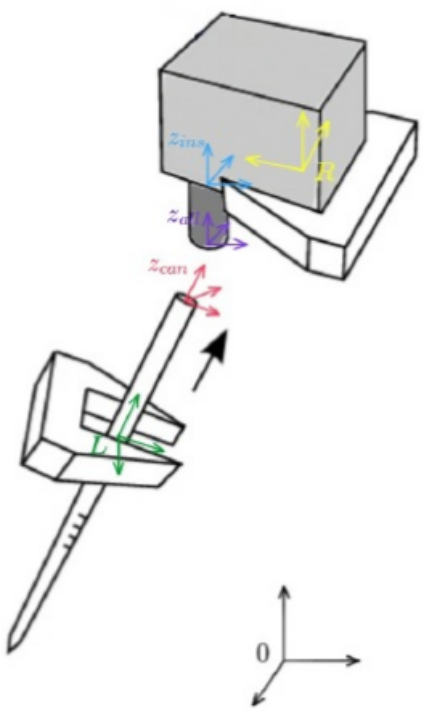

Figura 2.1: Definizione delle terne per ogni end effector per l'operazione di Inserimento.

Dalla Fig. (2.1) si può notare come vengano definite le terne *can* e *all*. Per una corretta esecuzione di questa operazione, è sufficiente allineare gli assi z_{can} e z_{all}.
Questa assunzione è vera a patto che sia fatta un'ulteriore semplificazione: il corpo e la fessura devono essere definibili come cilindri coassiali. In caso contrario si dovrebbe effettuare un allineamento anche degli assi x e y. In questa tesi sia lo stantuffo sia la parte centrale del corpo non presentano sporgenze o asimmetrie lungo la direzione z, quindi possono essere considerati oggetti cilindrici.

2.2.2 Avvitamento

La seconda operazione che sarà definita è l'operazione di avvitamento, o capping. Nel nostro caso verrà avvitato il cappuccio della pipetta sopra il corpo.
Questa operazione è più complicata rispetto all'inserimento, poichè non basta definire due terne corrispondenti agli end effector.

Corpi cilindrici

Verrà prima analizzato il caso in cui i corpi da avvitare siano riportabili a cilindri.

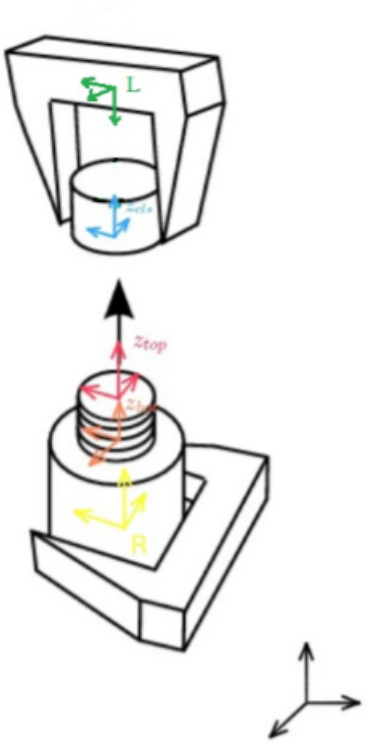

Figura 2.2: Definizione delle terne per ogni end effector per l'operazione di Avvitamento.

Dalla Fig. (2.2), le terne di interesse riportate sono la terna *cls*, la terna *top* e la terna *bot*. Lo scopo dell'avvitamento è fare in modo che la terna *cls*, posizionata alla base del cappuccio, raggiunga la terna *bot*, posizionata alla base del corpo, rappresentante la fine della filettatura di avvitamento.
In questo caso, l'operazione di Avvitamento può essere scomposta in:

1. allineamento del cappuccio con il corpo;
2. avvicinamento lungo la direzione z;
3. rotazione ϕ_z attorno all'asse z.

In un caso generico, i primi due passaggi sono associabili agli stessi passaggi di un'operazione di inserimento, sfruttando le terne *cls* e *top*. Per compiere la terza operazione, la rotazione ϕ_z deve essere congiunta ad un movimento lungo la stessa direzione z, affinchè la terna *cls* raggiunga la terna *bot*.
Come nell'operazione di inserimento, nel caso in cui i corpi siano definibili come cilindri coassiali, la concentrazione ricade maggiormente sulla definizione delle operazioni lungo l'asse z, non prestando particolare attenzione agli assi x e y.

Corpi generici

In questa tesi la semplificazione di assimilare i corpi a cilindri non è applicabile. Il cappuccio della pipetta, infatti, presenta tre fessure, corrispondenti a tre sporgenze presenti sul corpo della stessa pipetta.

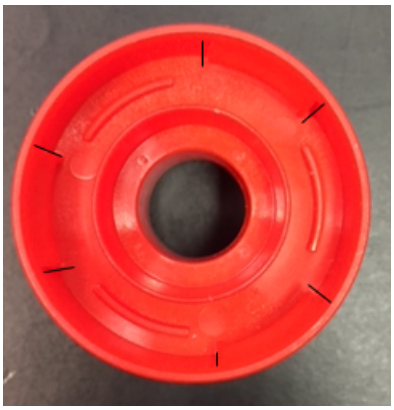

Figura 2.3: Cappuccio della pipetta, con marcatura delle fessure.

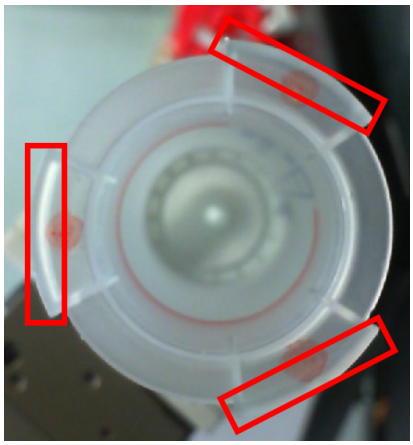

Figura 2.4: Corpo della pipetta, con marcatura delle sporgenze.

Per completare l'operazione sarà necessario allineare questi elementi, per poi procedere con l'avvicinamento lungo l'asse z e successivamente compiere l'avvitamento. La soluzione di questo problema viene trovata nell'inserimento di un controllo di visione nello schema di controllo.
Viceversa, rispetto ad un avvitamento tradizionale, può essere compiuta una semplificazione per quanto riguarda l'operazione di rotazione: è sufficiente una rotazione ϕ_z di pochi gradi per permettere l'incastro tra i due oggetti, senza effettuare alcun movimento lungo la direzione z.
Di conseguenza la terna *top* e la terna *bot* definite in precedenza, coincidono.
Le terne degli oggeti manipolati verranno definite come in Fig (2.5).

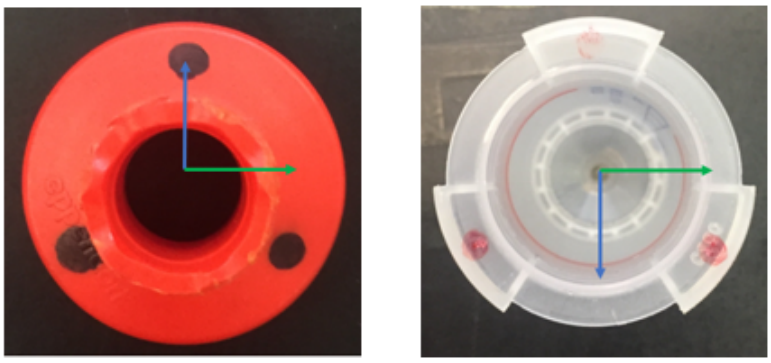

Figura 2.5: Definizione delle terne per il cappuccio e il corpo della pipetta.

In blu viene rappresentato l'asse x, in verde l'asse y. L'asse z viene definito in modo da formare una terna destrorsa.

2.3 Formalizzazione geometrica delle matrici rototraslazionali

In questa sezione verranno formalizzate geometricamente le matrici rototraslazionali relative alle terne degli end effector. Questa sezione è necessaria per dare la giusta definizione geometrica al problema, rendendo più comprensibili le operazioni da svolgere.
Nel caso generale, per descrivere un'operazione Dual Arm sono necessari 12 parametri:

- 6 parametri sono necessari per descrivere posizione e orientamento del braccio destro;
- 6 parametri per definire gli stessi valori, ma per il braccio sinistro.

Come abbiamo descritto nella prima sezione di questo capitolo, ciò che realmente importa nella definizione di un task Dual Arm, non è la posizione assoluta delle braccia. Dobbiamo concentrare la nostra attenzione sulla posizione relativa e sull'orientamento relativo tra le terne degli end effector delle braccia.
Dalla cinematica diretta, a questo scopo, si possono definire inizialmente le terne di posizione e orientamento assoluti per ogni braccio.
Le matrici che contengono queste informazioni sono:

$$A_{Right} = \begin{bmatrix} R_{Right} & p_{Right} \\ 0 & 1 \end{bmatrix}; \qquad A_{Left} = \begin{bmatrix} R_{Left} & p_{Left} \\ 0 & 1 \end{bmatrix} \tag{2.1}$$

R_{Right} e R_{Left} sono matrici di rotazione, p_{Right} e p_{Left} rappresentano la posizione dell'end effector e possono essere scritti come:

$$p_{Right} = \begin{bmatrix} x_{Right} \\ y_{Right} \\ z_{Right} \end{bmatrix}; \qquad p_{Left} = \begin{bmatrix} x_{Left} \\ y_{Left} \\ z_{Left} \end{bmatrix} \tag{2.2}$$

A_{Right} e A_{Left} descrivono posizione e orientamento degli end effector destro e sinistro rispettivamente.
Per ogni oggetto manipolato possiamo descrivere una terna che rappresenti posizione e orientamento rispetto al punto di afferraggio. Sotto l'ipotesi che gli oggetti, così come i gripper, possano essere considerati rigidi, definiamo due matrici, una per braccio.
Queste matrici, R per il braccio destro e L per il braccio sinistro, sono descrivibili come:

$$T_{R_{tool}} = \begin{bmatrix} R_{R_{tool}} & p_{R_{tool}} \\ 0 & 1 \end{bmatrix}; \qquad p_{R_{tool}} = \begin{bmatrix} x_{R_{tool}} \\ y_{R_{tool}} \\ z_{R_{tool}} \end{bmatrix} \tag{2.3}$$

$$T_{L_{tool}} = \begin{bmatrix} R_{L_{tool}} & p_{L_{tool}} \\ 0 & 1 \end{bmatrix}; \qquad p_{L_{tool}} = \begin{bmatrix} x_{L_{tool}} \\ y_{L_{tool}} \\ z_{L_{tool}} \end{bmatrix} \tag{2.4}$$

Le matrici che descrivono la posizione e l'orientamento assoluti degli oggetti afferrati rispetto alla terna base possono essere calcolate. Applicando la regola di composizione di matrici omogenee di trasformazione, otteniamo:

$$A_{R_{tool}} = A_{Right} \cdot T_{R_{tool}} \tag{2.5}$$

$$A_{L_{tool}} = A_{Left} \cdot T_{L_{tool}} \tag{2.6}$$

Poichè il punto di interesse maggiore è la posizione relativa tra le due braccia, dovrà essere costruita una matrice di rototraslazione relativa tra le terne appena definite. Definendo una delle due braccia come leader dell'operazione, nel nostro caso il braccio destro, la matrice di rototraslazione relativa diventa:

$$A_{RL} = A_{R_{tool}} \cdot [A_{L_{tool}}]^{-1} = \begin{bmatrix} R_{RL} & p_{RL} \\ 0 & 1 \end{bmatrix} \tag{2.7}$$

Una volta definita questa matrice, è necessario estrarre le informazioni su posizione e orientamento da essa.
La posizione è ricavabile direttamente dal vettore p_{RL}. L'orientamento è estraibile dalla matrice R_{RL} espressa in termini angoli di Eulero.
In questa tesi è stata scelta la configurazione XYZ, dalla quale è possibile ricavare l'orientamento nel seguente modo:

$$\begin{aligned} \alpha &= atan2(-r_{23}, r_{33}); \\ \beta &= atan2(r_{13}, cos(\alpha \cdot r_{33} - sin(\alpha) \cdot r_{23}); \\ \gamma &= atan2(-r_{12}, r_{11}); \end{aligned}$$

dove r_{ij} rappresenta l'elemento della matrice R_{RL} in posizione (i,j).
Nell'ipotesi che solo un braccio mantenga un oggetto da manipolare e l'altro sia libero, tutto ciò che è stato appena descritto resta comunque valido. L'unico passaggio da

applicare è sostituire la matrice $T_{R_{tool}}$, nel caso in cui il braccio libero sia il braccio destro, o $T_{L_{tool}}$, nel caso in cui il braccio libero sia quello sinistro, con una matrice identità:

$$I_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

Questa formalizzazione geometrica può essere applicata a qualsiasi operazione che utilizzi un robot Dual Arm, con o senza oggetti da manipolare.

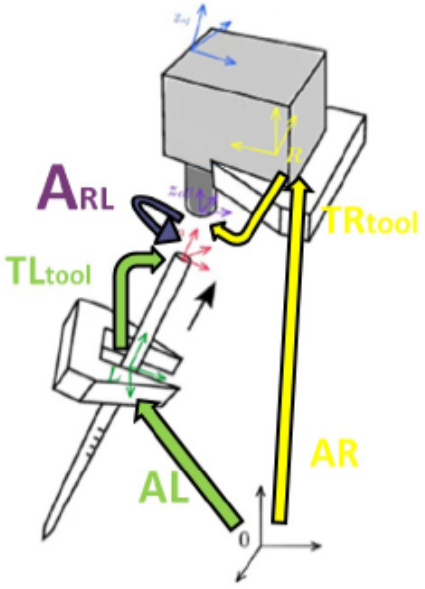

Figura 2.6: Rappresentazione grafica delle matrici dei tool.

2.4 Definizione geometrica delle terne

In questa sezione verrà descritto come procedere alla costruzione delle terne degli end effector, finalizzate a formalizzare un task.
Il metodo più semplice per definire $T_{R_{tool}}$ e $T_{L_{tool}}$ si basa sulla visualizzazione degli angoli di Eulero corrispondenti alla rotazione degli end effector rispetto alla terna base.
Come prima operazione, dobbiamo muovere manualmente le braccia ed assicurarci che gli angoli di Eulero corrispondenti (presi in convenzione XYZ) siano a 0 gradi. Questo ci assicura che le terne degli end effector siano allineate con la terna base (Fig 2.7).

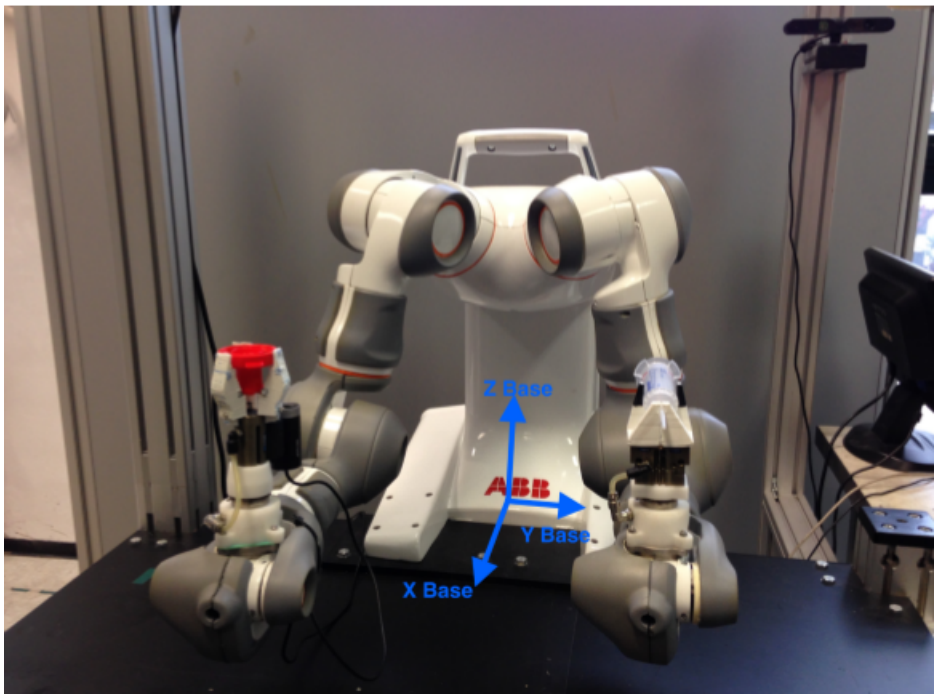

Figura 2.7: Terna base del robot, braccia a 0 gradi.

In questo modo, è possibile definire le terne degli oggetti manipolati con facilità.
In Fig. (2.8) viene riportato un esempio di definizione di terne per oggetto manipolato.

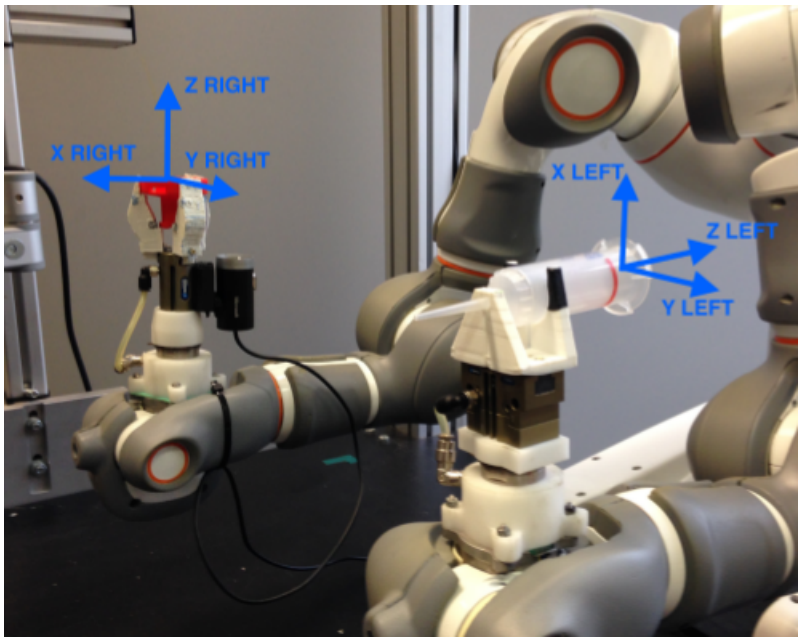

Figura 2.8: Esempio di definizione di terne, per oggetti manipolati.

2.4.1 Braccio destro

Il braccio destro del robot, leader nelle operazioni, è adibito alla manipolazione del cappuccio della pipetta e dello stantuffo. Su di esso è posizionata anche la telecamera utilizzata per il controllo basato sulla visione.
L'orientamento delle terne rappresentanti gli oggetti manipolati dal braccio destro è lo stesso della terna base.
Poiché vengono compiute azioni manipolando oggetti diversi, non è sufficiente definire un'unica terna per questo braccio. Vengono definite 2 terne:

- una terna per la manipolazione dello stantuffo;
- una terna per la manipolazione del cappuccio.

Le terne assumono la seguente forma:

$$T_{R_{tool}} = \begin{bmatrix} 1 & 0 & 0 & x_{R_{tool}} \\ 0 & 1 & 0 & y_{R_{tool}} \\ 0 & 0 & 1 & z_{R_{tool}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Il vettore

$$p_{R_{tool}} = [x_{R_{tool}} \quad y_{R_{tool}} \quad z_{R_{tool}}]^T$$

rappresenta la distanza dell'origine della terna oggetto, rispetto alla terna dell'end effector. Questi valori esprimono distanze espresse in metri.

1. La terna che descrive lo stantuffo della pipetta è rappresentata dal vettore:
   - $p_{R_{tool1}} = [0 \;\; 0 \;\; 0.270]$;
2. La terna che descrive il cappuccio della pipetta è rappresentata dal vettore:
   - $p_{R_{tool2}} = [0 \;\; 0 \;\; 0.160]$.

Si può notare come l'unica differenza risieda nel valore $z_{R_{tool}}$. Questo avviene poichè i gripper sono stati disegnati appositamente per ottenere un effetto di concentricità tra gli oggetti manipolati dal braccio destro.

2.4.2 Braccio sinistro

Il braccio sinistro del robot, in questa tesi, è adibito alla manipolazione del corpo della pipetta.
L'asse z è orientato in modo tale che, durante l'operazione di Inserimento, gli assi z di entrambe le braccia siano allineati. L'asse y mantiene lo stesso orientamento dell'asse y della terna base. Di conseguenza, seguendo la regola della mano destra, troviamo

l'orientamento dell'asse x.
La terna viene espressa come:

$$T_{L_{tool}} = \begin{bmatrix} 0 & 0 & 1 & x_{L_{tool}} \\ 0 & 1 & 0 & y_{L_{tool}} \\ -1 & 0 & 0 & z_{L_{tool}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Il vettore

$$p_{L_{tool}} = [x_{L_{tool}} \;\; y_{L_{tool}} \;\; z_{L_{tool}}]^T$$

come per il braccio destro, rappresenta la distanza dell'origine della terna oggetto rispetto alla terna dell'end effector.
I valori utilizzati, espressi in metri, sono:

- $p_{L_{tool}} = [-0.09 \;\; 0 \;\; 0.123]$.

2.5 Relazione tra velocità di giunto e velocità nello spazio operativo

In questa ultima sezione del secondo capitolo, verrà trattata la relazione tra le velocità di giunto e la velocità nello spazio operazionale. Successivamente verrà ricavata l'espressione dello Jacobiano relativo, necessario alla definizione dei vincoli cinematici.
I vettori velocità degli end effectors rispetto alla terna base sono definiti come:

$$\begin{aligned} v_r &= \begin{bmatrix} J_r & 0 \end{bmatrix} \cdot \dot{q} = \begin{bmatrix} \dot{p}_r^T & \omega_r^T \end{bmatrix} \\ v_l &= \begin{bmatrix} 0 & J_l \end{bmatrix} \cdot \dot{q} = \begin{bmatrix} \dot{p}_l^T & \omega_l^T \end{bmatrix} \end{aligned} \tag{2.9}$$

Le due matrici rappresentano gli Jacobiani degli end effector, i termini ω sono le velocità angolari.
Richiamiamo ora le espressioni delle matrici $A_{R_{tool}}$, $A_{L_{tool}}$ e A_{RL}. Queste matrici rappresentano posizione e orientamento assoluti del braccio destro, del braccio sinistro e posizione relativa tra le due braccia. Le ridefiniamo come:

$$A_{R_{tool}} = \begin{bmatrix} R_r & p_r \\ 0 & 1 \end{bmatrix}, \quad A_{L_{tool}} = \begin{bmatrix} R_l & p_l \\ 0 & 1 \end{bmatrix}, \quad A_{RL} = \begin{bmatrix} R_{rel} & p_{rel} \\ 0 & 1 \end{bmatrix} \tag{2.10}$$

Esprimiamo le seguenti relazioni:

$$\begin{aligned} \dot{p}_{rel} &= R_r^T \cdot \dot{p}_l - R_r^T \cdot \dot{p}_r + p_{rel} \times R_r^T \cdot \omega_r \\ \omega_{rel} &= R_r^T \cdot \omega_l - R_r^T \cdot \omega_r \end{aligned}$$

dalle quali otteniamo lo Jacobiano relativo come:

$$v_{rel} = \begin{bmatrix} R_r^T & 0 \\ 0 & R_r^T \end{bmatrix} \cdot v_l - \begin{bmatrix} R_r^T & 0 \\ 0 & R_r^T \end{bmatrix} \cdot v_r + \begin{bmatrix} 0 & S(P_{rel}) \cdot R_r^T \\ 0 & 0 \end{bmatrix} \cdot v_r \tag{2.11}$$

La matrice $S(p_{rel})$ rappresenta il prodotto vettoriale $p_{rel} \times R_r^T$ in forma matriciale. Sostituendo in (2.11) le espressioni di v_r e v_l, definite in (2.9), otteniamo:

$$v_{rel} = \left\{ \begin{bmatrix} R_r^T & 0 \\ 0 & R_r^T \end{bmatrix} \cdot \begin{bmatrix} -J_r & J_l \end{bmatrix} + \begin{bmatrix} 0 & S(p_{rel}) \cdot R_r^T \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} J_r & 0 \end{bmatrix} \right\} \cdot \dot{q} \tag{2.12}$$

La matrice ottenuta è lo Jacobiano relativo, J_{RL}:

$$v_{rel} = J_{RL} \cdot \dot{q} \tag{2.13}$$

J_{RL} esprime la relazione tra la velocità relativa del braccio sinistro, rispetto al braccio destro, e la velocità dei giunti. Derivando questa espressione:

$$\dot{v}_{rel} = J_{RL} \cdot \ddot{q} + \dot{J}_{RL} \cdot \dot{q} \tag{2.14}$$

ricaviamo la relazione tra le velocità relative nello spazio operazionale e l'accelerazione nello spazio dei giunti.

Capitolo 3

Ottimizzazione vincolata

Lo scopo di questo capitolo riguarda l'esecuzione delle operazioni Dual Arm attraverso l'utilizzo di un algoritmo di ottimizzazione vincolata. L'algoritmo si basa sul controllo della definizione delle variabili di giunto da imporre per soddisfare determinati vincoli. Questo algoritmo è suddiviso in due parti:

1. Ottimizzazione destinata a imporre una certa posa;
2. Ottimizzazione destinata al controllo di forza.

Lo stato dell'arte riguardo l'ottimizzazione vincolata è molto ampio, ma la maggior parte degli algoritmi descritti non soddisfa completamente tutte le richieste. Degno di nota l'algoritmo iTasc (Instantaneous Task Specification using Constraints), descritto in [15] e [16], che permette l'ottimizzazione di vincoli, anche se solo applicati all'attuale stato di moto. [17], [18] e [19] descrivono metodi per ottenere la variabile di controllo ottimizzata, aggiungendo funzioni di costo.
[4] propone un approccio diverso con il quale affrontare il problema di generazione di traiettoria.

3.1 Schema di controllo

Lo schema di controllo utilizzato per fornire le variabili di riferimento al robot a partire da variabili obiettivo nello spazio operativo è rappresentato dal diagramma a blocchi proposto in Fig (3.1).

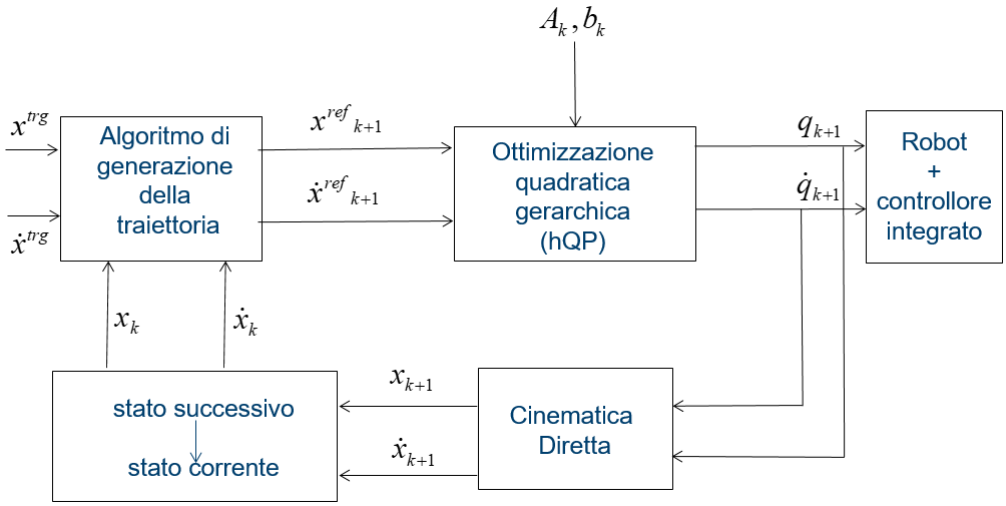

Figura 3.1: Schema a blocchi completo del diagramma di controllo con Ottimizzazione Vincolata.

Il blocco di generazione della traiettoria riceve in input una coppia di valori rappresentanti lo stato desiderato: $(x^{trg}, \dot{x}^{trg})$, una coppia di valori rappresentante lo stato attuale all'istante k: $(x_k, \dot{x}_k)$, e restituisce in output una coppia di valori rappresentanti lo stato da raggiungere al passo successivo k+1: $\left(x^{rif}_{k+1}, \dot{x}^{rif}_{k+1}\right)$. Il percorso generato deve rispettare i vincoli di velocità e accelerazione: $(\dot{x}_{max}, \ddot{x}_{max})$.
L'operazione di generazione di traiettoria può essere effettuata tramite l'utilizzo della libreria *Reflexxes Type II* descritta in [20]. Questa libreria permette la generazione di profili di velocità trapezoidali che verranno usati nell'implementazione dell'algoritmo.
L'output del precedente blocco viene sfruttato come input dall'algoritmo di Ottimizzazione Vincolata. Questo blocco compie un'inversione cinematica e cerca di seguire i vincoli imposti durante l'assegnamento dell'operazione.
L'output di questa funzione esprime la traiettoria di riferimento, espressa in variabili di giunto: $(q_{k+1}, \dot{q}_{k+1})$, che verrà fornita al controllore integrato del robot.

3.2 Vincoli cinematici

In questa sezione verranno descritti i vincoli cinematici nello spazio dei giunti e nello spazio operazionale.

3.2.1 Vincoli cinematici nello spazio dei giunti

Durante l'esecuzione di una generica operazione, può avvenire l'introduzione di vincoli istantanei. Questi vincoli possono essere descritti nella seguente forma:

$$A_k \cdot u_k \leq b_k \tag{3.1}$$

dove A_k e b_k sono quantità tempo-varianti che definiscono l'operazione, mentre u_k è la variabile di controllo. Nel nostro caso u_k rappresenta $\ddot{q}_k$, accelerazione di giunto. Questo implica che tutte le strategie di controllo andranno ad assegnare un valore all'accelerazione, con lo scopo di soddisfare i vincoli cinematici e di forza.
Gli stessi vincoli possono essere descritti in una forma di secondo ordine:

$$lbA \leq A \cdot \ddot{q}_k \leq ubA \tag{3.2}$$

A è una matrice costante, lbA e ubA i valori limite che possono essere assunti, inferiore e superiore rispettivamente. I solutori che verranno descritti in seguito, accettano problemi di ottimizzazione espressi in questa forma.
Il seguente doppio integratore rappresenta, invece, la dinamica del sistema sotto controllo:

$$\begin{cases} q_{k+1} = q_k + T_s \cdot \dot{q}_k + \frac{T_s^2}{2} \cdot \ddot{q}_k \\ \dot{q}_{k+1} = \dot{q}_k + T_s \cdot \ddot{q}_k \end{cases} \tag{3.3}$$

- q_k rappresenta il vettore delle posizioni dei giunti all'istante k;
- $\dot{q}_k$ rappresenta il vettore delle velocità dei giunti all'istante k;
- T_s rappresenta il tempo di campionamento del controllore;
- $\ddot{q}_k$ rappresenta la variabile di controllo.

Possiamo esprimere i vincoli di posizione, velocità e accelerazione da imporre al manipolatore come:

$$\begin{aligned} q_{min} \leq q_k \leq q_{max} \\ \dot{q}_{min} \leq \dot{q}_k \leq \dot{q}_{max} \\ \ddot{q}_{min} \leq \ddot{q}_k \leq \ddot{q}_{max} \end{aligned} \tag{3.4}$$

Si può definire un insieme Q_∞ per descrivere questi vincoli. [21] definisce che Q_∞ è il più grande insieme descritto nello spazio $(q - \dot{q})$ nel quale esiste almeno una soluzione $\ddot{q}$ tale per cui il sistema di controllo rimanga nella stessa regione.

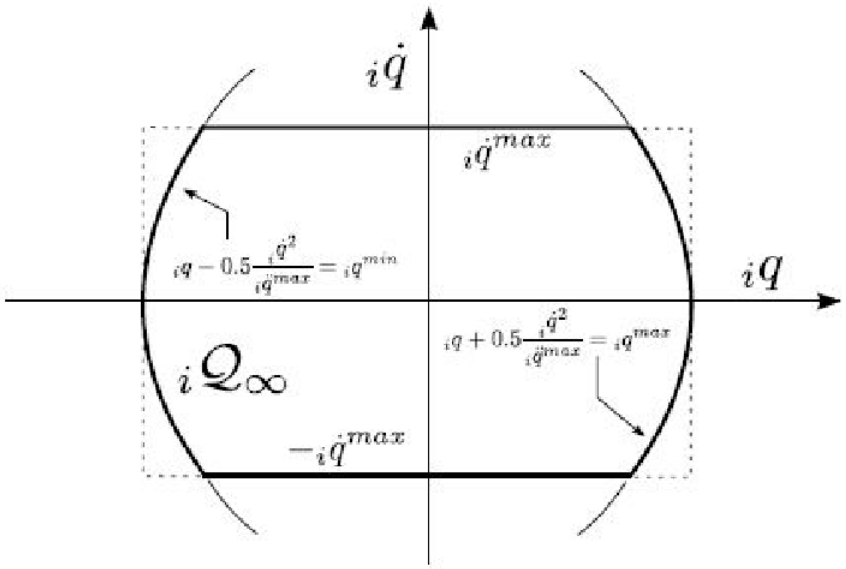

Figura 3.2: Rappresentazione grafica dell'insieme Q_∞ in relazione al giunto i.

I vincoli cinematici possono essere espressi come:

$$\begin{aligned} q &\leq q_{max} - \frac{1}{2} \cdot \frac{\dot{q}^2}{\ddot{q}_{max}} \\ q &\geq q_{min} + \frac{1}{2} \cdot \frac{\dot{q}^2}{\ddot{q}_{max}} \end{aligned} \tag{3.5}$$

Se l'attuale stato di moto e l'accelerazione massima e minima delle variabili di giunto sono conosciute, lo stato di moto al passo successivo apparterrà sempre all'insieme Q_∞. Questo garantisce la stabilità del sistema espresso in (3.3).
Sostituendo le espressioni di (3.3) in (3.5), possiamo esprimere i vincoli nella seguente forma:

$$\begin{aligned} q_{k+1} = q_k + T_s \cdot \dot{q}_k + 0.5 \cdot T_s^2 \cdot \ddot{q}_k &\leq q_{max} - 0.5 \cdot \frac{(\dot{q}_k + T_s \cdot \ddot{q}_k)^2}{\ddot{q}_{max}} \\ q_{k+1} = q_k + T_s \cdot \dot{q}_k + 0.5 \cdot T_s^2 \cdot \ddot{q}_k &\geq q_{min} + 0.5 \cdot \frac{(\dot{q}_k + T_s \cdot \ddot{q}_k)^2}{\ddot{q}_{min}} \\ \dot{q}_{k+1} = q_k + T_s \cdot \ddot{q}_k &\leq \dot{q}_{max} \\ \dot{q}_{k+1} = q_k + T_s \cdot \ddot{q}_k &\geq -\dot{q}_{max} \\ -\ddot{q}_{max} \leq \ddot{q}_k &\leq \ddot{q}_{max} \end{aligned} \tag{3.6}$$

Esprimendo le relazioni rispetto alla variabile di ottimizzazione $\ddot{q}$, otteniamo:

$$\begin{gathered}
\frac{T_s^2}{2 \cdot \ddot{q}_{max}} \cdot \ddot{q}_k^2 + \left(\frac{T_s \cdot \dot{q}_k}{\ddot{q}_{max}} + \frac{T_s^2}{2} \right) \cdot \ddot{q}_k + q_k - q_{max} + T_s \cdot \dot{q}_k + \frac{\dot{q}_k^2}{2 \cdot \ddot{q}_{max}} \leq 0 \\
\frac{T_s^2}{2 \cdot \ddot{q}_{max}} \cdot \ddot{q}_k^2 + \left(\frac{T_s \cdot \dot{q}_k}{\ddot{q}_{max}} - \frac{T_s^2}{2} \right) \cdot \ddot{q}_k - q_k + q_{min} - T_s \cdot \dot{q}_k + \frac{\dot{q}_k^2}{2 \cdot \ddot{q}_{max}} \leq 0 \\
\ddot{q}_k \leq \frac{\dot{q}_{max} - \dot{q}_k}{T_s} = ub_3 \\
\ddot{q}_k \geq -\frac{\dot{q}_{max} + \dot{q}_k}{T_s} = lb_3 \\
lb_4 = -\ddot{q}_{max} \leq \ddot{q}_k \leq \ddot{q}_{max} = ub_4
\end{gathered} \tag{3.7}$$

lb_1 e ub_1 sono ottenuti dalla prima equazione, lb_2 e ub_2 dalla seconda.
Generalmente i vincoli possono essere espressi nella forma:

$$\max_i(lb_i) \leq \ddot{q}_k \leq \min_i(ub_i) \tag{3.8}$$

Soddisfando i vincoli espressi nella seguente forma, abbiamo la certezza che i limiti cinematici non saranno mai violati.

3.2.2 Vincoli cinematici nello spazio operazionale

Dopo aver espresso i vincoli cinematici nello spazio dei giunti, come passo successivo, si possono esprimere i vincoli cinematici nello spazio operazionale. Questi vincoli verranno espressi in termini di posizione x e angolo di Eulero ϕ, come descritto in [22].
Esprimiamo la relazione tra le variabili x e le variabili di giunto q nella seguente forma:

$$\begin{gathered}
x = f(q) \\
\dot{x} = J(q) \cdot \dot{q} \\
\ddot{x} = J(q) \cdot \ddot{q} + \dot{J}(q) \cdot \dot{q}
\end{gathered} \tag{3.9}$$

Il sistema espresso in (3.3) diventa:

$$\begin{cases}
x_{k+1} = x_k + T_s \cdot J_k \cdot \dot{q}_k + 0.5 \cdot T_s^2 \cdot \left(\dot{J}_k \cdot \dot{q}_k + J_k \cdot \ddot{q}_k \right) \\
\dot{x}_{k+1} = J_k \cdot \dot{q} + T_s \cdot \left(\dot{J}_k \cdot \dot{q}_k + J_K \cdot \ddot{q}_k \right)
\end{cases} \tag{3.10}$$

I vincoli possono essere applicati nella forma:

$$\begin{gathered}
-\dot{x}_{max} \leq \dot{x} \leq \dot{x}_{max} \\
-\ddot{x}_{max} \leq \ddot{x} \leq \ddot{x}_{max}
\end{gathered} \tag{3.11}$$

Queste equazioni sono utilizzate insieme a funzioni di costo appropriate, in modo tale da minimizzare l'errore tra i valori attuali di moto $(x_{k+1}, \dot{x}_{k+1})$ e quelli di riferimento $(x_{k+1}^{rif}, \dot{x}_{k+1}^{rif})$, ad ogni passo.
La minimizzazione di una funzione di costo generica può essere espressa come:

$$\min_{\ddot{q}_k}(\| \dot{x}_{k+1} - \dot{x}_{k+1}^{rif} \|_{Q_v}^2 + \| x_{k+1} - x_{k+1}^{rif} \|_{Q_v}^2) \tag{3.12}$$

3.3 Gestione della ridondanza cinematica

Come descritto precedentemente, un robot cinematicamente ridondante ha un numero di gradi di libertà superiore a quelli strettamente necessari alla risoluzione di un problema, di conseguenza può risolvere una medesima operazione in un numero infinito di combinazioni. Questa ridondanza viene introdotta al fine di proporre nuove possibilità di soluzione che garantiscano benefici maggiori. I vincoli di ridondanza vengono introdotti con livello di priorità basso. Come proposto in [23] i vincoli di ridondanza cinematica vengono espressi come problema di minimo nella seguente forma:

$$\min_{\ddot{q}} \parallel \dot{q}_{k+1} \parallel^2 \tag{3.13}$$

La ridondanza cinematica viene utilizzata per ridurre al minimo la velocità dei giunti. La formulazione generale di questo problema viene affrontata in modo differente: l'accelerazione dei giunti viene ricavata dalla cinematica differenziale di secondo ordine e successivamente sostituita nella dinamica del robot, come descritto in [24]. [25] mostra però che questa formulazione può portare ad un comportamento interno instabile del robot.
Viene aggiunto un vincolo di ottimalità, con funzione di costo:

$$\min_{\ddot{q}} \left(\frac{1}{2} \cdot \ddot{q}_k^T \cdot H \cdot \ddot{q}_k + g^T \cdot \ddot{q}_k \right) \tag{3.14}$$

3.4 Vincoli di forza nello spazio operazionale

I vincoli di forza sono relazionati alla forza di contatto sull'end effector con riferimento agli assi x, y e z, rispetto alle variabili di giunto. Per esprimere i vincoli di forza in una forma tale da essere accettata dai solutori, è necessario stabilire la relazione tra la forza e $\ddot{q}$.
La forza è collegata alle variabili geometriche nello spazio operazionale secondo il modello elastico:

$$f = K_{env} \cdot (z - z_0) = K_{env} \cdot \Delta z \tag{3.15}$$

K_{env} rappresenta la rigidezza dell'ambiente, parametro da considerare incerto. Di conseguenza deve essere contenuto in un intervallo di valori: deve essere conosciuto l'ambiente nel quale si vuole operare per poter stabilire questo intervallo.

$$K_{min} \leq K_{env} \leq K_{max}$$

Δz rappresenta la posizione relativa tra i due oggetti manipolati, espressa nel punto di contatto lungo l'asse z.
I vincoli di forza possono essere espressi come:

$$f_{min} \leq f_k \leq f_{max} \tag{3.16}$$

Questo implica che la forza non deve seguire un riferimento come in un controllo classico [26], ma deve essere contenuta in un range di valori. La dimensione di questo intervallo dipende sia dalla velocità di esecuzione dell'operazione, sia dai possibili valori di K_{env}.
Le derivate della forza possono essere espresse come:

$$\begin{aligned} f &= K_{env} \cdot (z - z_0) \\ \dot{f} &= K_{env} \cdot \dot{z} \\ \ddot{f} &= K_{env} \cdot \ddot{z} \end{aligned} \tag{3.17}$$

Usando la relazione (3.9) si ottiene:

$$\begin{aligned} \dot{f} &= K_{env} \cdot J_A \cdot \dot{q} \\ \ddot{f} &= K_{env} \cdot (J_A \cdot \ddot{q} + \dot{J}_A \cdot \dot{q}) \end{aligned} \tag{3.18}$$

J_A indica lo Jacobiano analitico.
Questo implica che il sistema controllato può essere rappresentato in tempo discreto come:

$$\begin{cases} f_{k+1} = f_k + T_s \cdot \dot{f}_k + \frac{T_s^2}{2} \cdot \ddot{f}_k \\ \dot{f}_{k+1} = \dot{f}_k + T_s \cdot \ddot{f}_k \end{cases} \tag{3.19}$$

Sostituendo (3.18) nel sistema e dividendo per il parametro K_{env} si ottiene:

$$\begin{cases} f_{k+1} = f_k + T_s \cdot K_{env} \cdot J_A \cdot \dot{q} + \frac{T_s^2}{2} \cdot K_{env} \cdot \left(J_A \cdot \ddot{q} + \dot{J}_A \cdot \dot{q} \right) \\ J_A \cdot \dot{q} = J_A \cdot \dot{q} + T_s \cdot \left(J_A \cdot \ddot{q} + \dot{J}_A \cdot \dot{q} \right) \end{cases} \tag{3.20}$$

Poichè il controllo di forza è applicato lungo l'asse z, il sistema diventa:

$$\begin{cases} f_{k+1} = f_k + T_s \cdot K_{env} \cdot \dot{z}_k + \frac{T_s^2}{2} \cdot K_{env} \cdot \ddot{z}_k \\ \dot{z}_{k+1} = \dot{z}_k + T_s \cdot \ddot{z}_k \end{cases} \tag{3.21}$$

3.5 Solutori e Implementazione in Matlab

In questa ultima sezione riguardante l'ottimizzazione, verrano descritti i due solutori utilizzati in Matlab: *qpOASES*, per la fase di simulazione, e *SOTH*, utilizzato nella fase di sperimentazione real time. Questa distinzione viene effettuata poichè *qpOASES* richiede un carico computazionale molto superiore a rispetto a *SOTH*, rendendo quest'ultimo più adatto per gli esperimenti in tempo reale.

3.5.1 qpOASES

qpOASES [27] è un solutore open source, implementazione di Online Active Set Strategy [28]. L'idea di questo solutore nasce dall'osservazione che l'insieme ottimale di vincoli attivi non cambia tra un problema quadratico all'altro. La principale ragione di sviluppo

di *qpOASES* riguarda la soluzione di problemi nel contesto del Model Predictive Control, MPC.
qpOASES risolve i problemi nella forma:

$$\min_x \left(\frac{1}{2} x^T H x + x^T g \right) \tag{3.22}$$

Il problema di ottimizzazione è soggetto a vincoli del tipo:

$$lbA \leq Ax \leq ubA \tag{3.23}$$

H è la matrice Hessiana, g il vettore gradiente, lbA e ubA sono i limiti superiore e inferiore da soddisfare per ottenere l'ottimalità della variabile di controllo x.

Soluzione dei vincoli cinematici

Per risolvere i vincoli cinematici, bisogna riportare il problema di ottimizzazione (3.12) nelle forme appena descritte: (3.22) e (3.23), utilizzando $\ddot{q}$ come variabile di controllo.
Verranno analizzate separatamente le due sezioni della funzione di costo (3.12).
Consideriamo il primo termine:

$$\min_{\ddot{q}} \parallel \dot{x}_{k+1} - \dot{x}_{k+1}^{rif} \parallel_{Qv}^2$$

Sostituendo la relazione tra variabili di giunto e variabili nello spazio operazionale, otteniamo:

$$\left[T_s^2 \ddot{q}_k (J_k^T J_k) \ddot{q}_k + 2 T_s J_k \ddot{q}_k^T (T_s \dot{J}_k \dot{q}_k + \dot{x}_k - \dot{x}_{k+1}^{rif})^T + (...)^2 \right]_{Qv}$$

dove $(...)$ indica i termini infinitesimali, omessi per evitare inutili complicazioni. Dividendo il tutto per $\frac{1}{2T_s^2}$ otteniamo:

$$\left[\frac{1}{2} \ddot{q}_k^T (J_k^T J_k) \ddot{q}_k + \ddot{q}_k^T \left(\frac{1}{T_s} J_k (T_s \dot{J}_k \dot{q}_k + \dot{x}_k - \dot{x}_{k+1}^{rif}) \right) \right] \cdot Q_v \tag{3.24}$$

Da questa equazione otteniamo:

$$H_1 = J_k^T Q_v J_k$$

$$g_1 = \frac{1}{T_s} J_k (T_s \dot{J}_k \dot{q}_k + \dot{x}_k - \dot{x}_{k+1}^{rif})^T Q_v$$

Allo stesso modo possiamo operare sulla seconda parte della funzione di costo:

$$\min_{\ddot{q}} \parallel x_{k+1} - x_{k+1}^{rif} \parallel_{Qp}^2$$

Sostituendo la relazione tra le variabili di giunto e le variabili nello spazio operazionale, otteniamo:

$$\left[\frac{T_s^4}{4} \ddot{q}_k^T (J_k^T J_k) \ddot{q}_k + T_s^2 \ddot{q}_k^T J_k^T a_0^T + (...)^2 \right] \cdot Q_p$$

con $a_0 = x_k + T_s\dot{x}_k + \frac{T_s^2}{2}\dot{J}_k\dot{q}_k - x_{k+1}^{rif}$.
Dividendo l'equazione per $\frac{1}{2T_s^2}$ ricaviamo:

$$\left[\frac{1}{2}\ddot{q}_k^T(\frac{T_s^4}{4}J_k^T J_k)\ddot{q}_k + \ddot{q}_k^T(\frac{1}{2}J_k^T a_0^T)\right] \cdot Q_p \tag{3.25}$$

dalla quale otteniamo:

$$H_2 = \frac{T_s^2}{4}J_k^T Q_p J_k$$
$$g_2 = \frac{1}{2}J_k^T a_0^T Q_p$$

Sommando $H_1 + H_2$, $g_1 + g_2$ otteniamo la matrice Hessiana H e il vettore gradiente g utilizzati per minimizzare il problema precedentemente espresso.
I vincoli cinematici vengono disposti in una forma accettata dal risolutore, diventando:

$$\begin{gathered} -\dot{x}_{max} \leq \dot{x}_{k+1} \leq \dot{x}_{max} \\ -\dot{x}_{max} \leq \dot{x}_k + T_s\dot{J}_k\dot{q}_k + T_s J_k\ddot{q}_k \leq \dot{x}_{max} \\ -\dot{x}_{max} - T_s\dot{J}_k\dot{q}_k - \dot{x}_k \leq T_s J_k\ddot{q}_k \leq \dot{x}_{max} - T_s\dot{J}_k\dot{q}_k - \dot{x}_k \end{gathered} \tag{3.26}$$

In accordo con (3.23) possiamo ottenere facilmente:

$$\begin{gathered} lbA = -\dot{x}_{max} - T_s\dot{J}_k\dot{q}_k - \dot{x}_k \\ A = T_s J_k \\ ubA = \dot{x}_{max} - T_s\dot{J}_k\dot{q}_k - \dot{x}_k \end{gathered} \tag{3.27}$$

La velocità dei giunti è utilizzata per minimizzare la ridondanza cinematica:

$$\min_{\ddot{q}} \parallel \dot{q}_{k+1} \parallel^2 \tag{3.28}$$

Se sostituiamo l'espressione del sistema (3.3) in questa espressione otteniamo:

$$\begin{gathered} \min_{\ddot{q}} \parallel \dot{q}_k + T_s\ddot{q}_k \parallel^2 \\ \min_{\ddot{q}} \left(\frac{1}{2}\ddot{q}_k^T I\ddot{q}_k + \frac{1}{T_s}\dot{q}_k^T\ddot{q}_k\right) \end{gathered} \tag{3.29}$$

dalla quale possiamo ricavare:

$$\begin{gathered} H = I \\ g = \frac{1}{T_s}\dot{q}_k \end{gathered} \tag{3.30}$$

Infine, traduciamo in una forma accettabile dal solutore i vincoli sulle accelerazioni, ai quali vogliamo imporre la seguente relazione:

$$\ddot{x} = \ddot{x}_{qpp1}^{opt} \tag{3.31}$$

dove $\ddot{x}_{qpp1}^{opt}$ è la variabile di controllo ottima, ottenuta risolvendo i precedenti problemi di ottimizzazione. *qpOASES* viene chiamata una seconda volta per ottenere la variabile di controllo ottimale finale.
Imponiamo $lbA = ubA$, ottenendo $J_k\ddot{q}_{pp1}^{opt} \leq J_k\ddot{q}_k \leq J_k\ddot{q}_{pp1}^{opt}$, che conduce a
$lbA = ubA = J_k\ddot{q}_{pp1}^{opt}$

Soluzione dei vincoli di forza

qpOASES e l'algoritmo di generazione di traiettoria hanno bisogno di vincoli di forza definiti a livello di accelerazione $\ddot{q}$.
Per eseguire le nostre operazioni, abbiamo bisogno di un controllo di forza tale per cui sia mantenuto sempre un contatto tra gli oggetti interagenti. Questo vincolo può essere espresso come:

$$f_{min} \leq f_k \leq f_{max} \tag{3.32}$$

Per ottenere la forma richiesta da *qpOASES* abbiamo bisogno della relazione tra spazio operazionale e spazio dei giunti, avendo come riferimento solo l'asse z:

$$\begin{aligned} z &= f(q) \\ \dot{z} &= J_z \cdot \dot{q} \\ \ddot{z} &= J_z \cdot \ddot{q} + \dot{J}_z \cdot \dot{q} \end{aligned} \tag{3.33}$$

J_z è lo jacobiano relativo lungo l'asse z, composto da 6 righe e 14 colonne che descrivono la posizione e l'orientamento del braccio destro e del braccio sinistro rispettivamente. A questo punto possiamo esprimere il vincolo espresso in (3.32) nella forma:

$$\ddot{z}_{min} \leq \ddot{z} \leq \ddot{z}_{max} \tag{3.34}$$

I valori di z_{min} e z_{max} sono due valori per cui la forza f è sempre contenuta tra f_{min} e f_{max}. Dobbiamo considerare anche tutte le incertezze che possono comparire: rigidezza dell'ambiente K_{env}, disturbi nelle condizioni iniziali di f e $\dot{z}$, incertezza sulla superficie di contatto z_0.
Considerando (3.34) e sostituendola nell'espressione (3.33), esprimiamo la variabile di ottimizzazione $\ddot{q}$ come segue:

$$\ddot{z}_{min} - \dot{J}_z \cdot \dot{q} \leq J_z \cdot \ddot{q} \leq \ddot{z}_{max} - \dot{J}_z \cdot \dot{q} \tag{3.35}$$

Per concludere, dobbiamo esprimere i vincoli (3.32) nella forma richiesta da *qpOASES*, ottenendo:

$$\begin{aligned} A_{force} &= J_z \\ lbA_{force} &= \ddot{z}_{min} - \dot{J}_z \cdot \dot{q} \\ ubA_{force} &= \ddot{z}_{max} - \dot{J}_z \cdot \dot{q} \end{aligned} \tag{3.36}$$

Esprimendo questi vincoli, siamo sicuri che gli oggetti manipolati rimarranno a contatto finchè l'operazione sarà conclusa.

3.5.2 SOTH

qpOASES risolve i problemi di ottimizzazione, ma il suo tempo computazionale è troppo elevato per ottenere i valori ottimi desiderati entro i limiti del real time. Di conseguenza, viene utilizzato il solutore *SOTH* (Stack of Task Hierarchical Solver), descritto in [29].

$SOTH$ è un solutore basato sulla soluzione in ordine gerarchico di più problemi di ottimizzazione: i problemi di ottimizzazione vengono suddivisi in blocchi e vengono risolti in ordine di priorità decrescente. Se il problema non può essere risolto soddisfacendo tutti i vincoli, $SOTH$ genera una soluzione che soddisfa solo i vincoli a più alta priorità; i vincoli rimanenti vengono rilassati e la soluzione viene ottimizzata di conseguenza.
Il problema di ottimizzazione viene espresso nella forma:

$$\begin{aligned} lA(1) \leq A(1)x \leq uA(1) \\ A(2)x = uA(2) \\ \vdots \\ A(n-1)x \geq lA(n-1) \\ A(n)x = uA(n) \end{aligned} \tag{3.37}$$

x, nel nostro caso, rappresenta $\ddot{q}$, A è una generica matrice di coefficienti di x, uA e lA sono i limiti del problema di ottimizzazione. Ogni riga di (3.37) rappresenta un insieme di equazioni e disequazioni, dipendenti dal valore della variabile da ottimizzare.
Il problema, a questo punto, viene diviso in gruppi con diverse priorità. I problemi con priorità maggiore verranno ottimizzati prima, mentre quelli con priorità minore verranno ottimizzati rispetto al null-space delle soluzioni delle ottimizzazioni precedenti.
Viene ora proposto un esempio in cui viene data diversa priorità ai vincoli.

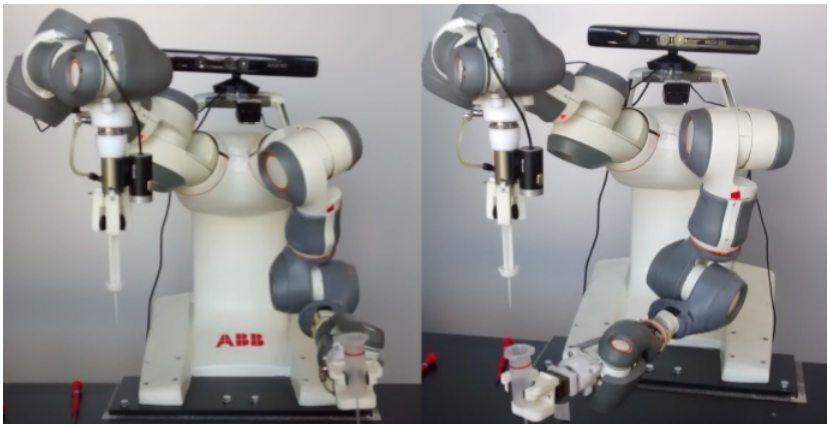

Figura 3.3: Comportamento del robot con vincoli del braccio destro a priorità maggiore.

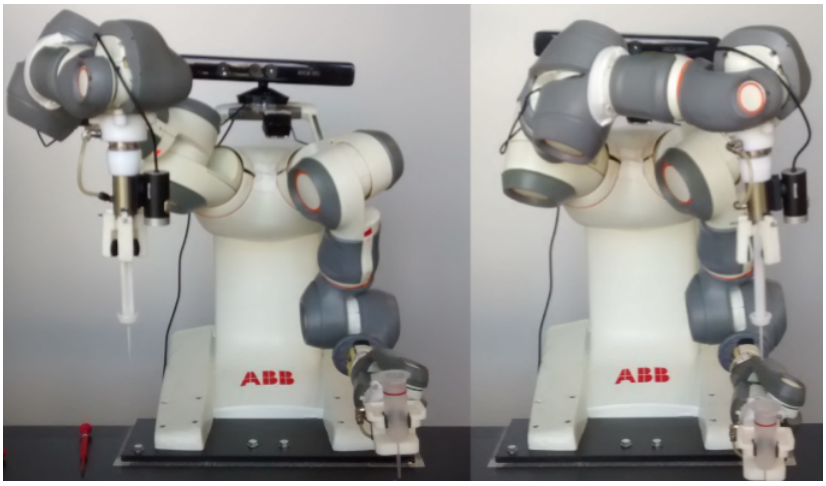

Figura 3.4: Comportamento del robot con vincoli del braccio sinistro a priorità maggiore.

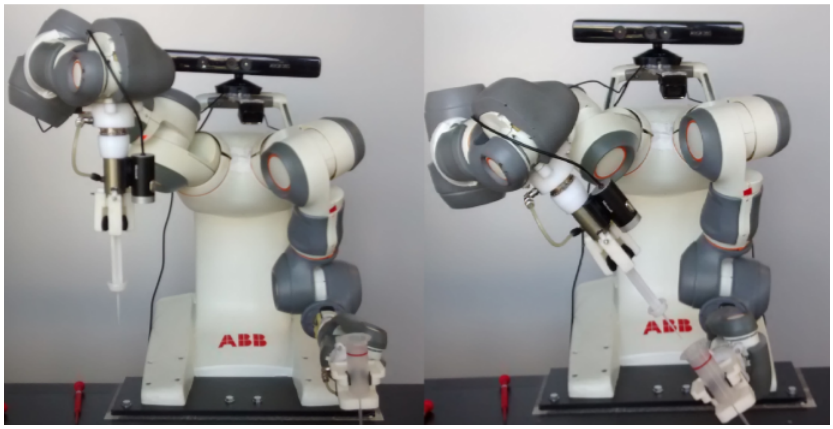

Figura 3.5: Comportamento del robot con minimizzazione delle velocità dei giunti nello spazio nullo.

Soluzione dei vincoli cinematici

In questa sezione verrà descritto il problema di ottimizzazione relativo ai vincoli cinematici, tradotto nella forma accettata da $SOTH$. I vincoli verranno descritti in ordine di priorità decrescente.

1. Vincoli sull'accelerazione dei giunti:

$$\begin{aligned} \ddot{q}_k^{inf} \leq \ddot{q}_k \leq \ddot{q}_k^{sup} \\ lA(1) = \ddot{q}_k^{inf} \\ A(1) = I \\ uA(1) = \ddot{q}_k^{sup} \end{aligned} \tag{3.38}$$

2. Vincoli sulla velocità dei giunti:

$$\begin{gathered} -\dot{x}^{max} - \Delta J_k \dot{q}_k - \dot{x}_k \leq T_s J_k \ddot{q}_k \leq \dot{x}^{max} - \Delta J_k \dot{q}_k - \dot{x}_k \\ lA(2) = -\dot{x}^{max} - \Delta J_k \dot{q}_k - \dot{x}_k \\ A(2) = T_s J_k \\ uA(2) = \dot{x}^{max} - \Delta J_k \dot{q}_k - \dot{x}_k \end{gathered} \tag{3.39}$$

3. Minimizzazione dei riferimenti di posizione e velocità:

$$\min_{\ddot{q}_k} \left(\| \dot{x}_{k+1} - \dot{x}_{k+1}^{rif} \|_{Q_v}^2 \right) + \min_{\ddot{q}_k} \left(\| x_{k+1} - x_{k+1}^{rif} \|_{Q_p}^2 \right) \tag{3.40}$$

Il vincolo riguardante la minimizzazione della velocità può essere espresso come:

$$\sqrt{Q_v} \dot{x}_{k+1} = \sqrt{Q_v} \dot{x}_{k+1}^{rif}$$

il quale, sostituendo il valore di $\dot{x}_{k+1}$, diventa:

$$\begin{gathered} \sqrt{Q_v} \left(\dot{x}_k + T_s \dot{J}_k \dot{q}_k + T_s J_k \ddot{q}_k \right) = \sqrt{Q_v} \left(\dot{x}_{k+1}^{rif} \right) \\ A(3) = \sqrt{Q_v} T_s J_k \\ lA(3) = uA(3) = \sqrt{Q_v} \left(-\dot{x}_k - T_s \dot{J}_k \dot{q}_k + \dot{x}_{k+1}^{rif} \right) \end{gathered} \tag{3.41}$$

Il vincolo riguardante la minimizzazione della posizione può essere espresso come:

$$\sqrt{Q_p} x_{k+1} = \sqrt{Q_p} x_{k+1}^{rif}$$

il quale, sostituendo il valore di x_{k+1}, diventa:

$$\begin{gathered} \sqrt{Q_p} \left(x_k + T_s \dot{x}_k + \tfrac{T_s^2}{2} \dot{J}_k \dot{q}_k + \tfrac{T_s^2}{2} J_k \ddot{q}_k \right) = \sqrt{Q_p} \left(x_{k+1}^{rif} \right) \\ A(4) = \sqrt{Q_p} T_s^2 J_k \ddot{q}_k \\ lA(4) = uA(4) = \sqrt{Q_p} \left(x_{k+1}^{rif} - x_k - T_s \dot{x}_k - \tfrac{T_s^2}{2} \dot{J}_k \dot{q}_k \right) \end{gathered} \tag{3.42}$$

4. Gestione della ridondanza cinematica:

$$\min_{\ddot{q}_k} \left(\| N^T \dot{q}_{k+1} \| \right) \tag{3.43}$$

SOTH minimizza le velocità nel null-space, insieme rappresentante lo spazio delle velocità di giunto che non producono velocità di esecuzione dell'operazione. N è una matrice che proietta qualsiasi velocità di giunto nel null-space: vengono generati moti interni che non modificano la posizione dell'end effector.
La minimizzazione diventa:

$$\begin{gathered} T_s N^T \ddot{q}_k = -N^T \dot{q}_k \\ A(5) = T_s N^T \\ uA(5) = lA(5) = -N^T \dot{q}_k \end{gathered} \tag{3.44}$$

La gerarchia è quindi composta da 4 livelli di priorità. Il manipolatore avrà quindi:

- 14 vincoli con priorità 1: accelerazione dei giunti di entrambe le braccia;
- 6 vincoli con priorità 2: velocità nello spazio operazionale;
- 12 vincoli con priorità 3: 6 per la minimizzazione dell'errore di posizione, 6 per la minimizzazione dell'errore di velocità;
- 8 vincoli con priorità 4: gestione della ridondanza.

Con priorità 1 si intende il livello di priorità maggiore.
Questa soluzione prevede un calcolo computazionale troppo elevato, quindi il livello di priorità 1 viene diviso in due livelli di priorità differente da 7 vincoli ciascuno. In questo modo un braccio avrà priorità maggiore rispetto all'altro.

Soluzione dei vincoli di forza

L'ottimizzazione dei vincoli di forza avviene nelle medesime modalità viste nell'implementazione tramite *qpOASES*.
L'ottimizzazione di forze risulta fondamentale nell'operazione di avvitamento e, di conseguenza, deve avere un livello di priorità elevato.
Il livello di priorità assegnato all'ottimizzazione è inferiore solamente al livello di priorità assegnato all'ottimizzazione delle accelerazioni di giunto, quindi sarà il livello 2.

Capitolo 4

Computer vision

Questo capitolo verrà dedicato alla Computer Vision, ramo scientifico dedicato all'applicazione di tecniche visive e all'interpretazione dell'immagine. Inizialmente verranno analizzate le principali funzioni utilizzate per la creazione degli algoritmi di controllo dedicati alla generazione di traiettoria. Successivamente verrà descritto l'approccio utilizzato per integrare la visione nello schema di controllo. Verranno analizzate gli ambienti software utilizzati per l'implementazione. Infine verranno analizzati i casi critici di applicazione e i limiti dell'algoritmo.

4.1 Image Analysis

In questa sezione verranno descritte le due principali funzioni utilizzate per il riconoscimento delle caratteristiche all'interno dell'immagine. Nello specifico, verranno utilizzate delle funzioni per andare a riconoscere dei segnalatori posti sul cappuccio e sul corpo della pipetta (Fig 4.11 e Fig. 4.12).
Inizialmente verrà descritta in maniera formale un'immagine digitale, con attenzione ai diversi livelli di descrizione dipendenti dal sensore d'immagine utilizzato. Successivamente verrà descritto il filtro di Canny, utilizzato per un primo processamento dell'immagine e finalizzato al riconoscimento dei bordi di oggetti. Infine verrà descritta la trasformata di Hough, processing di alto livello destinato al riconoscimento di forme (nello specifico circonferenze) all'interno dell'immagine.
Per la descrizione di queste funzioni si farà riferimento principalmente a [30].

4.1.1 Immagine digitale

Un'immagine digitale è una rappresentazione numerica di un'immagine bidimensionale e può essere principalmente di due tipologie: vettoriale o bitmap.
Un'immagine vettoriale è una rappresentazione poligonale di immagini nella Computer Graphic. In questa tipologia di rappresentazione vengono utilizzati vettori che collegano punti di interesse, rappresentati i loro estremi.

La definizione più classica di immagine digitale, però, viene rispecchiata dalla bitmap. Con bitmap si intende una rappresentazione matriciale dell'immagine: un'immagine viene descritta attraverso una griglia di pixel o punti di colore (Fig. 4.1).

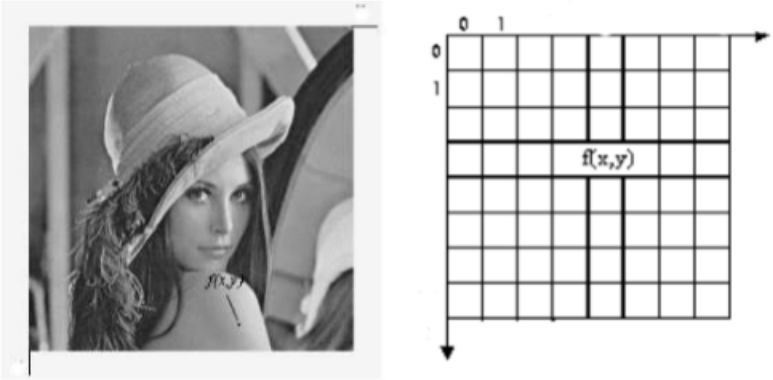

Figura 4.1: Rappresentazione di un'immagine digitale.

Un pixel è una regione rettangolare coincidente con una cella della griglia di campionamento e il valore ad esso associato è l'intensità media nella cella. Dal campionamento si determina la risoluzione spaziale: viene determinato il numero di pixel da usare lungo gli assi x e y, maggiore è la risoluzione, maggiore è il numero di dettagli visibili.
A seconda dell'immagine da analizzare, cambiano i valori associati ad un pixel: nelle immagini a colori, ad esempio, a ogni elemento della matrice corrisponde una terna ordinata di interi (R, G, B), rappresentanti rispettivamente le intensità di rosso, verde e blu. Possono assumere valori compresi tra 0 e 255 a seconda del livello di intensità della sorgente di colore. In questo modo un'immagine può essere vista come una terna di matrici rappresentanti i colori fondamentali (Fig. 4.2).

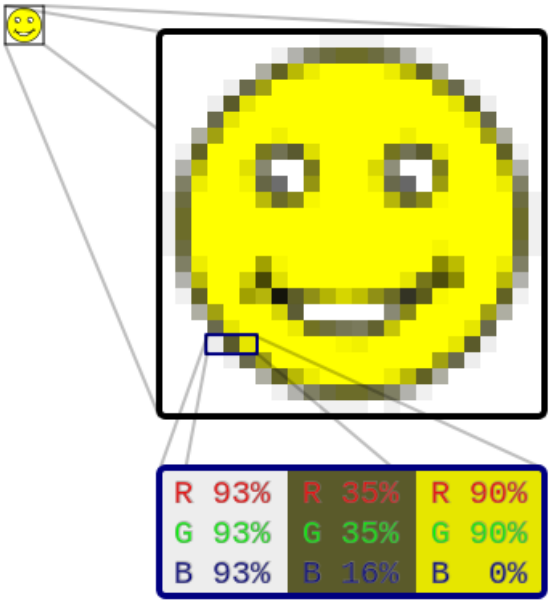

Figura 4.2: Rappresentazione dei pixel di un'immagine colorata.

Un'altra possibile rappresentazione di un'immagine colorata è attraverso la terna di colori complementari alla RGB, ovvero la terna CMY (ciano, magenta, giallo). La terna RGB viene utilizzata nei dispositivi di visualizzazione, la terna CMY nei dispositivi di stampa.

4.1.2 Filtro di Canny

Il primo passo per il riconoscimento dei segnalatori posizionati sul corpo e sul cappuccio della pipetta consiste nell'evidenziare tutti i possibili bordi all'interno di un'immagine. Per effettuare questo processo, viene applicato un filtro all'immagine basato sull'operatore di Canny.

Filtraggio

Un filtro spaziale, detto anche maschera o kernel, è una regione rettangolare caratterizzata da un'operazione predefinita, e ampiamente utilizzato nelle operazioni di miglioramento delle qualità delle immagini. L'operazione viene eseguita sui pixel dell'immagine corrispondenti alla regione e le modifiche indotte dal filtro generano un'immagine filtrata. Il filtro opera sovrapponendosi all'immagine e viene applicato tramite un procedimento di convoluzione:

$$g(x,y) = \sum_{s=-a}^{s=a} \sum_{t=-b}^{t=b} w(s,t) f(x+s, y+t) \tag{4.1}$$

$f(x,y)$ è l'immagine originale di dimensioni $n \times n$, $w(s,t)$ è il filtro di convoluzione $m \times m$, $g(x,y)$ è l'immagine di output.
Le tipologie di filtro più conosciute sono:

- filtri di smoothing: generalmente sono utilizzati per sfocare le immagini e l'eventuale presenza di rumore.
- filtri derivativi: utilizzati principalmente nelle operazioni di sharpening e di edge detection.

Operatore di Canny

L'operatore di Canny è un filtro utilizzato nel campo dell'elaborazione delle immagini come riconoscitore di bordi. Il metodo si propone di ottenere:

- Buon riconoscimento: individuare e marcare il maggior numero possibile di bordi;
- Buona localizzazione: posizionare un edge identificato il più vicino possibile ad un edge reale;
- Risposta minima: marcare un bordo solamente una volta.

L'applicazione di questo filtro si articola in quattro passaggi:

1. Image smoothing: smoothing di un'immagine tramite l'applicazione di un filtro gaussiano. Filtri piccoli producono una minore sfocatura e consentono di riconoscere dettagli più fini, viceversa filtri grandi portano ad una grande sfocatura con conseguente riconoscimento di dettagli più grandi.

2. Differenziazione: Calcolo del gradiente dell'immagine ottenuta dal filtraggio, seguito dal calcolo della magnitudo e dell'angolo del vettore gradiente.

3. Soppressione dei non massimi: Individuazione della direzione del gradiente per ogni punto e confronto con il modulo dei punti vicini, nella direzione del gradiente stesso. Se almeno uno dei punti ha modulo maggiore del pixel esaminato, questo viene soppresso (modulo posto a zero).

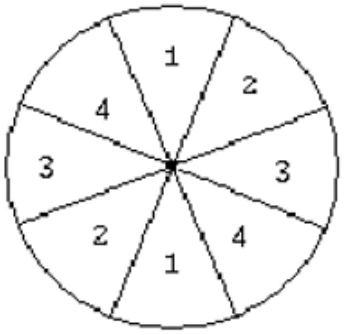

Figura 4.3: Intervalli delle normali agli edge in un intorno 3x3.

   Ad esempio guardando la figura (4.3), se il vettore ha valore angolare compreso nel settore 1 si avrà un edge orizzontale.

4. Thresholding: sogliatura dell'immagine ottenuta al passo precedente. Si definiscono due soglie T_L e T_H che vengono confrontate con i punti dell'immagine. Se il valore è inferiore a T_L viene scartato, se è superiore a T_H viene accettato, se è compreso tra le due viene accettato solamente se contiguo ad un punto precedentemente accettato. L'uso di due soglie permette il riconoscimento di falsi edge con più certezza rispetto all'utilizzo di una singola soglia T. Infatti, se T fosse troppo bassa si potrebbero identificare falsi edge (nello specifico un disturbo potrebbe essere considerato come importante), se T fosse troppo elevata punti validi potrebbero essere eliminati portando alla perdita di informazioni. Riassumendo, T_H identifica strutture significative dell'immagine, mentre T_L le connette.

Viene ora presentato un esempio di applicazione dell'operatore di Canny, con diversi livelli di soglie.

Figura 4.4: Esempio di immagine non filtrata.

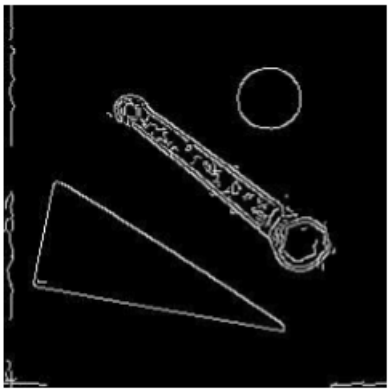

Figura 4.5: Applicazione filtro di Canny con soglie: $T_L = 30$ e $T_H = 70$.

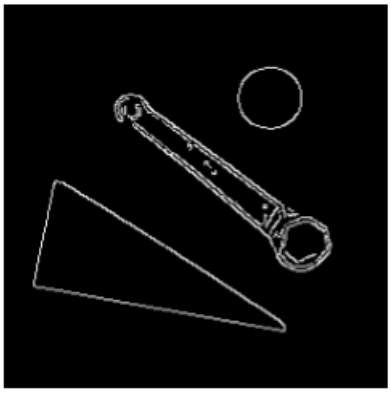

Figura 4.6: Applicazione filtro di Canny con soglie: $T_L = 60$ e $T_H = 120$.

Si può notare come nel secondo esempio, applicando diverse soglie, i contorni vengano riconosciuti in modo migliore rispetto al primo caso.

4.1.3 Trasformata di Hough

Dopo aver eseguito il primo processing, otteniamo un'immagine in cui sono presenti solamente i bordi degli oggetti presenti all'interno dell'immagine. A questo punto viene

utilizzata una seconda funzione, basata sulla Trasformata di Hough (HT), che permette il riconoscimento vero e proprio dei marker presenti sul corpo e sul cappuccio della pipetta.
La Trasformata di Hough permette di stabilire se particolari insiemi di pixel si trovano su una specifica curva oppure no. In generale la HT permette il riconoscimento di configurazioni globali come segmenti, curve e forme prestabilite, trasformando i punti dello spazio immagine in punti dello spazio dei parametri. La HT si basa sulla validazione di ipotesi: definita la curva che si vuole cercare nella scena, per ogni punto si calcolano i parametri delle curve che potrebbero passare da esso. Si ottiene una funzione di accumulazione definita nello spazio dei parametri i quali massimi andranno a rappresentare le curve con maggior probabilità di essere presenti nell'immagine.
Come definito in precedenza, andremo ad utilizzare la HT per il riconoscimento di forme circolari:

$$(x-a)^2 + (y-b)^2 = r^2 \tag{4.2}$$

La coppia di parametri (a, b) verrà utilizzata per individuare il centro della circonferenza, il parametro r per il raggio.
Vanno distinti due casi in cui può essere utilizzata questa funzione:

- Nel primo caso il raggio delle circonferenze da individuare è noto: r_0. Lo spazio dei parametri a questo punto si riduce a due dimensioni, poichè andranno stimate solo le posizioni dei centri.

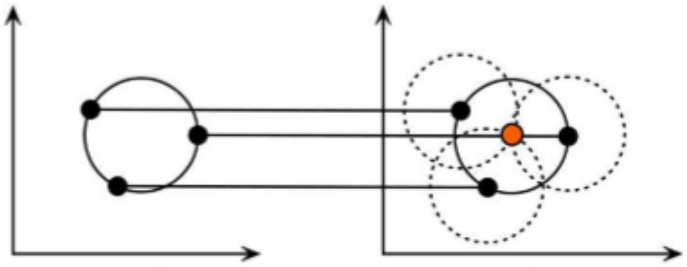

Figura 4.7: Rappresentazione di una circonferenza nello spazio immagine e nello spazio dei parametri.

In figura (4.7) a sinistra è rappresentata una circonferenza nello spazio immagine. Ad ogni punto (x,y) dello spazio immagine, considerando solamente i punti appartenenti ad un edge e non altri punti non interessanti, corrisponde un cerchio di raggio r_0 nello spazio dei parametri, avente centro in (x,y). Ogni punto della circonferenza genera una circonferenza a sua volta nello spazio dei parametri. I cerchi nello spazio dei parametri si intersecano in (a,b), che corrisponde al centro della circonferenza nello spazio immagine.

- Nel secondo caso, più complesso, il raggio non è noto a priori, ma compreso in un range di valori ammissibili. Ogni punto dello spazio immagine genera un cono nello spazio dei parametri, come in figura (4.8).

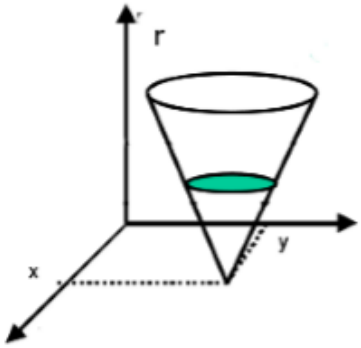

Figura 4.8: Superficie conica nello spazio dei parametri.

Per tutti i punti di interesse di una circonferenza di raggio non noto r, la terna da stimare (a,b,r) si troverà in corrispondenza della cella accumulatrice con il maggior numero di coni. Poiché il calcolo computazionale è molto superiore al caso precedente, l'operatore di Canny utilizzato al caso precedente svolge un ruolo fondamentale. Viene sfruttato il metodo del gradiente: per ogni punto non nullo dell'immagine filtrata si determina il gradiente locale. Poiché un edge è perpendicolare alla direzione del vettore gradiente nel punto in cui viene calcolato, il centro di un cerchio sarà dato dalle intersezioni delle rette normali alla circonferenza. Un accumulatore bidimensionale conterrà i centri candidati, il raggio verrà ottenuto traminte il calcolo della distanza dai pixel di edge al centro del cerchio.

Verrà ora fatto un esempio di applicazione di HT per individuazione di forme circolari.

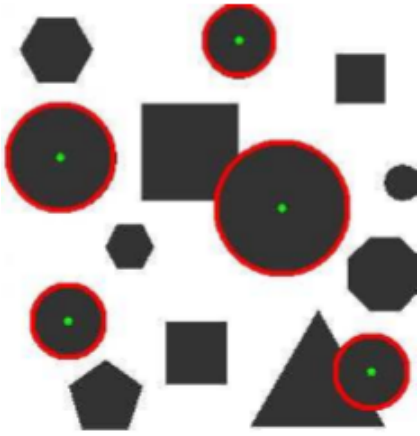

Figura 4.9: Esempio di trasformata di Hough per individuazione di forme circolari.

4.2 Algoritmo di controllo

In questa sezione verrà descritto l'approccio scelto con cui affrontare la parte relativa al controllo tramite visione. Dopo una breve introduzione sul problema, verrà introdotto il controllo utilizzato.

4.2.1 Approccio al problema

Come accennato precedentemente, per la risoluzione del processo di avvitamento non si può assumere che gli oggetti manipolati siano due cilindri. I denti del corpo della pipetta dovranno incastrarsi nelle fessure del cappuccio per effettuare una corretta esecuzione di Avvitamento. Questo porta a dover allineare, oltre agli assi z degli oggetti manipolati, anche gli assi x e y. Questa tipologia di controllo è necessaria poichè non è possibile posizionare gli oggetti manipolabili esattamente nella stessa posizione, sia per motivi di ottimizzazione dei tempi, sia per motivi di possibili disturbi. In caso di disallineamento, se non venisse effettuato alcun tipo di correzione, durante l'operazione di avvitamento potrebbero verificarsi delle rotture degli oggetti manipolati.
L' algoritmo di controllo proposto in seguito punta a fornire una corretta esecuzione dell'allineamento relativo degli oggetti manipolati, qualunque sia la loro posizione iniziale, per permettere una corretta esecuzione dell'operazione di Avvitamento.

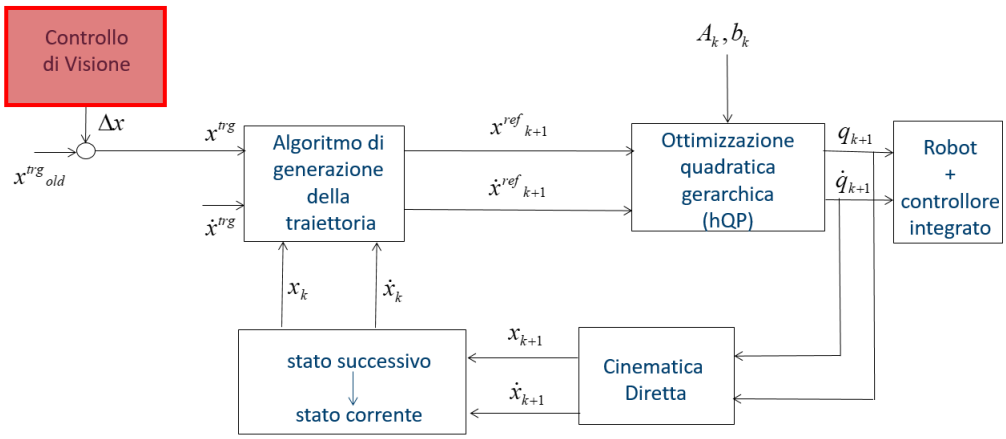

Figura 4.10: Schema di controllo con implementazione del controllo di visone.

Come si può notare dalla figura (4.10), lo schema di controllo precedente, viene ampliato aggiungendo un contributo correttivo. Il sistema di visione permette di calcolare la correzione della traiettoria e modificare la posizione target che verrà fornita successivamente all'algoritmo di generazione della traiettoria.
Nello schema a blocchi, il controllo di visione non ha nessuna freccia in input. Questa scelta serve ad identificare la diversità dei dati che riceve in ingresso rispetto ad un

controllore tradizionale. Il blocco dedicato alla correzione del target basato sulla visione, infatti, riceverà come input delle immagini e restituirà in output una variazione di posizione/orientamento (Δx).

4.2.2 Implementazione

Lo scopo di questo controllo consiste nel fornire una correzione all'allineamento degli assi, successiva all'allineamento fornito automaticamente dal robot.
Come descritto nella sezione riguardante la formalizzazione di task Dual Arm di questa tesi, le terne di rotazione degli oggetti manipolati devono essere fissate prima di poter formalizzare le azioni. Non va esclusa la considerazione che gli oggetti non debbano mantenere sempre la stessa posizione iniziale, portando continui disallineamenti lungo le direzioni x e y. Questo controllo serve a correggere eventuali errori di orientamento tramite il calcolo dell'obiettivo di rotazione lungo l'asse z.
La sequenza di operazioni prevista da questo controllo è la seguente:

- Allineamento della telecamera con il cappuccio della pipetta;
- Esecuzione di uno snapshot: viene catturata un'immagine tramite la telecamera;

Figura 4.11: Immagine catturata dopo l'allineamento con il cappuccio.

- Calcolo dell'orientamento: vengono calcolate le posizioni di alcuni marker posti sul cappuccio della pipetta (pallini neri in Fig. 4.11). In questo modo è possibile trovare l'orientamento dell'oggetto in questione. L'orientamento viene espresso tramite due parametri α_1, rappresentante lo sfasamento in gradi, β_1 rappresentante il verso di rotazione dello sfasamento. β_1 può assumere i valori -1 e +1;
- Afferraggio del cappuccio della pipetta;
- Allineamento della telecamera con il corpo della pipetta;

- Esecuzione di uno snapshot;

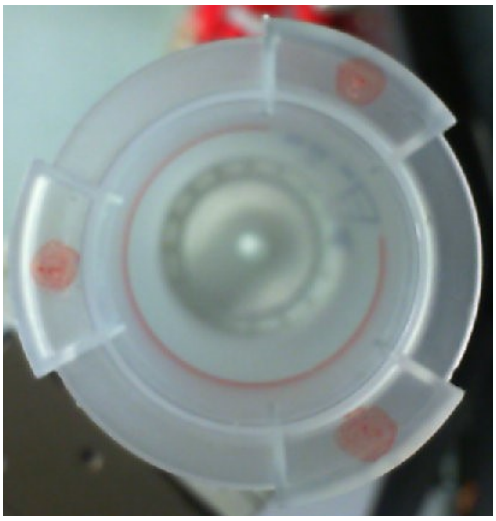

Figura 4.12: Immagine catturata dopo l'allineamento con il corpo.

- Calcolo dell'orientamento: vengono calcolate le posizioni di alcuni marker posti sul corpo della pipetta (pallini rossi in Fig 4.12). In questo modo è possibile trovare l'orientamento dell'oggetto in questione. L'orientamento viene espresso tramite due parametri α_2, rappresentante lo sfasamento in gradi, β_2 rappresentante il verso di rotazione dello sfasamento. β_2 può assumere i valori -1 e +1;
- Allineamento del cappuccio con il corpo della pipetta;
- Rotazione relativa tra i due corpi: rotazione di un angolo

$$\alpha = \alpha_2 * \beta_2 - \alpha_1 * \beta_1$$

  attorno all'asse z ;
- Procedura di Avvitamento: dopo aver eseguito la correzione dell'errore, viene eseguita l'operazione di Avvitamento sfruttando un controllo di impedenza e il controllo di forza.

4.2.3 Funzionamento dell'algoritmo

In questa sezione verrà descritto il funzionamento dell'algoritmo per il calcolo dell'orientamento di un oggetto rispetto ad una posizione ideale. Lo stesso algoritmo è applicato per riconoscere l'orientamento del corpo e del cappuccio della pipetta.
L'algoritmo per il calcolo dell'orientamento si articola in 4 passaggi, i quali verranno spiegati riportando un esempio.

- Recupero dell'immagine tramite sensore di visione: viene eseguito uno snapshot dell'ambiente di lavoro e utilizzato come punto di partenza per le successive operazioni.

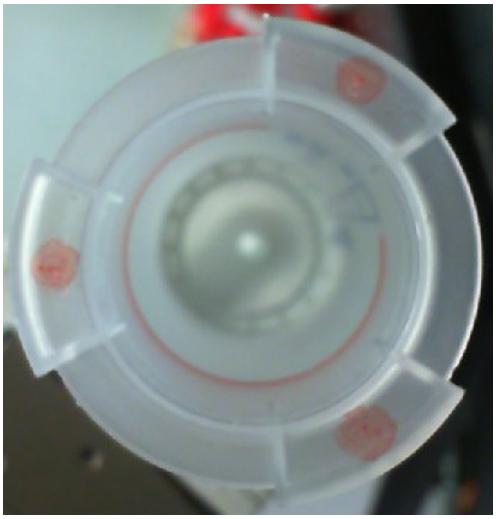

Figura 4.13: Esempio di immagine catturata.

- Applicazione del filtro di Canny: l'immagine precedente viene processata in modo tale che risaltino solamente i contorni degli oggetti.
- Applicazione della trasformata di Hough: l'immagine precedente viene utilizzata come base per il calcolo della posizione dei marker.

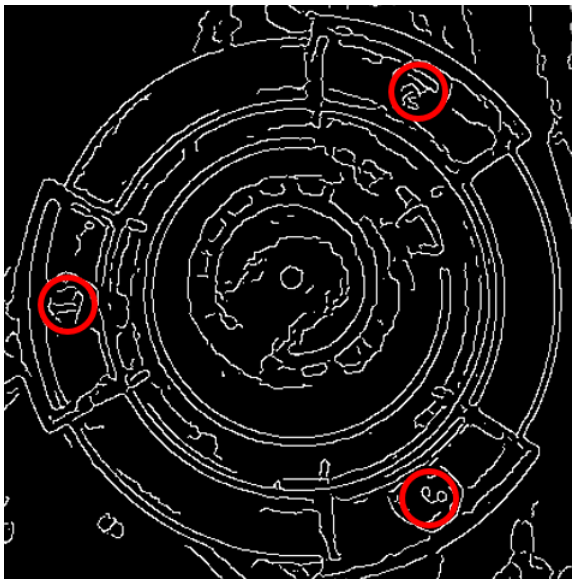

Figura 4.14: Esempio applicazione di filtro di Canny e trasformata di Hough.

Le posizioni del centro dei cerchi verrà utilizzata al passaggio successivo

- Calcolo dello sfasamento. Il calcolo dello sfasamento si articola a sua volta in diversi passi:

  1. Calcolo delle distanze dei centri rispetto a posizioni ideali;

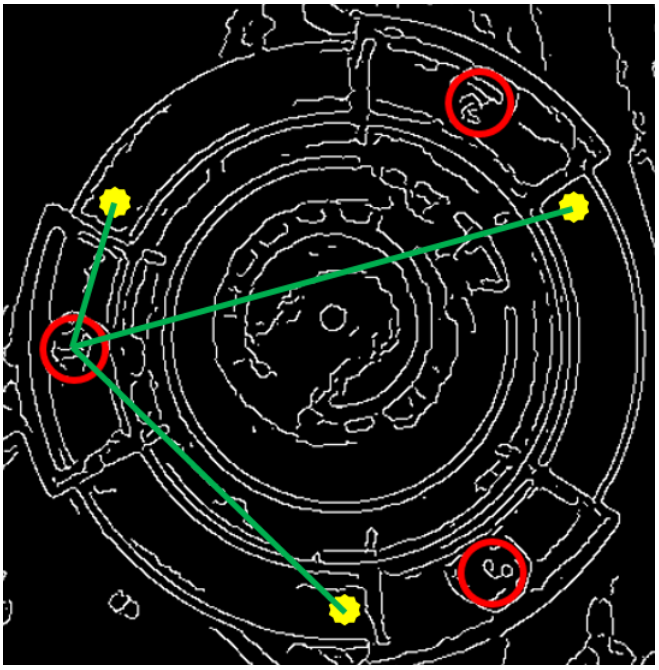

Figura 4.15: Calcolo delle distanze da posizioni ideali.

In rosso viene espressa la posizione reale dei cerchi, in giallo la posizione ideale, in verde la distanza tra i centri. L'operazione viene effettuata per tutti i centri.

  2. Associazione del centro reale al centro ideale corretto;

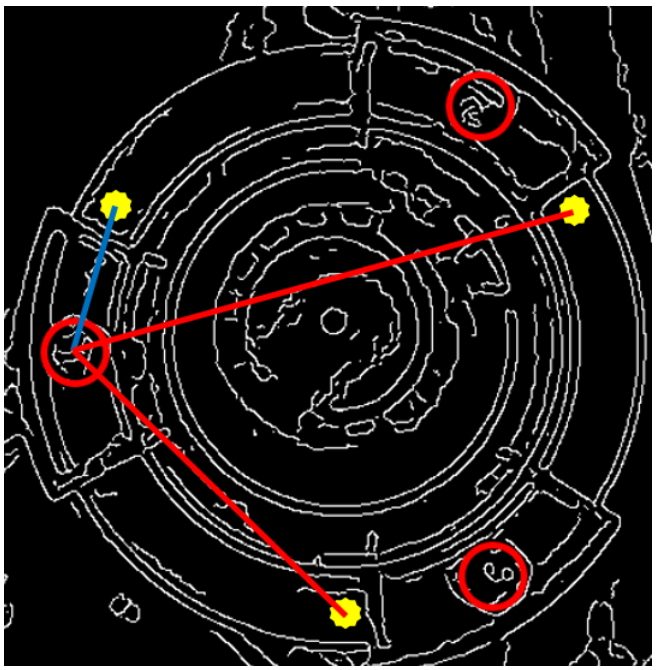

Figura 4.16: Associazione del centro reale con il centro ideale.

In rosso vengono espressi i centri ideali scartati, in blu quello accettato. L'operazione viene effettuata per tutti i centri e, alla fine, ogni centro reale sarà associato ad un centro ideale differente.

3. Calcolo dello sfasamento tramite operazioni geometriche;

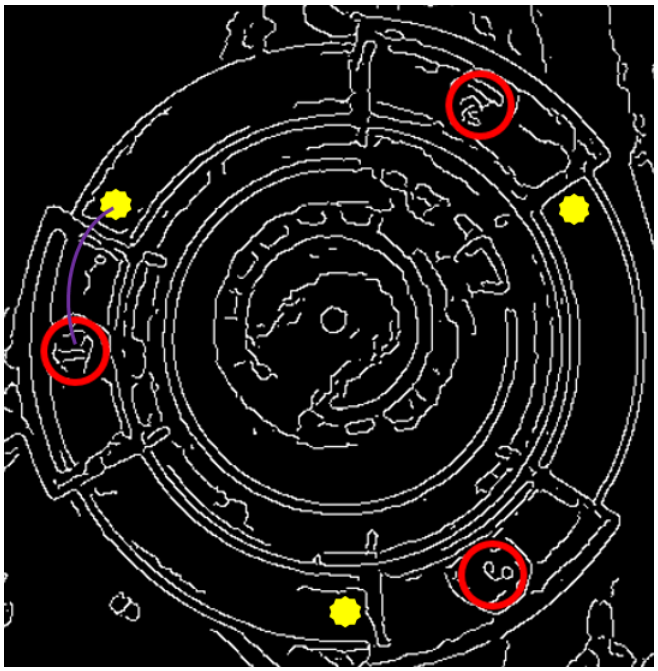

Figura 4.17: Calcolo dello sfasamento.

In viola viene espresso l'angolo α di sfasamento.

4.3 Implementazione

Questo paragrafo verrà dedicato all'implementazione delle funzioni appena descritte in ambiente di simulazione e in ambiente sperimentale.
Per quanto riguarda l'implementazione in simulazione, verranno descritte le funzioni Matlab utilizzate e i tempi di risposta medi per il calcolo. Successivamente verranno descritte le stesse funzioni ma tramite l'utilizzo di librerie OpenCV. Questa distinzione viene effettuata poichè in ambito sperimentale, le funzioni Matlab non rientrano nei tempi di ciclo di comunicazione robot-controllore.

4.3.1 Matlab

Come appena menzionato, verranno ora descritte funzioni rappresentanti il filtro di Canny e la trasformata di Hough in ambiente Matlab.

Operatore di Canny

Per l'implementazione dell'operatore di Canny in Matlab, viene utilizzata la funzione *edge*.

$$BW = edge(I)$$

Questa funzione prende in input l'immagine I (di intensità o binaria) e restituisce l'immagine BW. BW ha la stessa dimensione di I, contiene 1 dove viene riconosciuto un bordo e 0 dove non viene riconosciuto.
Di base la funzione edge sfrutta l'operatore di Sobel per il riconoscimento di bordi, quindi per utilizzare l'operatore di Canny bisogna specificarlo nel seguente modo:

$$BW = edge(I,'canny')$$

Altri operatori disponibili sono: Prewitt, Roberts, log, zerocross e approxcanny.
A questa funzione è possibile aggiungere due specifiche per rendere il riconoscimento più stretto:

$$BW = edge(I,'canny', threshold, sigma)$$

sigma rappresenta la deviazione standard del filtro Gaussiano da applicare. Di default il valore di *sigma* è $\sqrt{2}$.
threshold è un vettore bidimensionale in cui vengono definite la soglia inferiore e la soglia superiore per il riconoscimento di bordi. Di default i valori vengono scelti automaticamente dalla funzione. Nel caso in cui si voglia specificare un solo scalare, esso viene impostato al valore superiore della soglia, mentre il valore inferiore viene impostato a 0.4 del valore superiore.
Il tempo necessario per l'esecuzione di questa operazione è $T \approx 0.61$ secondi.

Trasformata di Hough

Per l'implementazione della trasformata di Hough in Matlab viene utilizzata la funzione $imfindcircles$

$$centers = imfindcircles(A, radius)$$

Questa funzione riceve in input l'immagine A. Restituisce un vettore bidimensionale centers contenente le coordinate di tutte le circonferenze presenti in A con raggio approssimativamente uguale a radius.
La funzione è ampliabile nel seguente modo:

$$[centers, radii, metric] = imfindcircles(A, radius/radiusRange, Name, Value)$$

I valori in output sono i seguenti:

- centers: matrice $(n \times 2)$, nella quale prima colonna sono contenuti i valori delle coordinate x delle circonferenze rilevate, e nella seconda colonna le coordinate y.

- radii: vettore colonna contenente i raggi delle circonferenze rilevate.
- metric: forza relativa delle circonferenze rilevate, ovvero la forza relativa delle coordinate dei centri rilevati.

I valori in input, invece, sono i seguenti:

- A: immagine ricevuta in input. Può essere in scala di grigi, colorata o binaria.
- radius/radiusRange: scalare (nel caso di radius) o intervallo di valori (radiusRange) utilizzato per identificare il raggio delle circonferenze.
- (Name,Value): coppia di elementi aggiuntivi per rendere più specifica la ricerca. Name deve comparire tra apici e il valore Value corrispondente deve essere accettato tra i valori predefiniti di Matlab. Possono essere applicate più coppie di specifiche in una stessa ricerca. Questi elementi possono essere:
    1. ObjectPolarity: serve a indicare se gli oggetti circolari da ricercare sono più chiari o più scuri del background. I valori che può assumere sono *bright* e *dark*.
    2. Method: serve a indicare la tecnica utilizzata per calcolare il vettore di accumulazione. I valori che può assumere sono $PhaseCode$ ([31]) e $TwoStage$ ([32], [33]).
    3. Sensitivity: sensitività per il vettore di accumulazione. Maggiore è la sensitività, maggiore è il numero di circonferenze riconosciute. Il valore che può assumere Sensitivity è compreso tra 0 e 1.
    4. EdgeThreshold: valore della soglia del gradiente per riconoscere bordi all'interno dell'immagine. Il valore deve essere compreso tra 0 (minimo gradiente) e 1 (massimo gradiente).

Il tempo necessario per l'esecuzione di questa operazione è $T \approx 1.69$ secondi.

4.3.2 OpenCV

Poichè i tempi necessari all'utilizzo delle funzioni precedenti sono troppo elevati in Matlab, è stato neccessario cercare un ambiente di lavoro alternativo. La soluzione migliore è stata riscontrata tramite l'utilizzo delle librerie OpenCV.
La Open Source Computer Vision Library è una libreria scritta in C e C++ costituita da oltre 500 funzioni utili nel campo dell'image processing e della Computer Vision. Essendo liberamente distribuita garantisce sicurezza al codice ed è in continua evoluzione. Le funzioni scritte con questa libreria richiedono poca potenza di calcolo, rendendola il miglior strumento per l'esecuzione di esperimenti real-time.

Operatore di Canny

Per l'implementazione dell'operatore di Canny tramite l'utilizzo delle librerie OpenCV, viene utilizzata la seguente funzione:

$$void\ cvCanny(const\ CvArr*\ \ image,\ CvArr*\ \ edges,\ double\ \ threshold1,\ double\ \ threshold2,\ int\ \ aperture\_size = 3);$$

I parametri descritti sono i seguenti:

- image: immagine da elaborare. Deve essere singolo canale con profondità pari a 8 bit unsigned;
- edges: immagine che dovrà contenere i bordi individuati. Deve essere a singolo canale con profondità pari a 8 bit unsigned;
- threshold: i valori threshold1 e threshold2 sono rispettivamente soglia inferiore e soglia superiore da utilizzare per riconoscere i bordi. La funzione è progettata in modo tale da correggere eventuali errori nella definizione delle soglie: se threshold1 è maggiore di threshold2, i loro valori vengono invertiti automaticamente.
- aperture_size: dimensione dell'operatore di Sobel impiegato per le operazioni di smoothing e differenziazione.

Trasformata di Hough

Per l'implementazione della trasformata di Hough tramite l'utilizzo delle librerie OpenCV viene utilizzata la seguente funzione:

$$cvHoughCircles(CvArr*\ \ image,\ void*\ \ circle\_storage,\ int\ \ method,\ double\ \ dp,\ double\ \ min\_dist,\ double\ \ param1 = 100,\ double\ \ param2 = 300,\ int\ \ min\_radius = 0,\ int\ \ max\_radius = 0);$$

I parametri descritti sono i seguenti:

- image: immagine di input. Deve essere a 8 bit ma non viene considerata binaria;
- circle_storage: struttura dati necessaria a contenere la forma circolare;
- method: ammette solamente il valore predefinito $CV\_HOUGH\_GRADIENT$, definisce il metodo del gradiente;
- dp: permette di creare un accumulatore con risoluzione inferiore all'immagine sorgente se fissato ad un valore maggiore di 1;
- min_dist: distanza minima che deve intercorrere tra due circonferenze affinchè l'algoritmo li consideri distinti;

- param1: soglia utilizzata dall'algoritmo di Canny. Il valore settato corrisponde alla soglia alta, la soglia bassa corrisponde alla metà di questo valore;
- param2: soglia utilizzata dall'accumulatore;
- min_radius: raggio minimo di riconoscimento di una circonferenza;
- max_radius: raggio massimo di riconoscimento di una circonferenza.

4.4 Criticità di applicazione

In questa ultima sezione del capitolo verranno presentate le criticità nell'applicazione dell'algoritmo. Verranno prima presentati i limiti dovuti alla cattura dell'immagine e successivamente i limiti dovuti all'algoritmo in sè.
Come accennato in precedenza, l'input da dare all'algoritmo è un'immagine recuperata da un sistema di visione. Questo induce il non sapere a priori ciò che vedrà il robot. La criticità di questa incertezza risiede nel fatto che le successive applicazioni delle funzione di Image Analysis potrebbero produrre risultati inaspettati. Come descritto nelle sezioni 4.3.1 e 4.3.2, sia in ambiente Matlab, sia in ambiente OpenCV, le funzioni devono ricevere come input un'immagine e dei parametri per funzionare correttamente.
Gli errori più comuni che possono verificarsi sono i seguenti:

- Scelta errata dei parametri: nel caso in cui vengano scelti erronamente i parametri di applicazione alle funzioni di visione, i dati risultanti potrebbero essere completamente diversi.

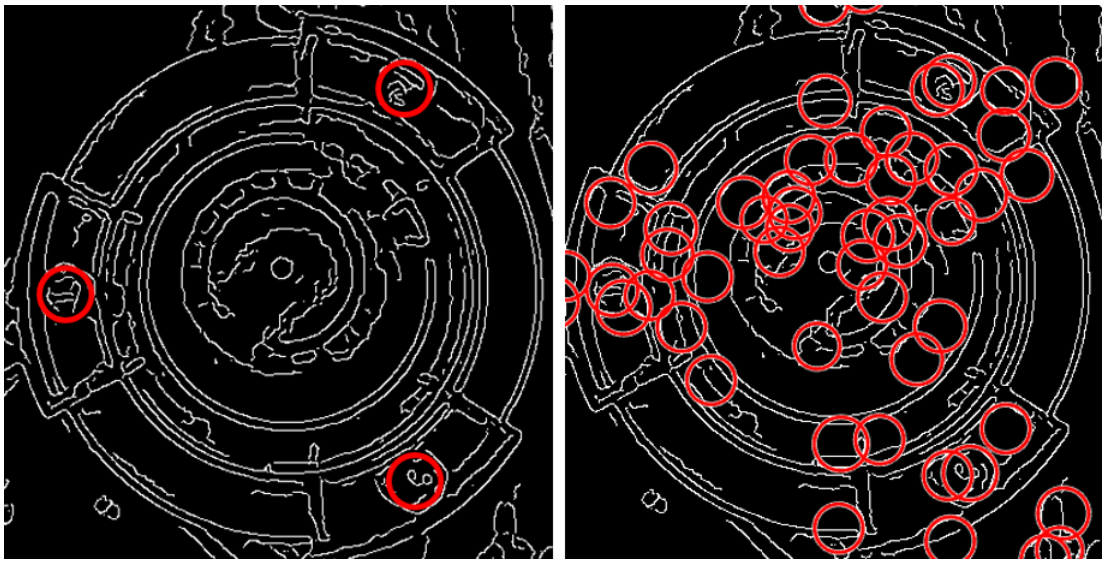

Figura 4.18: Confronto di applicazione di filtro di Canny con diversi valori di sensitività.

In figura (4.18) a sinistra viene utilizzata una sensitività pari a 0.9, a destra 0.95.

- Illuminazione sbagliata: nel caso in cui ci sia un'illuminazione sbagliata o che il materiale esaminato sia troppo riflettente, l'identificazione delle forme risulta impossibile. In Fig 4.11 si può notare un esempio di questo facendo riferimento al marker in alto a sinistra. La circonferenza ad esso associata risulta non calcolata.

- Occlusione dei segnalatori: nel caso in cui uno dei segnalatori venga occluso totalmente, o solo in parte, dalla presenza di un altro corpo, l'identificazione delle forme non avviene. In Fig 4.11 si può notare un esempio di occlusione facendo riferimento al marker posto nella parte inferiore dell'immagine.

- Sfocatura dell'immagine: nel caso in cui l'immagine di input sia sfocata, la lettura di qualsiasi dato risulta impossibile.

Figura 4.19: Esempio di immagine sfocata.

Una differente tipologia di errore può risultare dalla creazione dell'algoritmo di controllo. Come accennato durante la spiegazione dell'algoritmo, prima di calcolare l'angolo di sfasamento, ad ogni centro reale viene associato un centro ideale. Questo procedimento è necessario se l'angolo di sfasamento dovrà assumere segno positivo o negativo. L'associazione viene effettuata attraverso il calcolo della distanza minima tra i centri. L'identificazione dei centri non è perfetta, ma stimata.
Di conseguenza, nel caso in cui almeno due centri reali si trovino equidistanti dal medesimo centro ideale, potrebbe avvenire un assegnamento incorretto, portando al blocco dell'algoritmo.
I casi critici e i limiti dell'algoritmo sono stati testati eseguendo numerosi esperimenti in configurazioni differenti e multipli esperimenti nelle medesime configurazioni.

Capitolo 5

Stima delle forze e modello d'attrito

In questo capitolo verranno trattate la stima delle forze esterne e il modello di attrito. Poichè il robot utilizzato, ABB FRIDA, ha un carico molto limitato pari a circa 0.5Kg per braccio, risulta impossibile installare sensori di forza: questi, infatti, risultano troppo pesanti e ingombranti. Data la loro dimensione, questi sensori potrebbero portare alla limitazione dei movimenti, e a causa del loro peso potrebbero portare il robot a saturazione. Per questi motivi, si deve ricorrere all'utilizzo di stimatori di forza, che permettono la ricostruzione dei segnali di forza agenti sugli end effector.

5.1 Stima delle forze

Precedenti studi [34] e [35] hanno mostrato come gli osservatori potessero essere utilizzati per identificare failure di uno o più giunti, oppure come potesse essere identificata la collisione tra il robot e l'ambiente di lavoro. Questi studi sono stati eseguiti solo a livello di ambiente simulato.
Applicazioni sperimentali, condotte a livello di collaborazione tra uomo e robot, sono state compiute in [4].
Usando modelli precedenti proposti in [34] possiamo ricostruire le coppie ai giunti.
Consideriamo il modello dinamico del manipolatore:

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) + fr(q,\dot{q}) = \tau + \tau_{ext} \tag{5.1}$$

dove $B(q)$, $C(q,\dot{q})$, $g(q)$ e $fr(q,\dot{q})$ rappresentano rispettivamente la matrice d'inerzia, i termini centrifugali e di Coriolis, la forza gravitazionale e la forza d'attrito. τ indica la coppia motrice applicata al robot e τ_{ext} rappresenta le coppie esterne di interazione.
Definiamo i momenti generalizzati come:

$$p = B(q)\dot{q} \tag{5.2}$$

Derivando p e sostituendo $B(q)\ddot{q}$ nel modello (5.1), otteniamo:

$$\dot{p} = \dot{B}(q)\dot{q} - C(q,\dot{q})\dot{q} - g(q) - fr(q,\dot{q}) + \tau + \tau_{ext} = \alpha(q,\dot{q}) + \tau_{ext} \tag{5.3}$$

Se il termine $\alpha(q, \dot{q})$ è conosciuto, il residuale può essere espresso come:

$$r = K_{oss} \left[p - \int_0^{t_k} (\alpha(q, \dot{q}) + r) dt \right] \tag{5.4}$$

e, derivandolo, otteniamo:

$$\dot{r} = K_{oss}(\tau_{ext} - r) \tag{5.5}$$

Il vettore r converge asintoticamente alle coppie esterne dei giunti τ_{ext} solo se la matrice K_{oss} è definita positiva.
La relazione tra le coppie esterne τ_{ext} e il vettore f delle forze e dei momenti, può essere espresso come:

$$J(q)^T f = \tau_{ext} \simeq r \tag{5.6}$$

Poichè il manipolatore usato è cinematicamente ridondante, il sistema è sovravincolato. Questo implica che la soluzione dell'equazione è ottenuta da:

$$f = (J(q)^T)^\dagger \cdot \tau_{ext} \tag{5.7}$$

† indica la matrice pseudo-inversa di Moore-Penrose.
τ_{ext} è espressa per il braccio destro del robot, e f, ottenuta dalla precedente equazione, è espressa con riferimento alla terna di rotazione dell'end effector. Dobbiamo esprimere f nella terna di rotazione del tool, poichè vogliamo stimare la forza analizzando i valori della coppia quando i corpi sono a contatto.
Questa trasformazione è ottenuta mediante $A_{R_{tool}} = A_{right} \cdot T_{R_{tool}}$, descritte nel capitolo di formalizzazione di task Dual Arm.
In questo modo possiamo esprimere f come:

$$f = \begin{bmatrix} AR^{rot}_{tool} & 0 \\ 0 & AR^{rot}_{tool} \end{bmatrix} \cdot (J(q)^T)^\dagger \cdot \tau_{ext} \tag{5.8}$$

5.2 Modello di attrito

Come descritto precedentemente, per costruire l'osservatore di coppia è necessario conoscere il modello dinamico del manipolatore. La matrice d'inerzia, le forze di Coriolis e la forza gravitazionale possono essere ottenute con grande precisione se si hanno i dati forniti dal costruttore sul robot.
L'identificazione del modello di attrito, invece, può presentare particolari problematiche alle basse velocità.
La teoria classica mostra i forti limiti in queste regioni, mentre i modelli dinamici espressi in [36] offrono una stima migliore. Il modello adottato, che descrive le coppie correlate con il termine di attrito in (5.1) può essere ottenuto come [2]:

$$\tau^F = \tau^O + \tau^C \tanh(130\dot{q}) + \beta \tag{5.9}$$

τ^C e τ^O sono rispettivamente l'attrito Coulombiano e un offset.

$$\beta = D^+\dot{q} \quad se \;\; \dot{q} > 0$$
$$\beta = D^-\dot{q} \quad se \;\; \dot{q} < 0$$

D^+ e D^- rappresentano i coefficienti di attrito viscoso per velocità positive e negative rispettivamente.
Errori fatti nel processo di identificazione possono condurre a misure di forze e coppie che in realtà sono assenti. Questo può condurre, di conseguenza, a comportamenti non voluti, nel momento in cui è necessario identificare se il contatto tra gli oggetti è avvenuto o no. Per evitare questo problema e per aumentare le soglie oltre la quale una coppia è individuata, viene applicata una zona morta:

$$\tau^{thr} = \tau^{still} + \tau^{move}\,\mathrm{sec}(30\dot{q}) \tag{5.10}$$

τ^{still} e τ^{move} sono utilizzati per differenti valori di velocità. In particolare τ^{move} è utilizzato quando due oggetti differenti sono incastrati e si trascinano a vicenda.

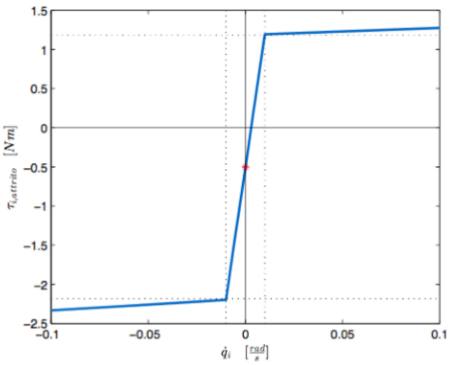

Figura 5.1: Identificazione del modello di attrito.

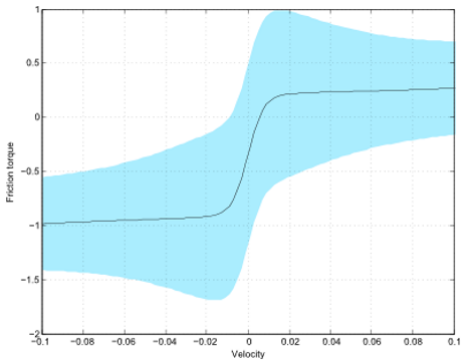

Figura 5.2: Soglie per il modello di attrito.

Capitolo 6

Controllo di Impedenza

Il controllo di impedenza è una metodologia di controllo dei robot, utilizzata per far assumere al robot un particolare comportamento meccanico. Come descritto in [37], l'impedenza meccanica è la relazione tra le forze applicate ad un oggetto e il moto risultante. L'obiettivo è quello di associare al robot un comportamento generale di tipo massa-molla-smorzatore caratterizzato da M, K e D matrici diagonali.

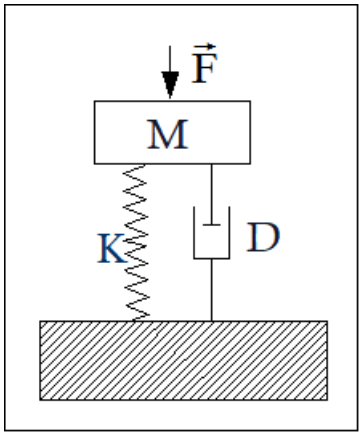

Figura 6.1: Esempio di modello massa-molla-smorzatore.

Una particolare applicazione per questo tipo di controllo si basa sulla guida manuale del robot, ovvero cercare di insegnare al robot i movimenti da compiere manualmente. Nel caso di robot a due braccia, un punto critico di questo controllo riguarda il posizionamento relativo tra le braccia stesse: uno scorretto posizionamento, infatti, potrebbe portare all'errata manipolazione degli oggetti, portandoli alla rottura o alla deformazione. Proprio per questo motivo, l'obiettivo del controllo di impedenza è quello di compensare gli errori dovuti al posizionamento relativo. Come compito aggiuntivo, ha quello di assecondare l'esecuzione del processo sulla base delle forze misurate o, come nel nostro caso, stimate. Esistono due tipi di impedenza meccanica:

1. l'impedenza meccanica traslazionale, dedita alla descrizione di spostamenti dovuti all'effetto di forze;

2. l'impedenza meccanica rotazionale, simile alla precedente, ma utilizzata per descrivere spostamenti come conseguenza dell'applicazione di momenti.

6.1 Impedenza traslazionale

La posizione dell'end effector può essere descritta da un vettore colonna di dimensione (3x1):

$$p_c = [x_c \;\; y_c \;\; z_c]^T$$

Supponendo di voler raggiungere la posizione desiderata:

$$p_d = [x_d \;\; y_d \;\; z_d]^T$$

definiamo un vettore

$$\Delta p_{cd} = p_c - p_d$$

rappresentante la differenza tra le due posizioni.
Possiamo definire l'impedenza traslazionale come:

$$M_t \Delta \ddot{p}_{cd} + D_t \Delta \dot{p}_{cd} + K_t \Delta p_{cd} = f \tag{6.1}$$

In questa equazione f rappresenta il vettore delle forze agenti sull'end effector; M_t, K_t e D_t sono matrice definite positive che rappresentano massa, elasticità e smorzamento rispettivamente.
Il primo termine dell'equazione (6.1) rappresenta la forza inerziale agente sul corpo di massa m_i e l'energia cinetica che dipende dalla velocità $\dot{p}_c$. Il secondo termine rappresenta le forze dissipative. L'ultimo termine, nella parte sinistra dell'equazione, rappresenta le forze imposte da una molla virtuale, la cui rigidezza è rappresentata dalla matrice K_t e la cui energia potenziale dipende da Δp_{cd}.

6.2 Impedenza rotazionale

L'orientamento dell'end effector può essere anch'esso rappresentato da un vettore colonna di dimensione (3x1):

$$\phi = [\varphi \;\; \theta \;\; \psi]^T$$

che ne racchiude la rappresentazione secondo gli angoli di Eulero.
La relazione tra la derivata di questi angoli $\dot{\phi}$ e le loro velocità angolari ω è espressa come:

$$\omega = T(\phi) \cdot \dot{\phi}$$

dove $T(\phi)$ è una matrice di trasformazione che dipende dalla scelta degli angoli di Eulero. Esprimiamo come ϕ_c l'orientamento effettivo dell'end effector, e come ϕ_d l'orientamento desiderato. Esprimiamo la loro differenza come:

$$\Delta \phi_{cd} = \phi_c - \phi_d$$

L"impedenza rotazionale può essere definita come:

$$M_r \Delta\ddot{\phi}_{cd} + D_r \Delta\dot{\phi}_{cd} + K_r \Delta\phi_{cd} = T^T(\phi_c) \cdot \mu \tag{6.2}$$

In (6.2) M_r, D_r e K_r sono matrici definite positive. Descrivono rispettivamente l'inerzia rotazionale, lo smorzamento rotazionale e la rigidezza rotazionale. μ rappresenta i momenti agenti sull'end effector e $T^T(\phi_c)$ è una matrice di rotazione.

6.3 Controllo di impedenza per un robot Dual Arm

In questa sezione verrà descritto come implementare un algoritmo basato sul controllo di impedenza per l'esecuzione di un task Dual Arm.
In questa tesi, l'implementazione è avvenuta sia per l'operazione di inserimento, sia per l'operazione di avvitamento. Nel primo caso è utilizzato come unico controllore, nel secondo, invece, viene utilizzato successivamente al controllo basato sulla visione, il quale ne permette un'esecuzione semplificata, e congiuntamente al controllo di forze.
Utilizzando l'algoritmo di generazione della traiettoria è possibile trovare il riferimento di posizione x_{k+1}^{rif} e il riferimento di velocità $\dot{x}_{k+1}^{rif}$ a partire dalle coordinate target x^{trg} e $\dot{x}^{trg}$, e dalle coordinate attuali x_k e $\dot{x}_k$. Lo scopo del controllo di impedenza consiste nella correzione della traiettoria da imporre al robot, basandosi sulle forze e sui momenti percepiti dall'end effector. Questo serve ad evitare la possibile rottura o deformazione dei componenti manipolati.
Poichè stiamo operando in un contesto Dual Arm, a differenza di un'operazione Single Arm in cui possiamo prevedere il movimento del braccio, entrambe le braccia vengono mosse contemporaneamente e in maniera non conosciuta a priori. Risulta impossibile stabilire posizione e orientamento raggiunti dalle braccia a causa di possibili disturbi e, di conseguenza, non è possibile utilizzare la molla virtuale.
In seguito a questa breve premessa, possiamo scrivere l'espressione dell'impedenza traslazionale come

$$M_t \Delta\ddot{x} + D_t \Delta\dot{x} = f \tag{6.3}$$

Da (6.3) ricaviamo l'espressione dell'accelerazione nello spazio operazionale:

$$\Delta\ddot{x} = M_t^{-1}(f - D_t \Delta x) \tag{6.4}$$

Nello stesso modo, possiamo scrivere l'impedenza rotazionale come:

$$M_r \Delta\ddot{\phi} + D_r \Delta\dot{\phi} = T(\phi)^T \mu \tag{6.5}$$

dalla quale ricaviamo l'accelerazione angolare nello spazio operazionale:

$$\Delta\ddot{\phi} = M_r^{-1}(T(\phi)^T \mu - D_r \Delta\dot{\phi}) \tag{6.6}$$

A questo punto possiamo calcolare la variazione di posizione e velocità desiderate come:

$$\begin{cases} \Delta x_{k+1} = T_s \cdot \Delta \dot{x}_k + \frac{T_s^2}{2} \cdot \Delta \ddot{x}_k \\ \Delta \dot{x}_{k+1} = \Delta \dot{x}_k + T_s \cdot \Delta \ddot{x}_k \end{cases} \tag{6.7}$$

Sostituendo (6.4) in (6.7) possiamo ricavare l'espressione di Δx_{k+1} e $\Delta \dot{x}_k + 1$.
La variazione di posizione e velocità angolari possono essere calcolate come:

$$\begin{cases} \Delta \phi_{k+1} = T_s \cdot \Delta \dot{\phi}_k + \frac{T_s^2}{2} \cdot \Delta \ddot{\phi}_k \\ \Delta \dot{\phi}_{k+1} = \Delta \dot{\phi}_k + T_s \cdot \Delta \ddot{\phi}_k \end{cases} \tag{6.8}$$

Similarmente al caso precedente, per quanto riguarda i valori angolari, sostituendo (6.6) in (6.8) possiamo ricavare $\Delta \phi_{k+1}$ e $\Delta \dot{\phi}_{k+1}$.
Per poter considerare i valori delle forze stimate agenti sull'end effector, i riferimenti di traiettoria devono essere modificati nel seguente modo:

$$\begin{aligned} x^{rif}_{k+1,new} &= x^{rif}_{k+1} + \Delta x_{k+1} \\ \dot{x}^{rif}_{k+1,new} &= \dot{x}^{rif}_{k+1} + \Delta \dot{x}_{k+1} \\ \phi^{rif}_{k+1,new} &= \phi^{rif}_{k+1} + \Delta \phi_{k+1} \\ \dot{\phi}^{rif}_{k+1,new} &= \dot{\phi}^{rif}_{k+1} + \Delta \dot{\phi}_{k+1} \end{aligned} \tag{6.9}$$

Lo schema a blocchi completo per la generazione di traiettoria, con il filtro di impedenza implementato in feed forward, è il seguente:

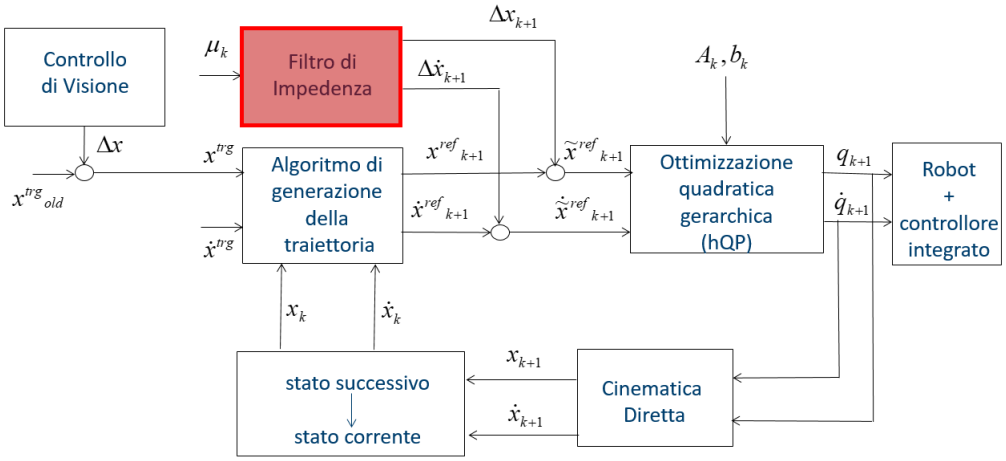

Figura 6.2: Schema a blocchi completo con Controllo di Impedenza.

Questo filtro riceve come input le forze stimate agenti sull'end effector all'istante k e restituisce come output la variazione di posizione Δx_{k+1} e velocità $\Delta \dot{x}_{k+1}$ all'istante $k+1$. La stessa operazione può essere fatta prendendo in considerazione i problemi di orientamento angolare.

Dobbiamo fornire come input al filtro, l'output della cinematica diretta. Questa operazione viene effettuata per evitare fenomeni di wind-up: in caso di saturazione, l'end effector potrebbe non essere in grado di seguire il riferimento fornito del nostro filtro.
Nell'implementazione del filtro, le componenti lungo l'asse z e ϕ_z non sono modificate

6.3.1 Implementazione del Controllore

Al fine di ottenere una descrizione ottima del comportamento del sistema, la scelta delle matrici M, K e D è essenziale per l'impedenza. Come descritto nel paragrafo precedente, non viene utilizzata la matrice K poichè stiamo trattando un'operazione Dual Arm.
La scelta dei parametri di controllo influenza fortemente la risposta del sistema:

- scegliendo i parametri in modo tale da incrementare il valore di D e diminuire il valore di M, il robot avrà un comportamento più reattivo, ovvero avrà una risposta più veloce. Quando vengono identificati i valori di forza e momento, il robot riduce molto velocemente gli errori di posizione e orientamento. Se il robot è troppo reattivo, però, si può finire in un caso di instabilità.

- viceversa, scegliendo i parametri in modo tale da aumentare il valore di M e diminuire il valore di D, il robot sarà più conservativo. Questo induce il sistema ad essere più lento nella risposta, una volta identificati i valori di forza e momento. La conseguenza di avere un sistema più lento, però, comporta uno stress minore da parte dei giunti.

In questa tesi vengono scelti valori diversi a seconda dell'operazione da compiere, seguendo i principi appena discussi.
Nell'operazione di Inserimento, vengono utilizzati i seguenti valori:

- Matrice di Inerzia Traslazionale $M_t = 0.5 \cdot \mathrm{I}\ [Kg]$
- Matrice di Smorzamento Traslazionale $D_t = 10 \cdot \mathrm{I}\ [Kg/s]$
- Matrice di Inerzia Rotazionale $M_r = 1 \cdot \mathrm{I}\ [Nms^2]$
- Matrice di Smorzamento Rotazionale $D_r = 10 \cdot \mathrm{I}\ [Nms]$

Nell'operazione di Avvitamento, vengono utilizzati i seguenti valori:

- Matrice di Inerzia Traslazionale $M_t = 0.5 \cdot \mathrm{I}\ [Kg]$
- Matrice di Smorzamento Traslazionale $D_t = 50 \cdot \mathrm{I}\ [Kg/s]$
- Matrice di Inerzia Rotazionale $M_r = 0.5 \cdot \mathrm{I}\ [Nms^2]$
- Matrice di Smorzamento Rotazionale $D_r = 50 \cdot \mathrm{I}\ [Nms]$

I rappresenta una matrice identità di dimensione opportuna.
Sono stati effettuati numerosi esperimenti e la risposta migliore del sistema è stata ottenuta utilizzando questi valori.

Capitolo 7

Risultati sperimentali

Quest'ultimo capitolo verrà dedicato all'implementazione della teoria trattata fino ad ora. Verrà inizialmente descritto l'apparato sperimentale, con particolare attenzione alla creazione dei gripper. Successivamente verranno descritti la macchina a stati del processo, la simulazione eseguita in Matlab e i risultati ottenuti sul robot.

7.1 Apparato sperimentale

In questa sezione verranno presentati i principali componenti necessari ad un'implementazione concreta del processo.
Il setup sperimentale è composto da:

- Manipolatore Dual Arm cinematicamente ridondante con 14 gradi di libertà;
- 2 gripper pneumatici;
- interfaccia di controllo e *teach pendant*;
- una telecamera Microsoft LifeCam;
- 2 tools personalizzati, disegnati appositamente per afferrare gli oggetti.

Il manipolatore robotico utilizzato è un prototipo di ABB Robotics chiamato FRIDA.

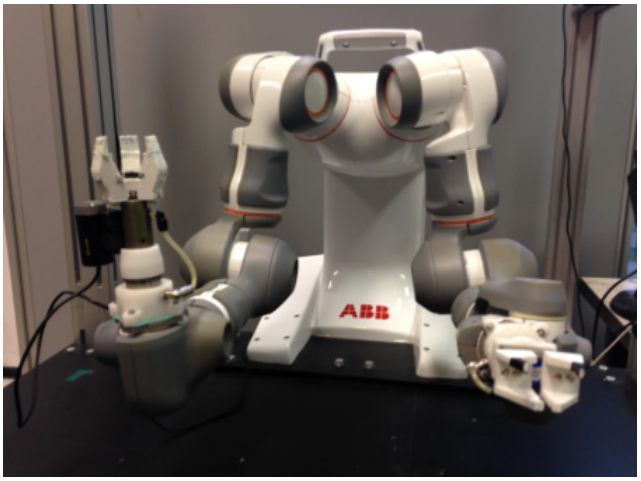

Figura 7.1: Robot ABB FRIDA presente in laboratorio.

Predecessore del più famoso robot ABB YuMi, è progettato per ottenere una collaborazione stretta con l'uomo e l'ambiente di lavoro. Grazie ai suoi 14 gradi di libertà, 7 per braccio, FRIDA è il manipolatore più adatto alla realizzazione dell'esperimento: la ridondanza cinematica permette infinite possibilità di risoluzione dei problemi precedentemente discussi.
Grazie alle soffici coperture gommose, rende sicuro il lavoro umano in prossimità del robot, poiché eventuali impatti accidentali vengono attutiti. Il robot è integrato con un unità di controllo centrale, IRC5, contenente sia l'elettronica di potenza destinata al rifornimento di energia, sia la parte di segnali, necessaria per l'implementazione degli anelli di controllo.
Il robot è equipaggiato con due gripper SCHUNK, uno centrifugale e uno parallelo, controllati da un circuito pneumatico. Questi gripper devono essere piccoli e leggeri, poichè, come citato in precedenza, il robot ha un carico massimo di 0.5Kg per braccio. I gripper sono controllati da valvole solenoidali installate in un circuito adatto e collegate ad una scheda di acquisizione.

Figura 7.2: Gripper SCHUNK centrifugale.

Il gripper centrifugale è montato sul braccio destro. Il suo scopo riguarda la manipolazione di 3 gripper personalizzati destinati all'afferraggio del cappuccio e dello stantuffo della pipetta.

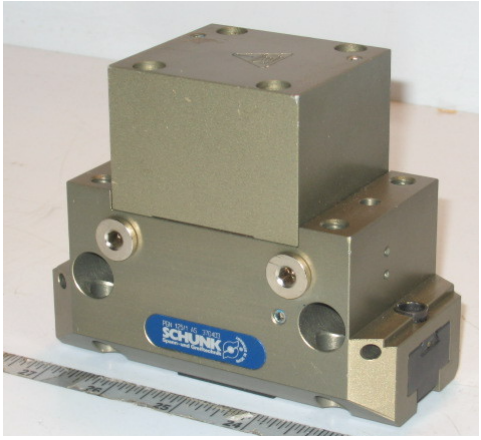

Figura 7.3: Gripper SCHUNK parallelo.

Il gripper parallelo è montato sul braccio sinistro ed è destinato a sorreggere il gripper personalizzato adibito a sostegno del corpo della pipetta.
Per il controllo di questi gripper e, in generale, del robot, viene utilizzata un'interfaccia di controllo: OPCOM. Inserendo i valori riportati in figura (7.4) si ottiene un effetto differente, di apertura o chiusura del gripper corrispondente. Per ottenere contemporaneamente due azioni distinte (esempio: apertura gripper sinistro e chiusura gripper destro) è necessario inserire una somma dei valori descritti in tabella.

Hex	Dec	Note
0x00	0	rest position
0x01	1	Frida left close
0x02	2	Frida left open
0x03	4	Frida right close
0x04	8	Frida right open
0x10	16	IRB140 close
0x20	32	IRB140 open

Figura 7.4: Valori utilizzati per modificare lo stato dei gripper.

Con l'interfaccia OPCOM, associata ad un controllore esterno, è possibile caricare i propri programmi sul robot, simularne il comportamento senza attivare il manipolatore o ottenere una realizzazione vera e propria del processo.
Parallelamente ad OPCOM, il *teach pendant* è utilizzato per controllare il robot.

Figura 7.5: Teach Pendant.

Il *teach pendant* è un'unita di controllo separata dal manipolatore, fornito di un touch screen, un joystick e altri bottoni per diverse funzionalità. Questa unità è semplice e intuitiva, e permette l'esecuzione di comandi basilari: movimento del robot, calibrazione degli encoder, blocco/sblocco dei freni meccanici.
La telecamera Microsoft LifeCam viene posta sul braccio destro del robot, leader nelle operazioni.

Figura 7.6: Telecamera.

Propone una risoluzione di 1280x720 pixels, la possibilità di realizzare video a 30 FPS ed è equipaggiata con una funzione di autofocus. Ha un peso di 95 grammi, 46 mm di profondità, 24 mm di altezza e 24 mm di larghezza. Questi dati sono fondamentali per la compensazione del tool dedita alla stima delle forze.

7.1.1 Gripper personalizzati

Particolare attenzione deve essere prestata ai gripper personalizzati. Tutti i componenti dei gripper personalizzati sono stati realizzati utilizzando un software di disegno CAD (SolidWorks) e stampati con una stampante 3D presente nel laboratorio MERLIN.

Figura 7.7: Stampante 3D CubePro.

I gripper sono stati pensati e realizzati per completare l'intero processo automaticamente e senza interruzioni. Poichè il robot ha un carico massimo molto ridotto, sono stati ridotti tutti i possibili eccessi di materiale.
Verrà prima analizzato il gripper sinistro.

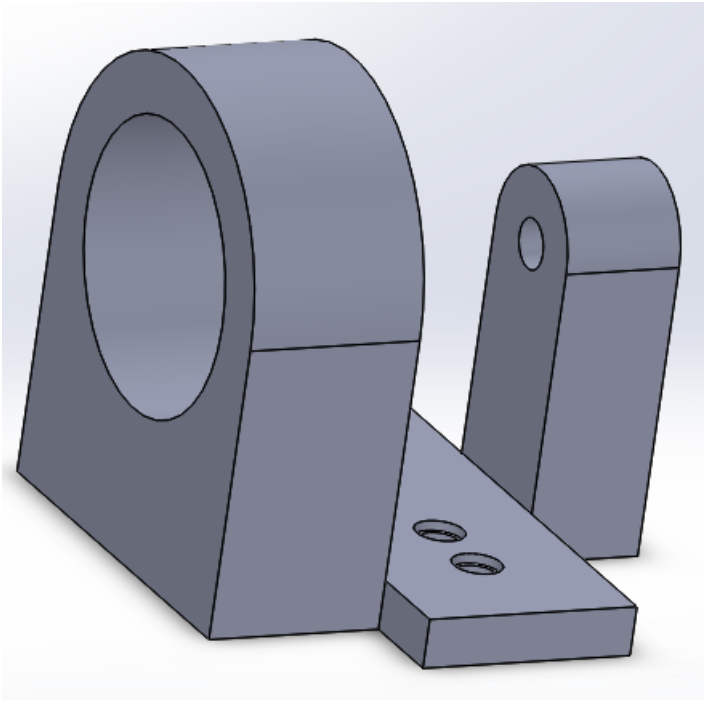

Figura 7.8: Modello CAD del gripper sinistro.

Il braccio sinistro del robot, durante l'intero processo, è adibito alla manipolazione del corpo della pipetta. Per questo motivo è stato realizzato un tool apposito che riuscisse a sostenere e mantenere bloccato l'oggetto sia nella fase di inserimento, sia nella fase di avvitamento. Per pura semplicità, è stato scelto di realizzare questo gripper a corpo unico, invece che in due sezioni separate, che si chiudessero ad afferrare l'oggetto, poichè durante tutta l'operazione non sono richiesti cambi di presa.

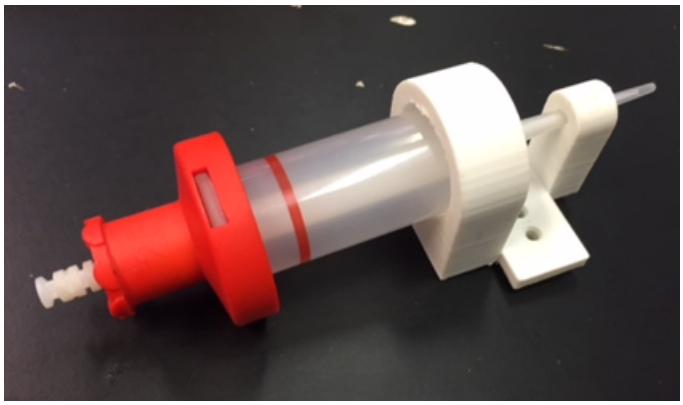

Figura 7.9: Gripper sinistro completo.

Il gripper destro, invece, ha bisogno di più attenzione.
Come anticipato, il braccio destro è adibito alla presa sia del cappuccio, sia dello stantuffo della pipetta. Per completare l'operazione in modo completamente automatico, è risultato necessario creare un gripper che riesca ad afferrare entrambi gli oggetti. Il gripper SCHUNK centrifugale permette l'utilizzo di 3 tool personalizzati, che si aprono a 120 gradi di distanza. L'apertura massima è di 3mm, quindi è stato necessario uno studio approfondito del problema.
La prima versione di singolo dito realizzata è quella proposta in figura (7.10).

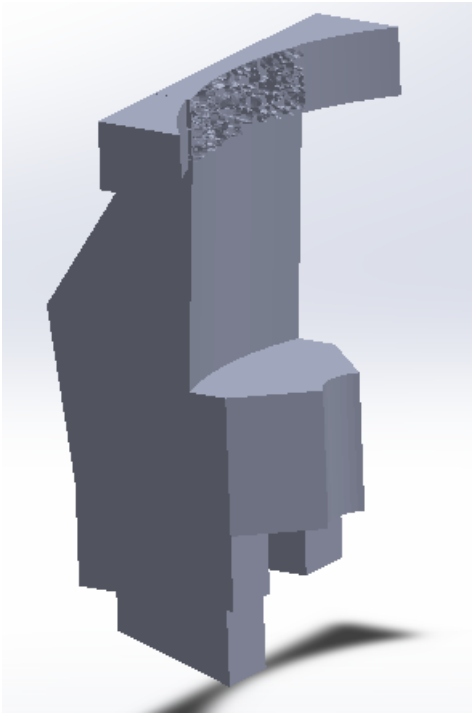

Figura 7.10: Modello CAD del gripper destro.

In questa versione, la parte centrale era destinata ad afferrare lo stantuffo. Il cappuccio veniva avvolto completamente dalla parte superiore. Seppur funzionante, è stata scartata: il braccio destro, oltre a sostenere il peso dei gripper e degli oggetti manipolati, deve sostenere la telecamera. Per evitare problemi dovuti al carico, è stata realizzata una seconda versione, ottimizzata, dello stesso dito.

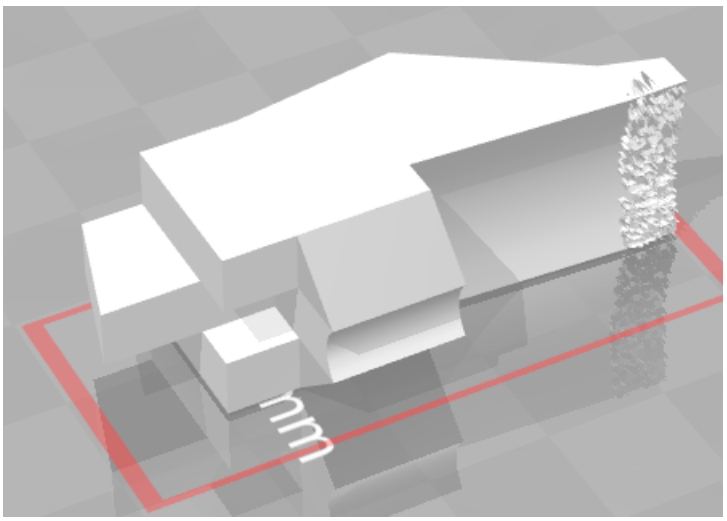

Figura 7.11: Modello CAD ottimizzato del gripper destro.

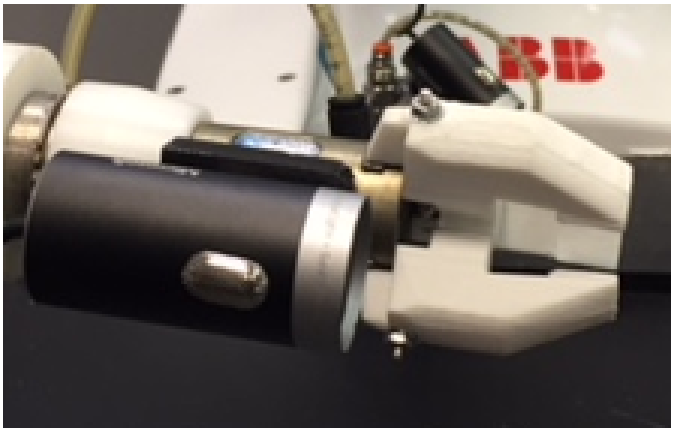

Figura 7.12: Gripper destro completo.

7.2 Stima del centro di gravità e dei momenti d'inerzia

L'algoritmo di controllo utilizzato durante l'operazione di inserimento e durante l'avvitamento, usa due strategie durante l'operazione: controllo di impedenza e controllo di forza. Il vettore delle forze generalizzate agenti sull'end effector viene stimato ed è fondamentale per una corretta implementazione del controllo. Questa stima, però, potrebbe risultare incorretta a causa della presenza dei gripper, degli oggetti o di qualunque elemento di supporto. La conseguenza riguarda la possibile misura incorretta di forze: potrebbero venire analizzate forze non dovute al contatto tra gli oggetti. Per evitare questo, vanno calcolati i tensori d'inerzia e il centro di gravità di tutti i corpi presenti tra l'end effector del manipolatore e l'oggetto manipolato. Questa operazione viene effettuata sul braccio destro, che è destinato a compiere più operazioni. Ricordando che nessun sensore di forza è utilizzato, questa operazione viene svolta tramite l'ausilio dello stesso software per il disegno CAD, SolidWorks.

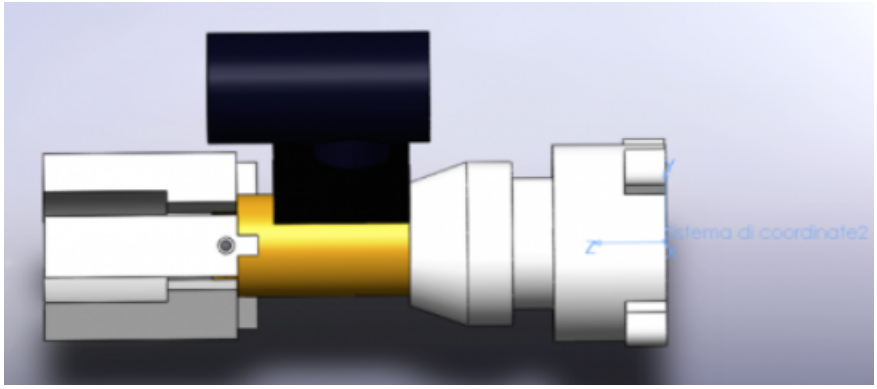

Figura 7.13: Modello CAD del gripper destro completo, vista laterale.

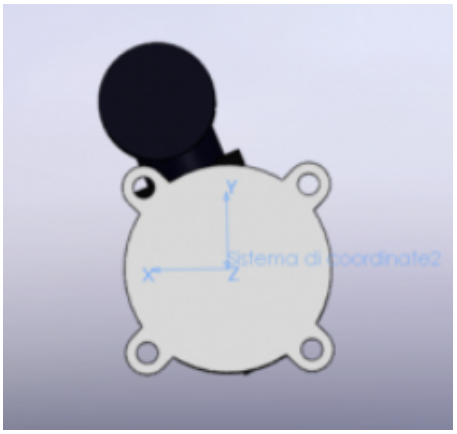

Figura 7.14: Modello CAD del gripper destro completo, vista posteriore.

Un tool di questo software permette di ottenere i valori corretti dei centri di gravità e calcolare i tensori di inerzia associati a una terna di riferimento. Questa operazione è necessaria per calcolare le forze e i momenti agenti in un punto di interesse. Dopo aver compiuto questa procedura, i valori ottenuti possono essere compensati mandando un comando al controllore esterno tramite OPCOM. Tutte le forze e i momenti stimati sono rifiutati se i corpi non sono a contatto.

7.3 Macchina a stati

In questa sezione verrà presentata la macchina a stati realizzata per una corretta e precisa descrizione del processo. Con macchina a stati viene intesa una sequenza di operazioni ben separate le une dalle altre, necessarie per completare l'operazione. La macchina a stati è stata realizzata in un codice Matlab, implementando tutte le strategie di controllo discusse in precedenza.
Le ipotesi da tenere in considerazioni sono molteplici, prima della descrizione della macchina a stati:

- oggetti manipolati considerati come corpi rigidi;
- robot in posizione adeguata: se il robot fosse in una posizione scorretta, potrebbe eseguire il suo compito, ma potrebbe collidere con l'ambiente circostante;
- modello di attrito non trascurabile: poichè la forma geometrica degli oggetti è particolare, bisogna tenere in considerazioni che potrebbero verificarsi condizioni di forte attrito. In caso questo accadesse, il manipolatore potrebbe entrare in uno stato di system failure, portandolo all'arresto;
- gripper saldamente chiusi attorno agli oggetti manipolati: piccoli errori nella posizione o nell'orientamento degli oggetti manipolati potrebbero portare ad un'inadeguata soluzione del compito.

Supponendo come condizione iniziale che il robot abbia già afferrato il corpo della pipetta nel gripper sinistro e lo stantuffo della pipetta nel gripper destro, la macchina a stati può essere così descritta:

1. Il primo stato è utilizzato per portare il robot in una determinata posizione iniziale. Ricordando i principi di ottimizzazione descritti nell'introduzione, la posizione iniziale e i vincoli imposti portano il robot ad operare in diverse posizioni dello spazio. Modificare la posizione iniziali permette di compiere l'operazione evitando particolari zone dell'ambiente di lavoro, magari occluse dalla presenza di altri oggetti. L'algoritmo di generazione della traiettoria è in grado di generare una traiettoria che il controllore esterno traduce in una posizione di riferimento che il manipolatore raggiunge. (Fig. 7.15)

Figura 7.15: Posizione iniziale.

2. Il secondo stato è utilizzato per l'allineamento degli assi. Il robot viene posto in una posizone tale per cui l'esecuzione dello stato successivo è facilitata. Vengono calcolati posizione e orientamento relativi tra braccio destro e braccio sinistro. Lo stantuffo viene posizionato a una distanza lungo l'asse z minima, rispetto al corpo della pipetta. Ricordando l'ipotesi che durante l'operazione di Inserimento gli oggetti manipolati possono essere considerati cilindrici, per semplificare l'esperimento, si cerca un allineamento anche tra gli altri assi (x, y, ϕ_x, ϕ_y, ϕ_z). Eventuali errori nell'allineamento di questi assi, vengono corretti con il controllo di impedenza. (Fig 7.16)

Figura 7.16: Allineamento degli assi.

In particolare viene utilizzata la posizione:

$$X^{target}_{position} = [0 \;\; 0 \;\; 0.005 \;\; 0 \;\; 0 \;\; 0]^T$$

Le distanze lungo x, y e gli assi ϕ_x, ϕ_y e ϕ_z sono poste a 0[mm]. La distanza lungo l'asse z è posta a 5[mm]

3. Il terzo stato è utilizzato per l'operazione di Inserimento. Viene fornita una velocità di riferimento lungo l'asse z. Lo stantuffo si infila nel corpo finchè non raggiunge il fondo di questo. Come anticipato, un controllo di impedenza è implementato per correggere eventuali errori di allineamento. Nel caso in cui non ci siano errori, il controllo di impedenza non modifica la traiettoria di riferimento; in caso contrario il controllo di impedenza, può modificare la traiettoria da compiere sulla base delle forze e dei momenti stimati. Lo stato termina nel momento in cui la forza stimata lungo l'asse z supera una determinata soglia, che non deve essere troppo elevata per evitare lo stato di system error. (Fig. 7.17)

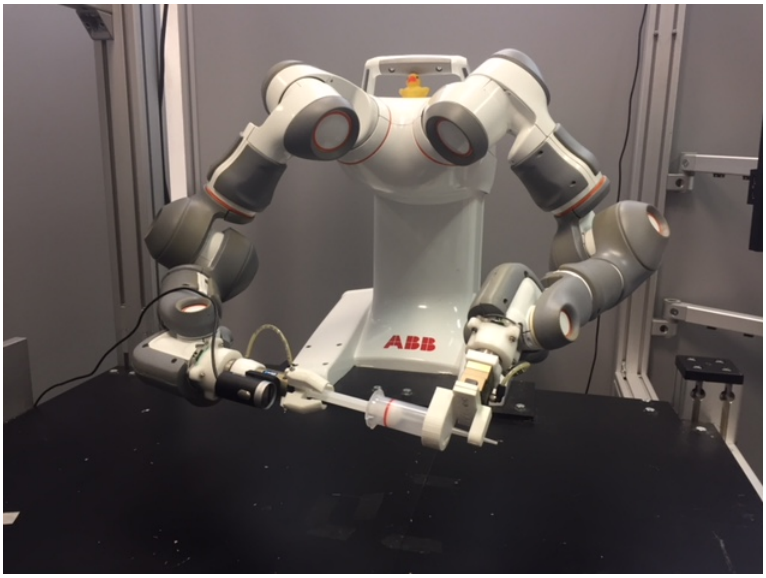

Figura 7.17: Inserimento.

In questo esperimento non vengono effettuati disallineamenti manuali. Viene settata la velocità di movimento lungo l'asse z a 10 [mm/s].

$$X_{velocity}^{target} = [0 \;\; 0 \;\; 0.01 \;\; 0 \;\; 0 \;\; 0]^T$$

Il processo è completo quando la forza stimata lungo l'asse z è maggiore di una determinata soglia F_{thr_z}. In particolare:

$$F_{thr_z} = 4\,[N]$$

4. Il quarto, il quinto, il sesto stato sono tutti stati di transizione. Dopo aver finito l'operazione di Inserimento, viene aperto il gripper destro. Lo stantuffo rimane incastrato nel corpo della pipetta, mentre il braccio destro del manipolatore si allontana, raggiungendo una posizione di sicurezza.

5. Il settimo stato è utilizzato per l'allineamento della telecamera con il cappuccio della pipetta. Dalla posizione di sicurezza precedentemente raggiunta, la telecamera viene posizionata in modo tale da guardare la parte superiore del cappuccio della pipetta, posizionato sul tavolo di lavoro.

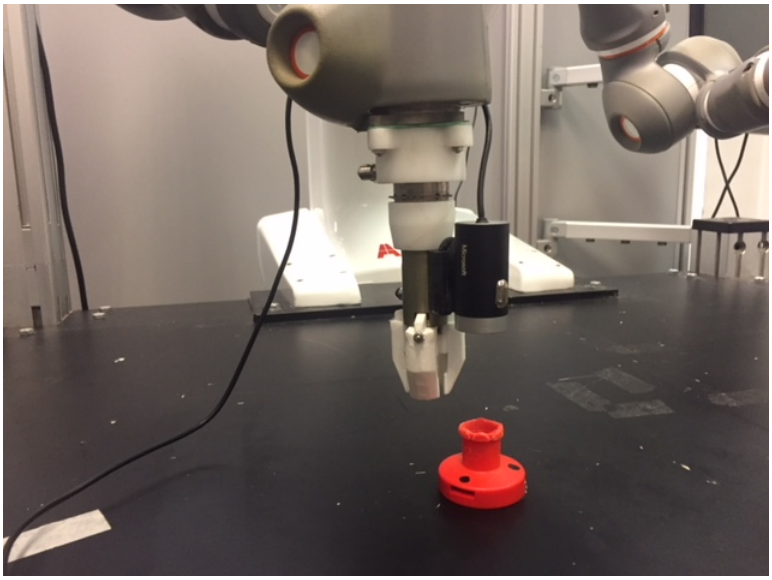

Figura 7.18: Allineamento telecamera-cappuccio.

6. L'ottavo stato è utilizzato per il calcolo dell'orientamento del cappuccio della pipetta. Tramite le funzioni di visione vengono calcolate le posizione dei segnalatori posti sull'oggetto per ricavarne l'orientamento relativo rispetto ad una terna ideale. I valori calcolati sono α_1 che esprime lo sfasamento in gradi e β_1 che determina la direzione dell'errore. β_1 può assumere i valori -1 e +1.

7. Il nono, il decimo e l'undicesimo stato sono utilizzati per l'afferraggio del cappuccio della pipetta. Il braccio destro raggiunge la posizione desiderata per l'afferraggio, il gripper destro viene chiuso e viene raggiunta una posizione di sicurezza.

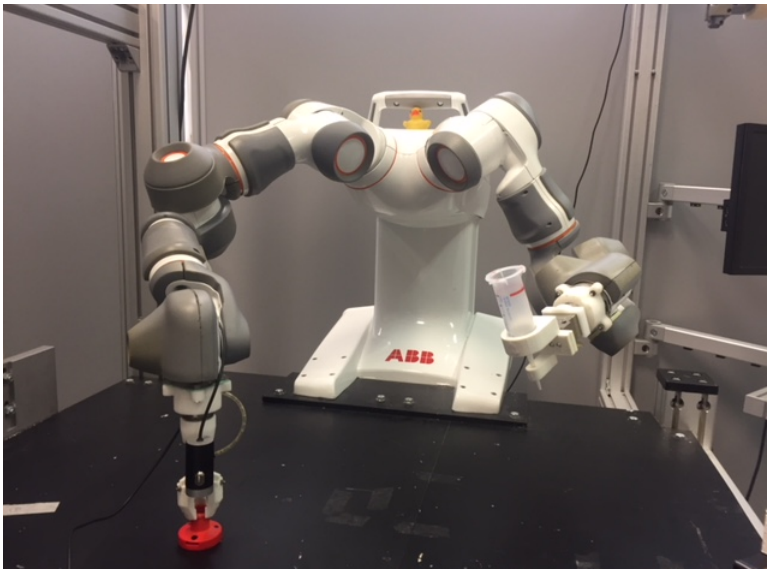

Figura 7.19: Transizione. Preparazione all'afferraggio del cappuccio.

8. Il dodicesimo stato è utilizzato per l'allineamento della telecamera con il corpo della pipetta. Similmente al secondo stato, viene fornita una posizione di riferimento al robot tale per cui la telecamera montata sul braccio destro del robot vada ad allinearsi con il corpo della pipetta. Rispetto al secondo stato, bisogna fornire un offset maggiore nella direzione z per due motivi:

   - nel corpo della pipetta è stato inserito lo stantuffo. Come si può notare dalla figura (7.20), la geometria dell'oggetto manipolato dal braccio sinistro è cambiata rispetto al caso precedente. Mantenendo gli stessi valori di allineamento, pur avendo cambiato la matrice che descrive la posizone dell'oggetto manipolato rispetto alla terna dell'end effector ($T_{R_{tool}}$), porterebbe ad una collisione tra gli oggetti manipolati. Questo provocherebbe l'ingresso del robot nello stato di system failure.
   - l'utilizzo della telecamera per l'azione di controllo richiede l'acquisizione di un'immagine nitida, al fine di ridurre al minimo gli errori. Nel caso in cui la telecamera fosse troppo vicina al corpo della pipetta, si rischierebbe di ottenere un'immagine distorta. Questo potrebbe portare ad un'errata azione di controllo.

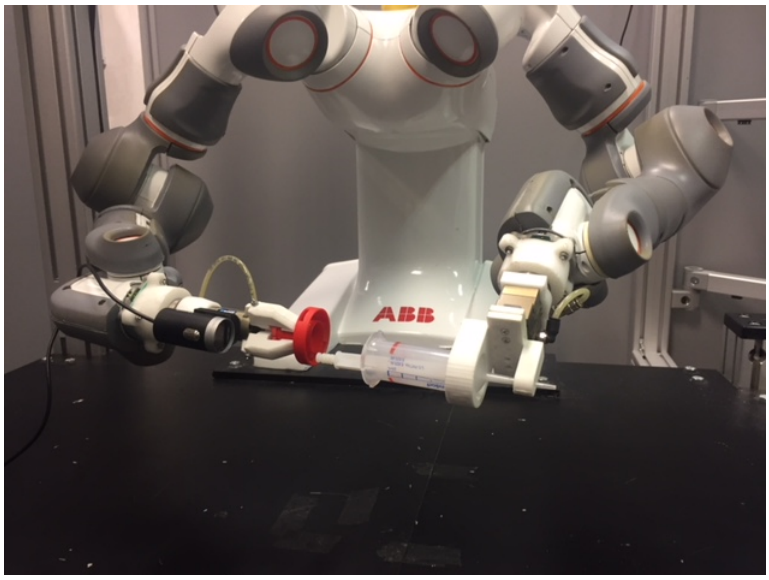

Figura 7.20: Allineamento telecamera-corpo.

9. Il tredicesimo stato è utilizzato per il calcolo dell'errore nel posizionamento del corpo della pipetta. Come accennato in precedenza, il cappuccio e il corpo della pipetta per incastrarsi tra loro devono essere allineati correttamente non solo lungo la direzione z, ma anche lungo le direzioni x e y. Vengono calcolati l'angolo α_2 che determina l'errore di orientamento e il valore β_2 che determina il verso di rotazione. α_2 viene espresso in gradi, β_2 può assumere i valori $+1$ e -1.

10. Il quattordicesimo stato è utilizzato per il corretto allineamento del cappuccio con il corpo della pipetta. Durante il quattordicesimo stato avviene un allineamento classico, come gli allineamenti precedentemente analizzati. Da ricordare che la geometria dell'oggetto sinistro è diversa rispetto alla precedente: lungo la direzione z bisogna tener conto della presenza dello stantuffo della pipetta, incastrato nel corpo. Di conseguenza va fornito un offset maggiore rispetto a prima, pari a 3 [cm]. Viene dunque utilizzata la posizione:

$$X_{position}^{target} = [0 \;\; 0 \;\; 0.03 \;\; 0 \;\; 0 \;\; 0]^T$$

11. Il quindicesimo stato è utilizzato per la correzione dell'errore di orientamento. Viene effettuata una rotazione lungo l'asse z in modo tale da ottenere il corretto allineamento relativo. In particolare viene fornita la posizione obiettivo:

$$X_{position}^{target} = \left[0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; \alpha \cdot \tfrac{\pi}{180}\right]^T$$

α viene calcolato come:

$$\alpha = \alpha_2 * \beta_2 - \alpha_1 * \beta_1$$

12. Il sedicesimo e il diciassettesimo stato sono utilizzati per l'operazione di Avvitamento.

    - Durante il sedicesimo stato avviene uno pseudo-inserimento: viene applicata una velocità target lungo l'asse z e un controllo di impedenza agisce affinchè non ci siano movimenti indesiderati. Questo movimento viene eseguito finchè non viene misurata una forza lungo l'asse z superiore ad una determinata soglia. Questa misura di forza coincide con il raggiungimento del corpo della pipetta da parte del cappuccio. La velocità impostata è pari a 1 [mm/s].

      $$X_{velocity}^{target} = [0 \;\; 0 \;\; 0.001 \;\; 0 \;\; 0 \;\; 0]^T$$

      L'operazione di pseudo-inserimento è conclusa nel momento in cui viene superata una soglia di forza F_{thr_z}. In particolare:

      $$F_{thr_z} = 0.5\,[N]$$

    - Durante il diciassettesimo stato viene fornita una velocità angolare target attorno all'asse z. Grazie all'allineamento effettuato al passaggio precedente, il movimento di rotazione da compiere è noto. Viene utilizzato un controllo di forza per mantenere i due corpi a contatto. La forza lungo l'asse z viene controllata in modo tale che rimanga tra due valori soglia f_{min} e f_{max}. In particolare:

      $$f_{min} = 0.3\,[N]$$
      $$f_{max} = 0.5\,[N]$$

Viene imposta una rotazione lungo l'asse z pari a 0.1 [rad/s].

$$X_{velocity}^{target} = [0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0.1]$$

Il processo è completo quando viene raggiunta una soglia di momento M_{thr_z} da parte del momento stimato lungo l'asse ϕ_z. In particolare:

$$M_{thr_z} = 0.5\,[Nm]$$

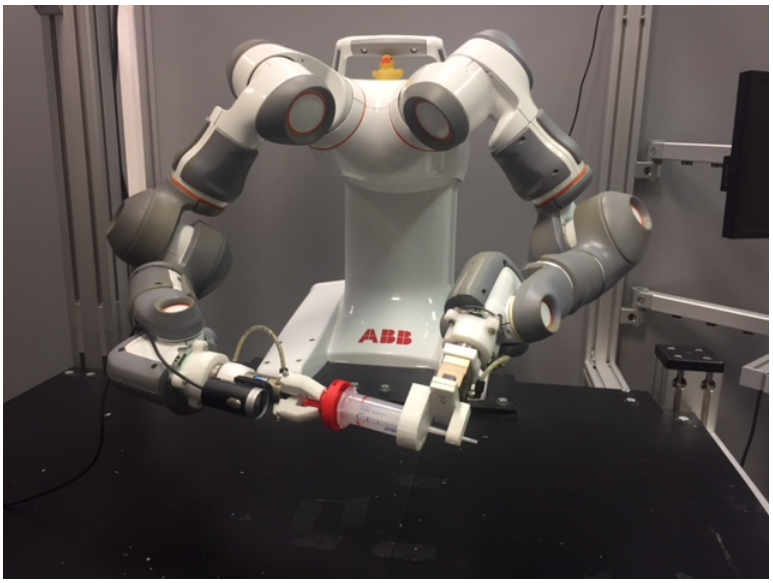

Figura 7.21: Avvitamento.

13. Il diciottesimo, il diciannovesimo e il ventesimo sono stati di transizione coincidenti con la fine dell'esperimento. Viene aperto il gripper destro, le braccia si allontanano tra loro ed infine raggiungono una posizione di sicurezza.

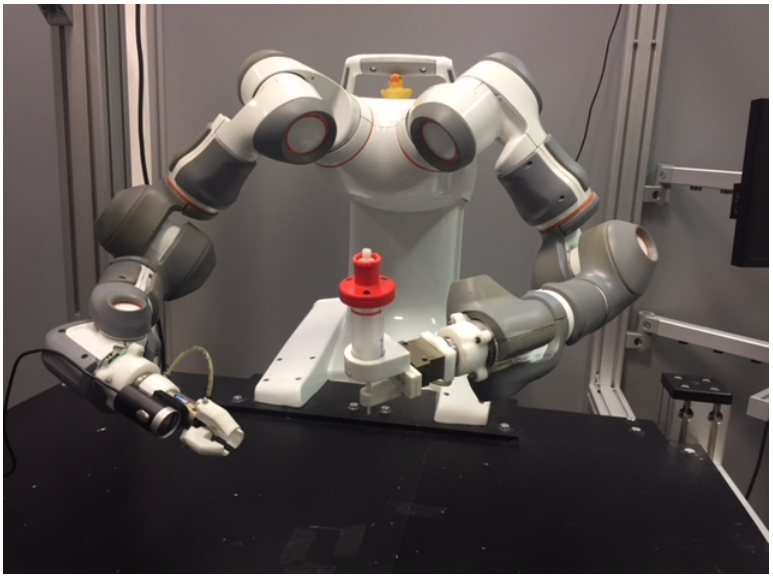

Figura 7.22: Fine esperimento.

7.4 Simulazione e Esperimenti

7.4.1 Simulazione

Il processo di montaggio della pipetta è stato implementato usando una funzione rappresentante la macchina a stati in Matlab e uno schema Simulink che permettesse la comunicazione con il controllore esterno.
Per ottenere la stima delle forze e dei momenti agenti sugli oggetti manipolati sono state utilizzate funzioni Matlab, poiché sarebbe stato impossibile ottenere la stima reale in ambito simulativo. Queste funzioni riportano in output un vettre di forze basati sulla stima della cinematica relativa e sullo Jacobiano analitico.
Per simulare l'ottimizzazione vincolata è stata prima utilizzato $qpOASES$, specificando la cinematica e i vincoli di forza nella forma richiesta dal simulatore. Successivamente è stato utilizzato $SOTH$. Questo passaggio è stato necessario per l'implementazione in ambito sperimentale: poichè il calcolo computazionale da parte di $qpOASES$ è troppo elevato, $SOTH$ è necessario per l'elaborazione di esperimenti real time. Analizzando la differenza di comportamento tra i due solutori, è stato possibile effettuare una corretta implementazione.
I processi di Inserimento e Avvitamento sono stati testati separatamente.

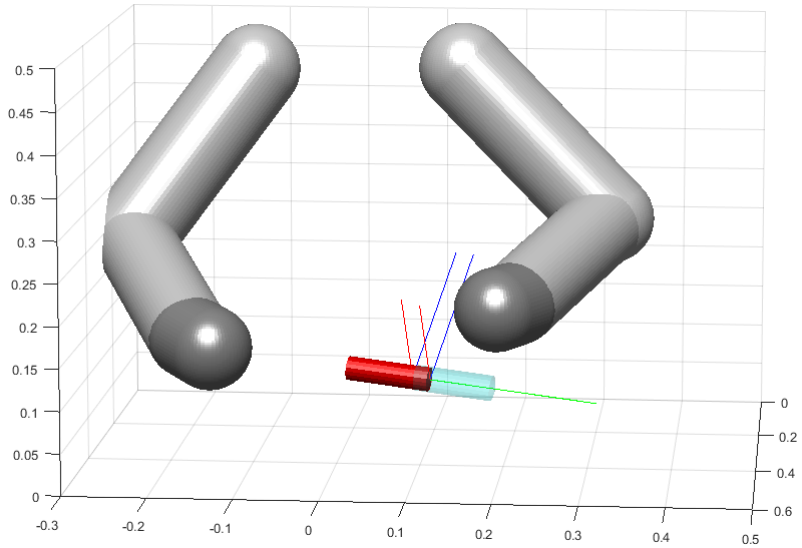

Figura 7.23: Simulazione dell'operazione di Inserimento.

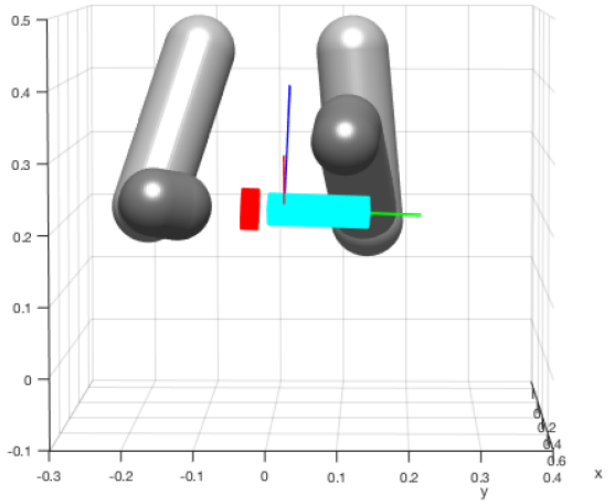

Figura 7.24: Simulazione dell'operazione di Avvitamento.

L'implementazione dell'algoritmo basato sulla visione è stato implementato, in ambito simulativo, sfruttando degli angoli fittizzi, poichè impossibile ottenere informazioni sulla rotazione relativa tra gli oggetti manipolati se non in ambito sperimentale.

7.4.2 Risultati sperimentali

In questa ultima sezione verranno analizzati i risultati sperimentali relativi all'implementazione dell'algoritmo di controllo basato sulla visione. In particolare verrà analizzato un esperimento in cui è stata fornita una posizione iniziale scelta in un insieme di valori che non permettessero la corretta esecuzione del problema senza l'utilizzo della visione. Questa tipologia di esperimento è stata effettuata per dimostrare la correttezza e l'utilità dell'algoritmo di controllo basato sulla visione.
Per iniziare correttamente un esperimento vanno definiti i limiti di posizione, velocità e accelerazione di ogni giunto del robot. Questi valori sono stati forniti dal produttore. I valori delle velocità vengono mantenuti bassi durante l'intero processo per evitare stress eccessivi ai giunti del robot. La posizione iniziale è stata scelta manualmente sfruttando il *teach pendant* e riportando i valori nella funzione Matlab.

Esperimento con sfasamento di 25 gradi

Per dimostrare la correttezza nell'esecuzione delle operazioni da parte dell'algoritmo di controllo, vengono ora mostrati i risultati relativi ad un esperimento che senza controllo di visione avrebbe riportato un risultato finale errato.
Viene ora richiamato brevemente lo scopo dell'algoritmo di controllo.
L'algoritmo di visione viene utilizzato per correggere disallineamenti lungo gli assi x e y degli oggetti manipolati, in modo tale da garantire casualità alla posizione iniziale degli

oggetti.
In questo esperimento, i corpi verranno posizionati in modo tale da avere uno sfasamento di ≈25 gradi tra gli assi x e y, mentre lo sfasamento tra gli assi z sarà di 0 gradi.
Durante la prima parte dell'esperimento viene effettuata correttamente l'operazione di Inserimento.
Poichè avviene un allineamento pressochè perfetto tra le braccia del robot, il controllo di Impedenza non modifica la traiettoria da seguire. In Fig. (7.25) viene riportato il grafico della forza lungo l'asse z.

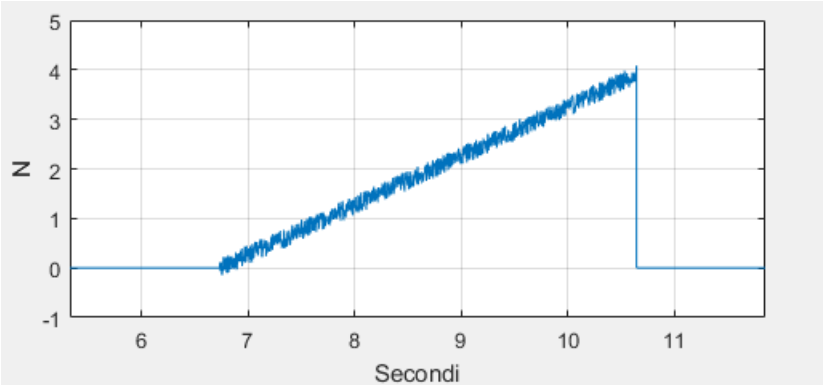

Figura 7.25: Forza riportata lungo l'asse z durante l'Inserimento.

Come si può notare dal grafico, le forze vengono calcolate solamente durante le operazioni richieste, altrimenti vengono considerate zero. Il raggiungimento della soglia di forza prescelta indica la fine dell'operazione di Inserimento.
Successivamente, viene calcolato l'orientamento del cappuccio della pipetta e viene afferrato. In seguito avviene il calcolo dell'orientamento del corpo della pipetta con conseguente calcolo dell'errore di orientamento tra i due oggetti.
Dopo aver conosciuto l'errore, avviene la correzione di quest'ultimo, attraverso una rotazione ϕ_z. In questo modo è possibile procedere ad una corretta esecuzione dell'operazione di Avvitamento.
Le immagini seguenti evidenziano la correzione dell'errore di orientamento precedente la fase di avvitamento.

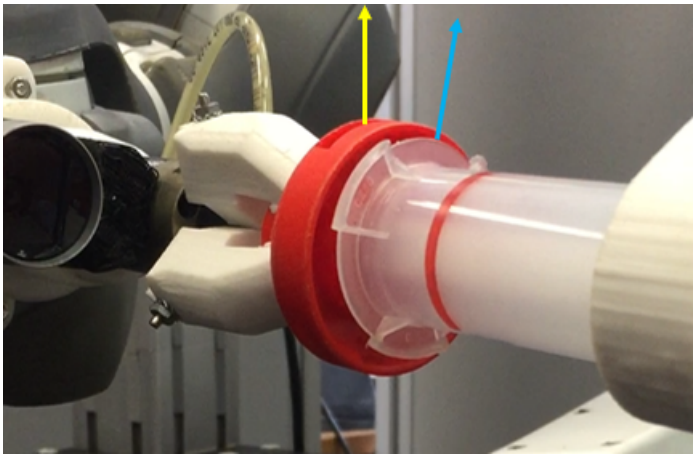

Figura 7.26: Orientamento prima della correzione.

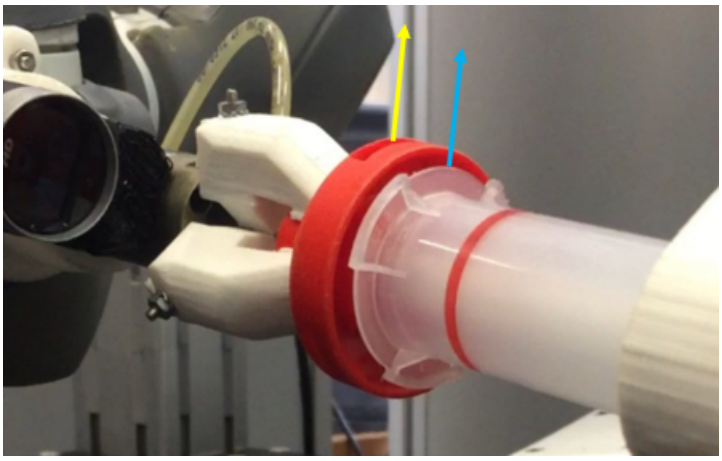

Figura 7.27: Orientamento dopo la correzione.

In giallo viene rappresentato l'asse x della terna costruita per il cappuccio della pipetta, in blu lo stesso asse per il corpo della pipetta.
Il seguente grafico riporta lo stesso dato.

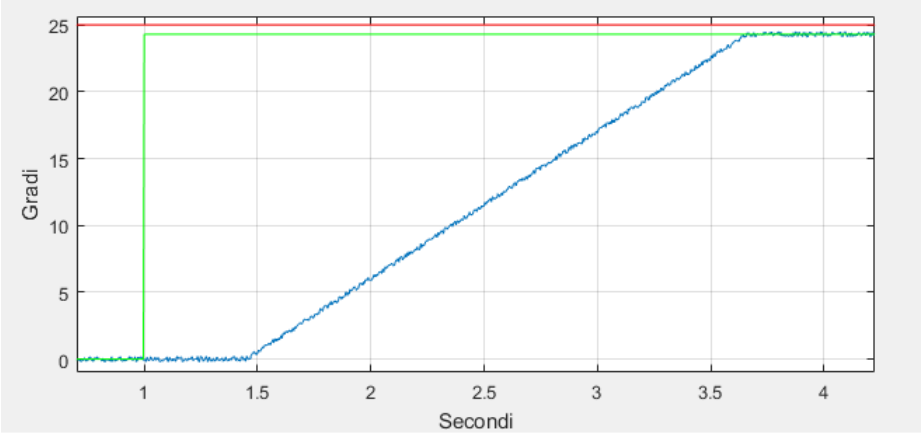

Figura 7.28: Errore nell'orientamento relativo.

In rosso viene mostrato l'errore effettivo introdotto, in verde viene mostrato l'errore calcolato, in blu il posizionamento relativo tra le braccia lungo l'asse ϕ_z.
Dal grafico si nota come l'errore calcolato sia diverso dall'errore effettivo. Questo, come discusso nel capitolo 6, dipende fortemente dalle immagini catturate. L'errore riportato tra la misura effettuata e la misura attesa può essere causato da più fattori. Ricordando che il calcolo dell'errore viene effettuato tramite la cattura di due immagini, il numero dei fattori di errore raddopia.
In particolare, l'errore effettivo è stato impostato a 25 gradi, mentre l'errore calcolato è pari a 24.3 gradi. Il tempo che intercorre tra il riconoscimento dell'errore (salita linea verde in Fig 7.27) e l'attuale moto (salita linea blu) è il tempo che intercorre tra lo snapshot che viene effettuato durante l'allineamento della telecamera con il corpo della pipetta e l'allineamento tra il cappuccio con il corpo, precedente l'operazione di correzione dell'errore. Questo tempo è pari circa a 1.41 secondi. Il tempo di correzione dell'errore dipende dai limiti impostati al movimento del robot e, in questo caso, è pari circa a 2.3 secondi.
L'errore tra l'angolo effettivo e l'angolo calcolato è pari a circa 0.7 gradi. Questo non crea però un problema poichè, per la natura degli oggetti manipolati, l'errore massimo tra le misure è accettato fino a 5 gradi.
L'esperimento si conclude con l'operazione di Avvitamento. Anche in questo caso il controllo di impedenza non modifica la traiettoria poichè l'allineamento effettuato è corretto.
Il controllo di forza mantiene la forza lungo l'asse z tra i valori desiderati, mentre il momento lungo ϕ_z viene calcolato.

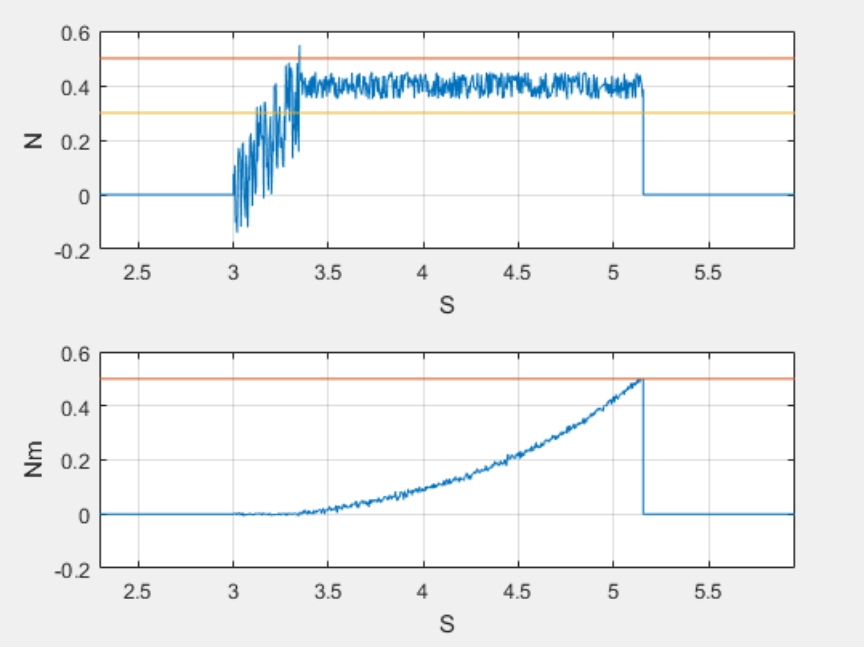

Figura 7.29: Forza lungo z e momento lungo ϕ_z durante l'Avvitamento.

Nello specifico, si può notare come durante la salita della forza, corrispondente allo

pseudo-inserimento, il momento venga calcolato, ma sia praticamente nullo. Dopo ≈0.4 secondi, finito l'Inserimento, inizia la fase di Avvitamento. La forza viene mantenuta tra i valori desiderati di 0.3 N e 0.5 N, il momento cresce. Appena viene superata la soglia di 0.5 Nm, l'operazione è conclusa.

Capitolo 8

Conclusioni e sviluppi futuri

Risultati ottenuti

Nel presente elaborato è stata sviluppata, testata e implementata con successo una tecnica per eseguire un'operazione di correzione di traiettoria in operazioni Dual Arm con un algoritmo di controllo basato sulla visione.
L'analisi e la formalizzazione eseguite sulle operazioni di tipo Dual Arm hanno permesso un'introduzione facilitata negli schemi di controllo ad una tipologia totalmente differente da quelle classiche.
Tutte le tecniche di controllo sono state utilizzate per l'operazione di avvitamento, mentre per l'operazione di inserimento è stato utilizzato solamente il controllo di impedenza. L'algoritmo di ottimizzazione vincolata permette l'esecuzione delle operazioni più complesse, il controllo di ammettenza permette la compensazione di alcuni errori di posizionamento, mentre l'algoritmo di visione fornisce ripetibilità e variabilità all'esecuzione degli esperimenti.
Durante le prove sperimentali non sono stati riportati gravi problemi, fatta eccezione per il calcolo computazionale richiesto dall'algoritmo di ottimizzazione, qualora fossero fissati troppi vincoli, o dall'algoritmo di visione.

Sviluppi futuri

Con questa tesi è stato proposto una nuova tipologia di controllo basata sulla visione.
Il metodo di controllo proposto propone una correzione in anello aperto della traiettoria, andando a modificare la posizione obiettivo. Questa metodologia di approccio è stata pensata come soluzione più semplice per la soluzione di un problema finalizzato all'operazione di Avvitamento. Un possibile sviluppo futuro potrebbe prevedere un'implementazione di controllo in anello chiuso, Visual Servoing. Con questa tipologia di controllo non verrebbe effettuata solamente una correzione della traiettoria, ma verrebbe controllato completamente il moto.

Come ulteriore possibile sviluppo, viene proposta una tipologia totalmente differente di problema. Nel capitolo 4 erano state accennate tutte le problematiche relative alla corretta implementazione dell'algoritmo di controllo basato sulla visione. Tra queste vengono prese in considerazione la problematica legata alla taratura dei parametri e le problematiche legate alla cattura dell'immagine. Possibile sviluppo su questo argomento, la creazione di un sistema di controllo con taratura automatica dei parametri, che permetta il corretto riconoscimento di tutte le features desiderate. Definendo opportune funzioni di costo si potrebbe creare un algoritmo di controllo che vada a tarare automaticamente le soglie del filtro di Canny e i parametri della trasformata di Hough.

Bibliografia

[1] Young-Loul K., Byeong-Sang K., Jae-Bok S. ''Hole detection algorithm for square peg-in-hole using force-based shape recognition". *IEEE International Conference on Robotics and Automation*, 2012.

[2] Parigi Polverini M., Zanchettin A., Castello S., Rocco P. ''Sensorless and Constraint Based Peg-In-Hole Task Execution with Dual-Arm Robot". *IEEE International Conference on Robotics and Automation*, 2016.

[3] Incocciati F., Zanchettin A.M., Parigi Polverini M., Rocco P. "Robust Constraint-Based Robot Control for Bimanual Cap Rotation". *IEEE International Conference on Intelligent Robot and Systems*, 2017.

[4] Zanchettin A.M., Rocco P. ''Reactive motion planning and control for compliant and constraint based task execution". *IEEE International Conference on Robotics and Automation*, 2015.

[5] Siciliano, Khatib. ''Handbook of Robotics". *Springer*, 2008.

[6] Uchiyama M., Dauchez P. ''A symmetric hybrid position force control scheme for the coordination of two robots". *IEEE International Conference on Robotics and Automation*, 1988.

[7] Nazrul H. A., Mahzan T. ''Grasp analysis using Multi-Fingered manipulation of objects". *IEEE International Conference on Systems, Process and Control*, 2015.

[8] Balletti L., Rocchi A., Belo F., Catalano M., Garabini M., Grioli G., Bicchi A. ''Towards variable impedance assembly: The VSA peg-in-hole". *IEEE International Conference on Robotics and Automation*, 2012.

[9] Ballard D.H., Brown C.M. ''Computer Vision". 1982.

[10] Huang T., Vandoni C. ''Computer Vision: Evolution and Promise". 1996.

[11] Jahne B., Haussbecker H. ''Computer Vision and Applications, a guide for students and pratictioners ". 2000.

[12] Turek F. ''How to make a robot see". 2011.

[13] Szeliski R. ''Computer Vision: Algorithms and Applications". 2010.

[14] Bruyninckx H., Dutre S., De Schutter J. ''Peg-in-Hole: a model based solution to peg and hole alignment". *IEEE International Conference on Robotics and Automation*, 1995.

[15] De Schutter J., Smits R., De Laet T. "iTasc: a Tool for Multi-Sensor Integration in Robot Manipulation". *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 2008.

[16] De Schutter J. De Laet T., Rutgeerts J. "An Application of Constraint based Task Specification and Estimation for Sensor based Robot System in the presence of geometric uncertainty". *IEEE International Conference on Intelligent Robot and Systems*, 2007.

[17] Decre W., Bruyninckx H., De Schutter J. "Extending the iTasc Constraint based Robot Task Specification Framework to time independent trajectories and User configurable Task horizon". *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[18] Tazaki Y., Suzuki T. "Constraint based Prioritized Trajectory planning for multibody Systems". *IEEE Transaction on Robotics*, 2014.

[19] Kermorgrant O., Chaumette F. "Dealing with Constraint in sensor based robot control". *IEEE Transaction on Robotics*, 2014.

[20] Kroger, Wahl M. "Online Trajectory Generation: basic concepts for instantaneous reactions to unforseen events". *IEEE Transaction on Robotics*, 2010.

[21] Blanchini F. "Set Invariance in Control". *Automatica*, 1999.

[22] Samson C.E. "Robot control: the task function approach". 1991.

[23] Siciliano B:, Slotine E. ''A general framework for managing multiple tasks in highly redundant robotic systems ". *IEEE International Conference on Advanced Robotics*, 1991.

[24] Nakamura Y., Hanafusa H. "Task Priority based control of robot manipulators". *International Journal of Robotics*, 1987.

[25] Hollerback M., Suh K. "Local versus global torque optimization of redundant manipulators". *International Conference on Robotics and Automation*, 1987.

[26] Rajni V. P., Heidar A. T. ''A robust Position and Force Control Strategy for 7-DOF Redundant Manipulators". *IEEE Transaction on Mechatronics*, 2009.

[27] Ferreau H.J., Potschka A., Kirches C. ''QPOASES webpage ". *http://www.qpOASES.org*.

[28] Ferreau H.J, Bock H.G., Diehl M. ''An online active set strategy to overcome the limitations of explicit MPC ". *International Journal of Robust and Non Linear Control*, 2008.

[29] Escande A., Mansard N., Wieber P.B. ''Hierarchical Quadratic Programming: fast online Humanoid-robt motion generation". *International Journal of Robotic Research*, 2014.

[30] Varano P., Casciola G. ''Elaborazioni di Immagini con la Libreria OpenCV". 2008-2009.

[31] Atherton T.J., Kerbyson D.J. ''Size invariant circle detection". 1999.

[32] Yuen H.K., Princen J., Illingworth J., Kittler J. ''Comparative study of Hough transform methods for circle finding". 1990.

[33] Davies E.R. ''Machine Vision: Theory, Algorithms, Practicalities". 2005.

[34] De Luca A., Mattone R. ''Actuator failure detection and isolation using generalized moments". *IEEE International Conference on Robotics and Automation*, 2003.

[35] De Luca A., Mattone R. ''Sensorless robot collision detection and hybrid force/motion control". *IEEE International Conference on Robotics and Automation*, 2005.

[36] Farid Al B., Lampaert V. ''The Generalized Maxwell-Slip Model: A Novel Model for Friction Simulation and Compensation". *IEEE Transactions on Automatic Control*, 2005.

[37] Jinoh L., Pyung Hun C., Jamisola S. ''Relative Impedance Control for Dual-arm Robots Performing Asymmetric Bi-manual Tasks". *IEEE Transactions on Industrial Electronics*, 2014.