POLITECNICO DI TORINO

Corso di Laurea Magistrale
in Ingegneria Meccanica

Tesi di Laurea Magistrale

Algoritmi di localizzazione dell'operatore umano per applicazioni di robotica collaborativa

Relatori
prof. Stefano Mauro
prof. Stefano Paolo Pastorelli

Candidato
Michael Scarcia

A.A. 2017/2018

RINGRAZIAMENTI

Innanzitutto, desidero ringraziare le persone che hanno lavorato con me ed hanno permesso lo sviluppo di questo lavoro di tesi dimostrando professionalità, impegno e dedizione al lavoro. Per questo ringrazio il Prof. Stefano Mauro ed il Prof. Stefano Pastorelli. Ringrazio, inoltre, i dottorandi Leonardo Scimmi e Matteo Melchiorre che hanno contribuito a creare un ambiente di lavoro e ricerca pieno di stimoli ed idee costruttive.

Ringrazio Pierpaolo, Fiorenzo e Matteo con cui ho condiviso questo percorso negli ultimi mesi ricchi di confronti e di scambi di idee.

Un ringraziamento speciale va a tutte quelle persone che hanno reso la mia permanenza a Torino in questi due anni universitari un'esperienza piacevole, divertente e piena di nuove scoperte.

Ringrazio i miei genitori e ringrazio mio fratello, perché se non fosse per loro non sarei la persona che sono oggi.

Ringrazio gli amici che ci sono stati durante questo percorso e che sempre ci saranno anche a più di un migliaio di chilometri di distanza, quelli che sono lì quando ritorni a casa e quelli che ti fanno davvero sentire a casa.

Torino, 25/10/18

Michael

SOMMARIO

INDICE DELLE FIGURE

INDICE DELLE TABELLE

INTRODUZIONE

Lo sviluppo tecnologico e l'avvento della quarta rivoluzione industriale spingono sempre maggiormente verso un ambiente ad elevata automazione e intercomunicazione tra uomo e macchine.

Lo scopo della presente tesi di laurea è analizzare alcuni particolari aspetti di quella che viene comunemente definita robotica collaborativa, ovvero, della cooperazione tra uomo e dispositivi robotici negli ambienti industriali più moderni e sviluppati. L'introduzione dei robot in ambiente industriale risale ormai a diverse decine di anni fa, ma, recentemente ci si è posti l'obiettivo di far collaborare i più avanzati dispositivi robotici con operai specializzati per migliorare l'efficienza di tutte quelle operazioni che vengono, ancora oggi, eseguite separatamente dall'uomo e dalla macchina che talvolta comportano maggiori tempi e costi in un ambiente di produzione industriale.

Il seguente testo è articolato in sette capitoli. Nel primo capitolo viene effettuata una presentazione al tema della robotica collaborativa che costituisce il macro argomento in cui si vuole inserire questo lavoro di tesi. In particolare, in questa prima parte ci si sofferma sul quadro storico e normativo relativo all'introduzione e all'utilizzo dei robot, sono poi riportati alcuni cenni sulla cinematica dei robot e si presenta un'analisi delle varie applicazioni dei robot cinematicamente ridondanti. Infine, il capitolo si conclude con un'analisi dell'importanza dello sfruttamento della ridondanza dei manipolatori robotici nell'ambito dello sviluppo di efficaci ed efficienti algoritmi di anticollisione per una sicura cooperazione dei robot con controparti umane.

In seguito, nel capitolo numero 2, si analizzano gli strumenti analitici utilizzati per costruire un modello virtuale in ambiente Matlab del manipolatore KUKA LBR R820, tale procedura applicata a questo particolare robot dotato di 7 gradi di libertà è estendibile per qualsiasi manipolatore. In particolare, si introduce la convenzione di Denavit-Hartenberg standard, le caratteristiche geometriche e inerziali del manipolatore selezionato, la fase di pianificazione delle traiettorie e la procedura per calcolare le coppie erogate dai motori dei giunti.

Nel terzo capitolo, si prosegue con l'analizzare alcune delle più note tecniche di anticollisione in ambito robotico. In particolare, si analizzano le tecniche proposte nell'ultimo decennio dai ricercatori dell'Università Sapienza di Roma (A. De Luca e collaboratori) e dal gruppo di ricerca del Politecnico di Milano (P. Rocco e collaboratori) inerenti ad efficaci algoritmi di anticollisione basati sul monitoraggio dell'ambiente di lavoro mediante dispositivi ottici e l'implementazione di algoritmi di controllo per l'anticollisione che sfruttano, tra le altre cose, la ridondanza cinematica dei bracci robotici.

Successivamente, analizzato criticamente lo stato dell'arte e i fondamenti della cinematica dei dispositivi robotici, si passa all'implementazione in ambiente di sviluppo Matlab delle tecniche di collision avoidance descritte nel capitolo precedente e alla modifica e miglioramento di tali approcci per lo sviluppo di un nuovo ed efficace algoritmo di anticollisione. Sviluppati tali algoritmi si conducono una serie di prove in ambiente virtuale affiancando l'ambiente di programmazione Matlab e l'ambiente di visualizzazione di dispositivi robotici V-REP. Questa

operazione di validazione dell'efficacia degli algoritmi di anticollisione sviluppati è eseguita nel capitolo quarto.

Nel capitolo 5 si descrive l'operazione di modellazione cinematica di un manipolatore UR3, l'adattamento degli algoritmi di anticollisione a tale nuovo manipolatore, l'esecuzione di test in ambiente virtuale e si descrivono alcune prove eseguite in laboratorio presso il dipartimento DIMEAS del Politecnico di Torino con un robot fisicamente disponibile ed utilizzabile.

In seguito, ritornando all'ambiente virtuale sono stati testati diversi approcci per il miglioramento degli algoritmi implementati mediante l'utilizzo di forme più complesse per la rilevazione della figura umana avvalendosi di sensori di profondità Kinect.

Il testo termina con un capitolo di conclusione in cui è riassunto il lavoro eseguito, i risultati ottenuti e si propongono eventuali lavori futuri per approfondire alcune criticità incontrate e sviluppare idee nate nel corso del periodo di lavoro.

1 CAPITOLO 1: INTRODUZIONE ALLA ROBOTICA COLLABORATIVA

1.1 Introduzione alla robotica

La **robotica** è una scienza moderna e pluridisciplinare che unisce meccanica, elettronica e informatica e che sempre di più negli ultimi decenni sta rivoluzionando il nostro modo di lavorare, produrre e pensare. Ciononostante, la robotica ha delle radici culturali profonde. L'uomo da sempre aspira a riprodurre, usando delle macchine, i suoi movimenti ed il suo comportamento. Numerose prove si trovano nell'arte e nella letteratura come ad esempio la moderna storia del mostro di Frankenstein (1818) ispirata alla figura Prometeo, risalente all'antichissima mitologia greca.

Il termine **robot**, invece, è molto più moderno ed è stato coniato da Karel Čapek, autore del dramma utopico fantascientifico R.U.R. del 1920 (acronico di "Rossumovi Univerzální roboti" traducibile in italiano con "I Robot Universali di Rossum"). Il termine robot deriva dal ceco robota che significa lavoro forzato. Sebbene, la visione di robot di Mary Shelley (autrice di Frankenstein) e di Čapek è quella di automi con sembianze umane, in materiale organico e dotati di sentimenti l'immagine di robot come opera umana ma dalla composizione meccanica risale agli anni '40. È, infatti, Isaac Asimov, autore russo di fantascienza a coniare il termine **robotica** come la scienza che studia i robot, automi umanoidi dalla composizione meccanica soggetti al rispetto di tre leggi fondamentali:

> 1) "Un robot non può recare danno ad un essere umano né può permettere che a causa del proprio mancato intervento un essere umano riceva danno".
>
> 2) "Un robot deve obbedire agli ordini impartiti dagli esseri umani purché tali ordini non contravvengano alla prima legge".
>
> 3) "Un robot deve proteggere la propria esistenza purché questo non contrasti con la prima e la seconda legge".

Già queste idee anticipano l'idea di robot collaborativo sviluppata solo negli ultimi decenni. Infatti, prerogativa principale di tali cobot è che essi debbano eseguire il task loro imposto con l'attenzione di non recare danno a nessun operatore nelle loro vicinanze.

Quello che intendiamo **oggi** con il termine robotica fa riferimento a sistemi altamente complessi rappresentati da svariati sottosistemi. In particolare, i dispositivi robotici sono dotati di: un **sistema meccanico** costituito di organi di manipolazione e/o di locomozione, un **sistema di attuazione** (idraulica, elettrica, pneumatica ecc.) che permette il movimento degli organi meccanici, dei **sensori** (ottici, di pressione, piezoelettrici, piezoresistivi ecc.) che permettono di interfacciare il dispositivo con l'ambiente esterno ed infine di una **unità di governo** che riceve i dati dai sensori e permette di controllare gli attuatori che movimentano le parti meccaniche del robot come se costituisse il cervello della macchina. (1)

1.1.1 Struttura meccanica dei robot

Ad oggi, esistono diverse tipologie di robot, industriali e non, ciascuno di essi è caratterizzato da una struttura meccanica, da una componentistica elettronica e da una unità di controllo programmata.

I più comuni robot industriali si rifanno alla struttura di un braccio umano e le loro sezioni prendono il nome proprio dalle parti del corpo umano da cui traggono ispirazione. Infatti, un braccio robotico è caratterizzato da una **base** che può essere ancorata ad un supporto mobile (ad esempio un carrello con ruote) o fisso (ad esempio pavimento, parete o soffitto), da una prima articolazione che prende il nome di **spalla** (shoulder) e dai successivi gradi di libertà che si articolano in **gomito** (elbow) e **polso** (wrist) del robot terminando con un organo terminale comunemente chiamato **end effector** (comunemente abbreviato in EE) che può assumere svariate forme e funzioni (pinze, pistola di verniciatura, ventosa ecc.).

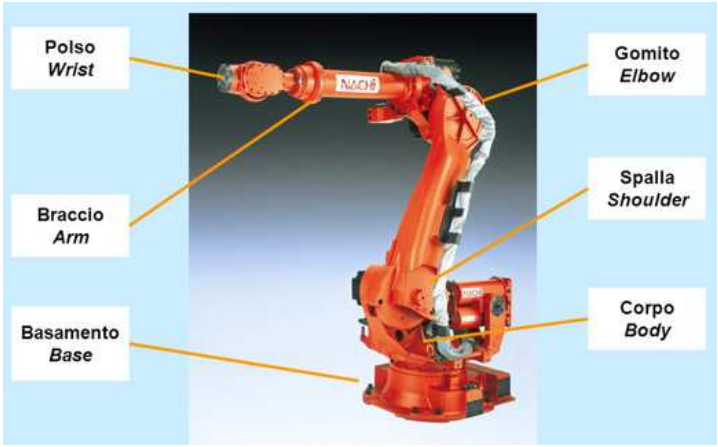


Figura 1 Nomenclatura struttura meccanica robot

Questa tipologia di dispositivi robotici è ovviamente solo una delle molteplici forme in cui i robot possono presentarsi ma è anche una tra le più comuni in ambiente industriale.

In modo prettamente meccanico un robot può anche essere valutato come una catena cinematica aperta caratterizzata da link e giunti. I **link** costituiscono il collegamento meccanico tra un giunto ed il successivo e possono essere di forma e dimensioni molto diverse ma possono, in prima approssimazione, essere visti come corpi rigidi descrivibili mediante specifiche caratteristiche geometriche ed inerziali. D'altro canto, i **giunti** hanno il duplice scopo di collegare due link successivi e permetterne un movimento meccanico relativo. In genere, i giunti dei bracci robotici possono essere di due tipologie: giunti prismatici che permettono una traslazione relativa dei due link collegati e giunti rotoidali che, invece, permettono una rotazione relativa tra i link.

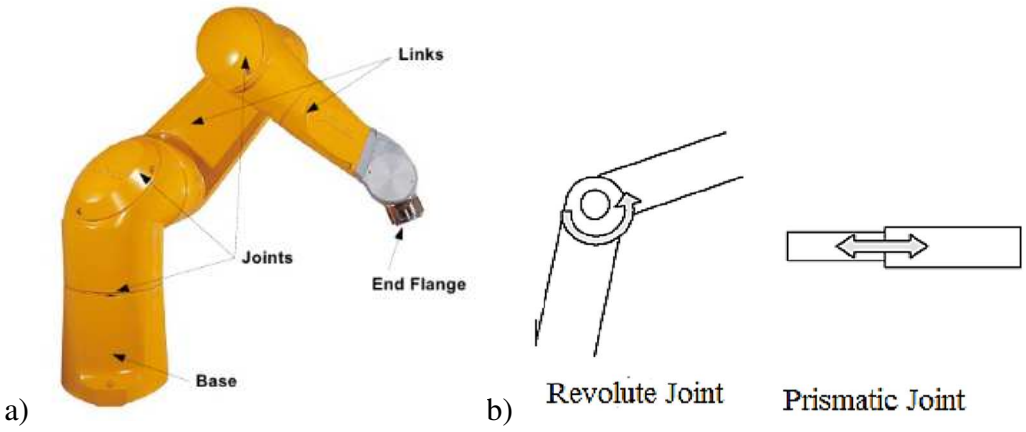


Figura 2 a) Schema giunti e link b) Distinzione giunti rotoidali e prismatici

1.1.2 Robotica industriale

I robot usati in ambito industriale sono tra i più disparati, una prima grande distinzione può essere fatta tra robot mobili e robot fissi. Concentrando l'attenzione sui robot con base fissa è possibile distinguere i più comuni robot industriali in base al tipo e al numero di gradi di libertà di cui sono dotati. Ogni tipologia di robot risulta poi applicabile in diversi ambiti per sfruttarne le caratteristiche peculiari.

Si passano ora in rassegna le più comuni strutture meccaniche dei robot industriali utilizzati attualmente. La più semplice è quella dei **robot a struttura cartesiana**. In questi robot i 3 gradi di libertà della struttura di base sono realizzati da coppie prismatiche. Questi robot vengono comunemente usati quando: è necessaria elevata precisione e ripetibilità di posizionamento, sono necessari elevati volumi di lavoro e vi è la necessità di lavorare su più stazioni di lavoro disposte su una stessa linea.

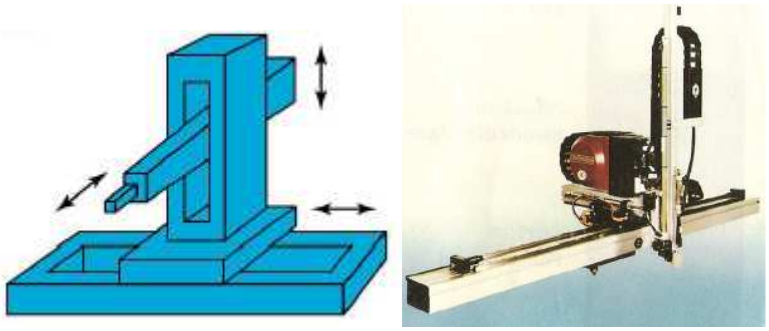


Figura 3 Robot con struttura cartesiana

Altra tipologia di robot industriale comunemente usata è la **struttura cilindrica**. I robot a struttura cilindrica sono caratterizzati da un giunto rotoidale e due prismatici, tale struttura è tipicamente usata per manipolazione e per asservimento di macchine utensili.

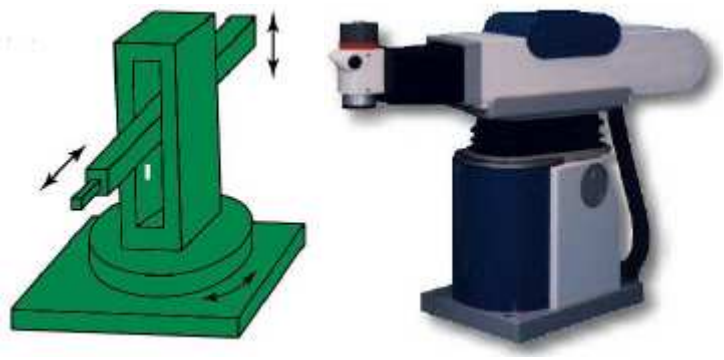


Figura 4 Robot con struttura cilindrica

Un'altra soluzione è quella di robot con **struttura sferica**. Tali robot sono dotati di due assi di rotazione e uno lineare, la struttura sferica si usa per l'asservimento di macchine utensili ma comunemente oggi è stata largamente sostituita dalla struttura antropomorfa.

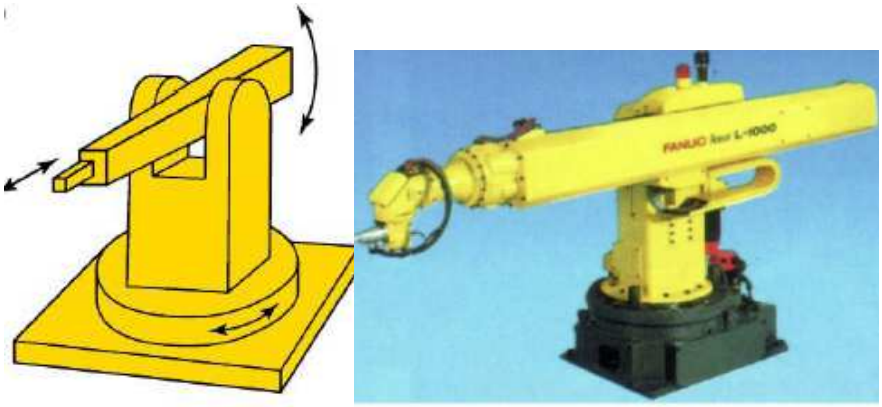


Figura 5 Robot con struttura sferica

La più comune tipologia di robot oggi però è quella che fa uso di una **struttura antropomorfa**. Infatti, questa è una struttura molto versatile e diffusa nell'industria manifatturiera. Questa configurazione si ispira molto al braccio umano ed è dotata di tre gradi di libertà di rotazione. Comunemente, la struttura antropomorfa è impiegata quando è necessaria grande versatilità e flessibilità, infatti, tale struttura permette di evitare ostacoli all'interno del volume di lavoro (workspace). Per tale motivo si presta bene ad operazioni di montaggio, asservimento macchine utensili, saldatura, verniciatura ecc.

Figura 6 Robot con struttura antropomorfa

Un'altra tipologia di struttura per dispositivi robotici che si sta sempre maggiormente diffondendo è la **struttura parallela**, questa si distingue dalle precedenti perché è costituita da una catena cinematica chiusa. Nella struttura parallela sono presenti più catene cinematiche che connettono la base con l'organo terminale. Questa struttura ha il vantaggio di essere particolarmente rigida, precisa e rapida ma presenta lo svantaggio di un workspace ridotto rispetto a dispositivi robotici delle stesse dimensioni ma con catena cinematica aperta. Pertanto, la maggiore applicazione dei robot con struttura parallela è la manipolazione di piccoli oggetti.

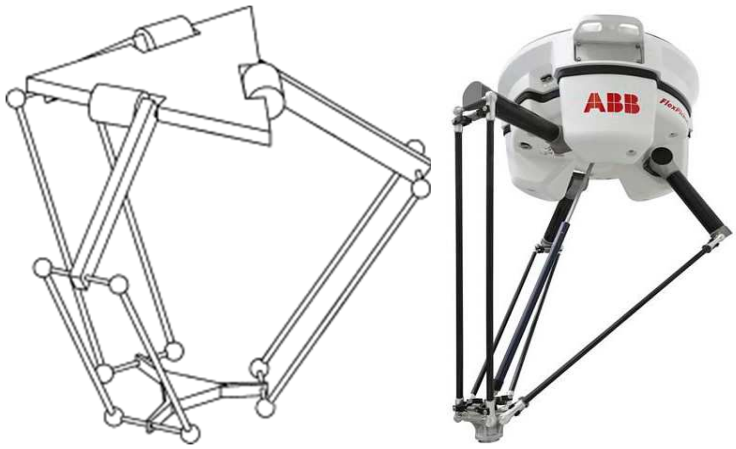


Figura 7 Robot con struttura parallela

Ovviamente questi descritti sono solo i gradi di libertà della struttura di base cui vanno aggiunti solitamente tre gradi di libertà di rotazione associati al polso del robot che conferiscono ulteriore versatilità.

In particolare, riferendosi al polso dei manipolatori antropomorfi, una comune soluzione è quella dei manipolatori antropomorfi con **polso monocentrico**, il polso del robot è definito in questo modo quando gli assi degli ultimi tre giunti si incrociano in un unico punto.

Come vedremo proseguendo con la presente trattazione, questa idea di versatilità tipica della struttura antropomorfa è oggetto di continue evoluzione che hanno condotto all'incremento dei gradi di libertà del braccio robotico al fine di permettere allo stesso di poter svolgere uno stesso task assumendo diverse configurazione e permettendo pertanto di ottimizzare il task nell'ottica di sfruttare al meglio la flessibilità del robot. Questo ha condotto allo sviluppo di sempre più articolate logiche di controllo del robot, ed è proprio su questo punto che si vuole focalizzare l'attenzione e procedere con la ricerca ed il lavoro di questa tesi di laurea.

1.2 Evoluzione storica dei robot industriali e requisiti di sicurezza

A causa della loro crescente flessibilità i robot sono sempre più utilizzati nella produzione industriale nei più svariati campi: manipolazione (pick and place), verniciatura, saldatura ecc. Infatti, i dispositivi robotici vantano grande ripetibilità, flessibilità, precisione, velocità e forza. Sebbene, l'elevata produttività sia l'aspetto cruciale di qualsiasi industria meccanica, questa deve sempre sottostare alla **sicurezza** per i lavoratori. Per questo motivo, l'introduzione in ambiente industriale dei robot genera numerose controversie. I primi robot introdotti a metà degli anni Sessanta sono ad attuazione idraulica, molto ingombranti e costituiscono un pericolo evidente per la sicurezza in ambiente di lavoro. Successivamente lo sviluppo tecnologico

permette l'introduzione dell'attuazione elettrica e l'evoluzione dei robot. In conseguenza all'incalzante sviluppo tecnologico sono redatte le prime **normative** americane ed internazionali per controllare l'uso dei robot in ambiente industriale compatibilmente con la sicurezza degli operatori che ci lavorano a stretto contatto.

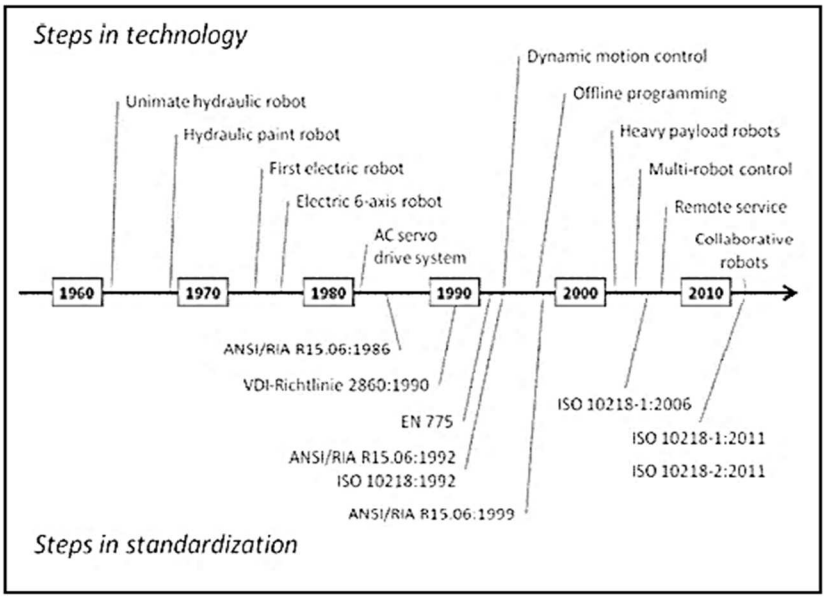


Figura 8 sviluppo tecnologico e standardizzazione in ambito della robotica industriale

Sebbene nel corso degli anni Ottanta – Novanta lo sviluppo tecnologico è incredibilmente rapido fino agli anni Duemila i robot sono quasi esclusivamente limitati in aree recintate a cui è precluso l'accesso di qualsiasi operatore quando il dispositivo robotico è in movimento. Infatti, è solo negli ultimi anni che nasce la cosiddetta robotica collaborativa con la quale si intende la libera, ma sicura, interazione tra robot e macchina senza problemi per l'incolumità dei lavoratori e continuando a mantenere elevata la produttività.

Come si può notare dal prospetto storico riportato sopra, per garantire la sicurezza in ambiente di lavoro, nel **1986** nasce la prima legislazione per fissare degli **standard di sicurezza** da parte del Robotic Indutries Association (RIA). In seguito, nel **1992** la ISO estende i criteri di sicurezza in ambito robotico anche in Europa. Nonostante il simile scopo dei due documenti il primo si concentra maggiormente sull'integrazione e l'utilizzo dei robot mentre la norma ISO 10218:1992 (EN 775) dà maggiori **informazioni ai produttori** di robot per una loro produzione sicura per il successivo inserimento in ambiente di lavoro. In seguito, con la norma ANSI/RIA R15.06-**1999** si introducono i criteri per condurre un **risk assessment** a seguito dell'installazione di qualsiasi dispositivo industriale, in quanto, il livello di pericolo che essi possono generare varia in funzione del sistema produttivo in cui sono inseriti. All'alba del nuovo millennio l'evoluzione tecnologica permette l'inserimento di molti nuovi elementi nel controllo dei robot e questo genera la necessità di un'evoluzione anche degli standard di sicurezza per permettere una migliore interazione tra uomo e robot (HRI, Human Robot Interaction). Per tali motivi si giunge alla pubblicazione nel **2006** della normativa tecnica ISO 10218-1. In seguito, la pubblicazione della norma ISO 10218-2:**2011** permette di avere un chiaro quadro sui requisiti di sicurezza per i sistemi robotici. Questa normativa unifica però gli standard di sicurezza in Europa mentre in USA ed in Canada sono condotti lavori che

conducono alla pubblicazione di ANSI/RIA R15.06 e CAN/CSA Z434 che riportano la serie degli standard presentati nella ISO 10218.

Negli **ultimi anni** l'avvento di robot definiti collaborativi e la crescente interazione tra robot e uomo porta ad un nuovo sviluppo degli standard industriali. Infatti, oggetto di studio crescente è l'analisi del pericolo in cui un uomo si può trovare a causa della posa assunta da un dispositivo robotico. Infatti, i manipolatori robotici sebbene abbiano subito molte evoluzioni possiedono ancora una elevata inerzia e pertanto possono trasmettere forze non trascurabili. La sfida che ci troviamo ad affrontare oggi è quella di studiare un sempre più evoluto ma flessibile ambiente industriale in cui l'uomo possa inserire la sua estrema versatilità e capacità di intervento in situazioni impreviste e movimenti particolari, mentre, la macchina possa intervenire con la sua elevata forza, rapidità e precisione compatibilmente con la presenza di operatori e dei loro movimenti. Alcuni aspetti di questa collaborativa interazione tra uomo e robot sono stati permessi nel soddisfacimento di requisiti presentati dallo standard **ISO 10218**. Sebbene questo documento fornisce alcuni basilari ma fondamentali requisiti di sicurezza ulteriori dettagli saranno disponibili in futuro. Sotto questa ottica è previsto che nella cella di lavoro del robot ci sia un volume chiamato "collaborative work space" (**CWS**) in cui è consentita la contemporanea presenza del robot e dell'operatore. Le più comuni operazioni collaborative oggi previste sono: arresto del robot quando l'operatore si trova nel CWS, movimento dello stesso mediante guida manuale da parte dell'operatore a velocità ridotte, monitoraggio della velocità a cui avviene il contatto tra uomo e robot, limitazione della potenza e della forza applicata dal robot in fase di collaborazione.

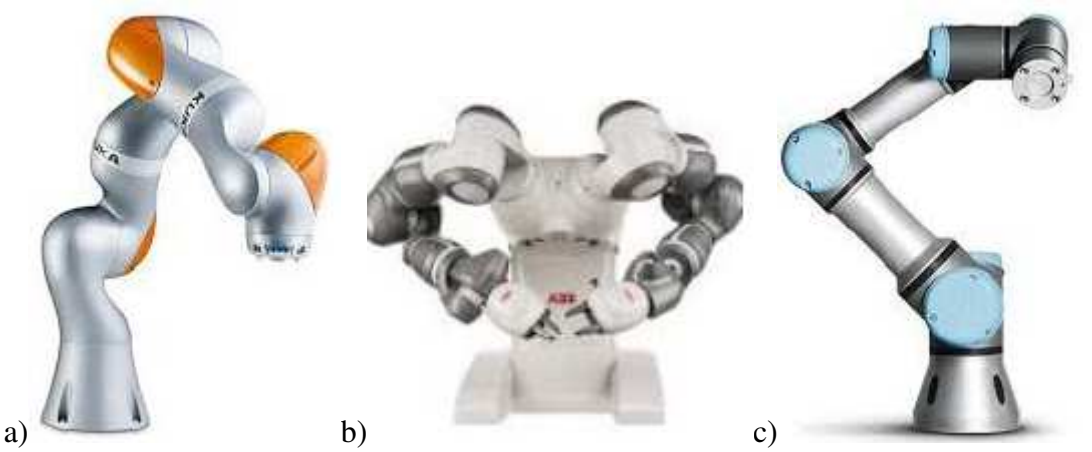
Il principale problema è stabilire dei **limiti** a questa **collaborazione** tra uomo e macchina, a tale proposito sono state condotte un gran mole di ricerche. Ad esempio, ci sono studi per valutare gli effetti dell'urto tra il robot e varie parti del corpo umano (principalmente il capo) in base a vari livelli di velocità, inclinazione, posa del manipolatore.

In conclusione, per garantire una crescente cooperazione tra robot e uomo, è fondamentale non solo che vengano studiati e definiti limiti di sicurezza ma anche che vengano scelti quali parametri valutare come fonte di "pericolo" nella collaborazione e che poi siano messi a punto accurati sistemi di valutazione e di controllo del dispositivo robotico. (2)

1.3 Esempi di robot collaborativi e loro applicazione

Già da alcuni anni sono prodotti ed utilizzati in ambito industriale robot collaborativi, spesso anche chiamati **cobot** (collaborative robot). L'obiettivo di eliminare le recinzioni che circondano i robot per permettere ad operatori umani di intervenire nel workspace del robot è perseguito da alcune aziende specializzate in robotica mediante la realizzazione di robot leggeri ed "intelligenti". Uno dei primi esempi di manipolatore collaborativo risale al 2014 quando KUKA lancia il primo esponente della serie **LBR iiwa**. L'innovazione di questo braccio robotico è quella di essere sensibile al contatto e pertanto, avendo la capacità di rilevare forze di resistenza durante l'esecuzione del task, può arrestarsi ed evitare l'insorgere di problematiche di sicurezza. LBR è l'acronimo di "Leichtbauroboter" ovvero robot leggero, mentre iiwa significa "intelligent industrial work assistent". Un altro esempio di cobot è **IRB 14000 YuMi** prodotto da ABB, altra azienda leader nel settore della robotica ed automazione industriale.

Questo è un robot collaborativo a doppio braccio, realizzato per l'assemblaggio di piccoli oggetti. Questo dispositivo robotico entra sul mercato il 13 aprile 2015. Infine, un terzo esempio, molto noto nel campo della robotica collaborativa, è la serie di robot collaborativi realizzati dall'azienda danese Universal Robot **UR3**, **UR5** e **UR10** con l'obiettivo di permettere una programmazione user friendly mediante apposito schermo touch screen definito teach pendant e con l'implementazione di opportune soglie di resistenza quando il robot incontra un ostacolo che incrementano il livello di sicurezza e la possibilità di una programmazione e movimentazione libera del robot semplicemente trascinando lo stesso nella posizione, o sequenza di posizioni, necessarie.



a) b) c)

Figura 9 esempi di robot collaborativi in produzione a) KUKA LBR iiwa 7 R800 b) ABB IRB 14000 YuMi c) Universal Robot UR3

Questi sono alcuni dei più noti ed utilizzati robot in ambito collaborativo presenti nelle industrie di tutto il mondo. Ciononostante, sono solo il primo passo verso una collaborazione tra uomo e macchina sempre maggiore. In tal senso è necessario una sempre maggiore concentrazione sul controllo del manipolatore con sofisticati algoritmi.

1.4 Cinematica dei robot

Un punto cruciale nell'analisi di un manipolatore robotico è la capacità di trasformare le coordinate nello spazio operativo nelle coordinate nello spazio dei giunti. Con il termine **spazio operativo** si intende lo spazio nel quale viene tipicamente specificata la posizione e l'orientazione del manipolatore e può essere pensato come uno spazio tridimensionale caratterizzabile con delle coordinate cartesiane e degli angoli che specificano l'orientazione dell'EE. D'altro canto, con **spazio dei giunti**, si intende lo spazio vettoriale nel quale sono specificate le variabili di giunto, le quali vengono fissate in fase di pianificazione della traiettoria e permettono all'EE (End Effector) del manipolatore di raggiungere una certa coordinata spaziale con eventualmente una specifica orientazione.

La **cinematica diretta** è la chiave di volta che permette il passaggio dalle coordinate dallo spazio dei giunti alle coordinate nello spazio operativo attraverso la nota relazione:

$$\mathbf{x_e} = \boldsymbol{f}\,(\boldsymbol{q}) \tag{1.1}$$

Dove: $\mathbf{x_e}$ è il vettore $(r\ x\ 1)$ delle coordinate nello spazio operativo;

$\boldsymbol{q}$ è il vettore $(n\ x\ 1)$ delle coordinate nello spazio dei giunti;

$\boldsymbol{f}$ è la funzione vettoriale continua non lineare la cui forma è ricavabile conoscendo la struttura del manipolatore.

La relazione alla base della **cinematica differenziale** e che permette il calcolo della velocità di traslazione e rotazione dell'EE conoscendo le velocità dei giunti costituenti la catena cinematica è:

$$\mathbf{v_e} = \boldsymbol{J}\ (\dot{\boldsymbol{q}}) \tag{1.2}$$

Dove: $\mathbf{v_e}$ è il vettore $(r\ x\ 1)$ delle velocità nello spazio operativo;

$\dot{\boldsymbol{q}}$ è il vettore $(n\ x\ 1)$ delle velocità nello spazio dei giunti;

$\boldsymbol{J}$ è la matrice Jacobina di dimensioni (r x n).

Così come la funzione vettoriale $\boldsymbol{f}$ è ricavabile nota la geometria del manipolatore e la tipologia di giunti costituenti la catena cinematica del dispositivo robotico analizzato, la matrice Jacobiana è ricavabile noto anche il vettore delle variabili di giunto $\boldsymbol{q}$.

Prima di procedere, è opportuno fare un'osservazione sulle dimensioni dei vettori e delle matrici riportate nella presente trattazione. Si osservi che è indicato con n il **numero di giunti** di cui è composto il manipolatore e facendo riferimento a giunti rotoidali o giunti prismatici, n corrisponde anche al numero di gradi di libertà di cui è dotato il manipolare (in quanto ciascuno di questi giunti dà al manipolatore un grado di libertà). Con m si indica, invece, la **dimensione dello spazio operativo**. Pertanto, risulta $m = 2$ con riferimento a traslazione su un piano bidimensionale, $m = 3$ nel caso di traslazione nello spazio e $m = 6$ se si indicano tutti i gradi di libertà di rotazione e traslazione nello spazio per garantire non solo un prefissato posizionamento ma anche una specifica orientazione dell'EE nel workspace. Infine, si indica con r il numero di componenti dello spazio operativo che sono richieste per la realizzazione di uno specifico task.

Fissata questa convenzione si riporta ora un'efficace rappresentazione grafica che permetta la comprensione della **relazione** tra lo **spazio operativo** e lo **spazio giunti**.

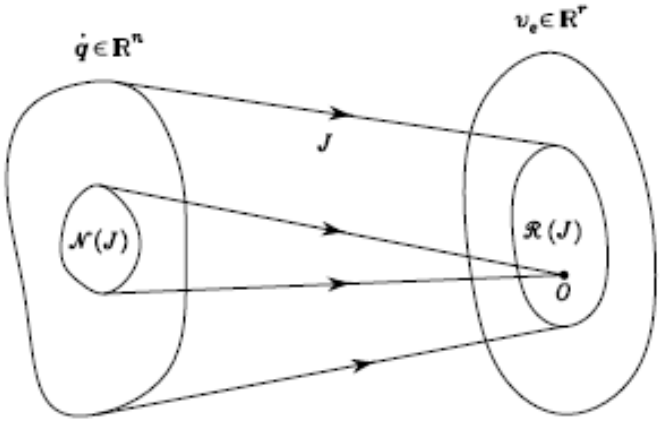


Figura 10 Relazione spazio operativo - spazio giunti

L'equazione della cinematica differenziale può essere espressa mediante l'immagine e il nullo della trasformazione ricordando che: l'**immagine** $\mathcal{R}(\boldsymbol{J})$ di $\boldsymbol{J}$ è il sottospazio in $\mathbb{R}^r$ che individua le velocità dell'organo terminale che possono essere generate dalle velocità di giunto, nella configurazione assegnata al manipolatore, il **nullo** $\mathcal{N}(\boldsymbol{J})$ di $\boldsymbol{J}$ è il sottospazio in $\mathbb{R}^n$ a cui appartengono le velocità di giunto che non producono alcuna velocità all'organo terminale nella configurazione assegnata al manipolatore. (1)

1.4.1 Singolarità

Si definisce **configurazione singolare** di un manipolatore una configurazione dei giunti $\boldsymbol{q}$ in cui lo Jacobiano geometrico non ha rango massimo. In tale condizione la dimensione dell'immagine dello Jacobiano diminuisce e allo stesso tempo aumenta quella del nullo in quanto vale sempre la relazione:

$$\dim\big(\mathcal{R}(\boldsymbol{J})\big) + \dim\big(\mathcal{N}(\boldsymbol{J})\big) = n \tag{1.3}$$

In una configurazione singolare è impossibile generare velocità in determinate direzioni e si ha una ridotta mobilità del manipolatore.

Esistono tre tipologie di singolarità:

- **Singolarità cinematiche**: sono dovute a particolari pose assunte dal manipolatore, in tali condizioni si ha perdita di mobilità e le soluzioni del problema cinematico inverso sono infinite. In tale condizione, ridotte velocità richieste nello spazio operativo possono richiedere elevate velocità nello spazio giunti.
- **Singolarità di rappresentazione**: sono quelle configurazioni in cui diventa singolare la matrice usata per la rappresentazione dell'orientamento, ovvero quella matrice che lega la velocità angolare e la derivata del vettore che esprime l'orientamento.
- **Singolarità algoritmiche**: si possono verificare quando si cerca di risolvere l'inversione cinematica con un sub-task.

Le singolarità di maggiore interesse per un manipolatore sono le singolarità cinematiche, in quanto, in tali configurazioni si riduce il rango della matrice Jacobiana rispetto al massimo rango. Nelle configurazioni di singolarità si riduce la mobilità della struttura, le soluzioni della cinematica inversa diventano infinite anche per manipolatori non ridondanti e nelle vicinanze di una condizione di singolarità piccole velocità nello spazio operativo possono causare elevate velocità nello spazio dei giunti.

Inoltre, le singolarità solitamente si distinguono ulteriormente in: singolarità interne e di bordo. Le **singolarità di bordo** si verificano in corrispondenza delle estremità del campo di mobilità del manipolatore e possono essere evitate molto più agevolmente rispetto alle **singolarità interne** che si verificano nello spazio operativo raggiungibile e sono solitamente causate dall'allineamento di uno o più assi del manipolatore (1).

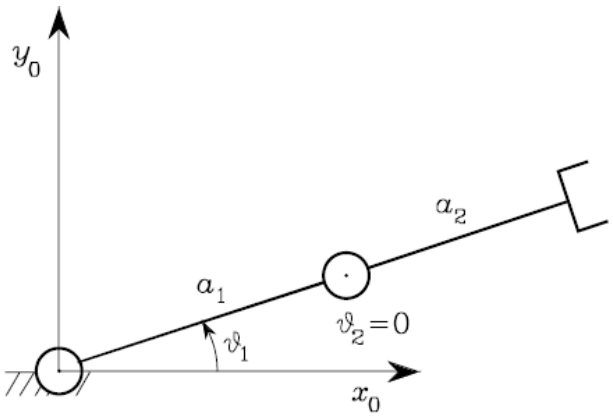


Figura 11 Singolarità di bordo di un manipolatore planare a 2 DOF (Degree of Freedom)

Le più comuni singolarità interne di un manipolatore antropomorfo con polso monocentrico sono:

- Singolarità del polso;
- Singolarità della spalla;
- Singolarità del gomito.

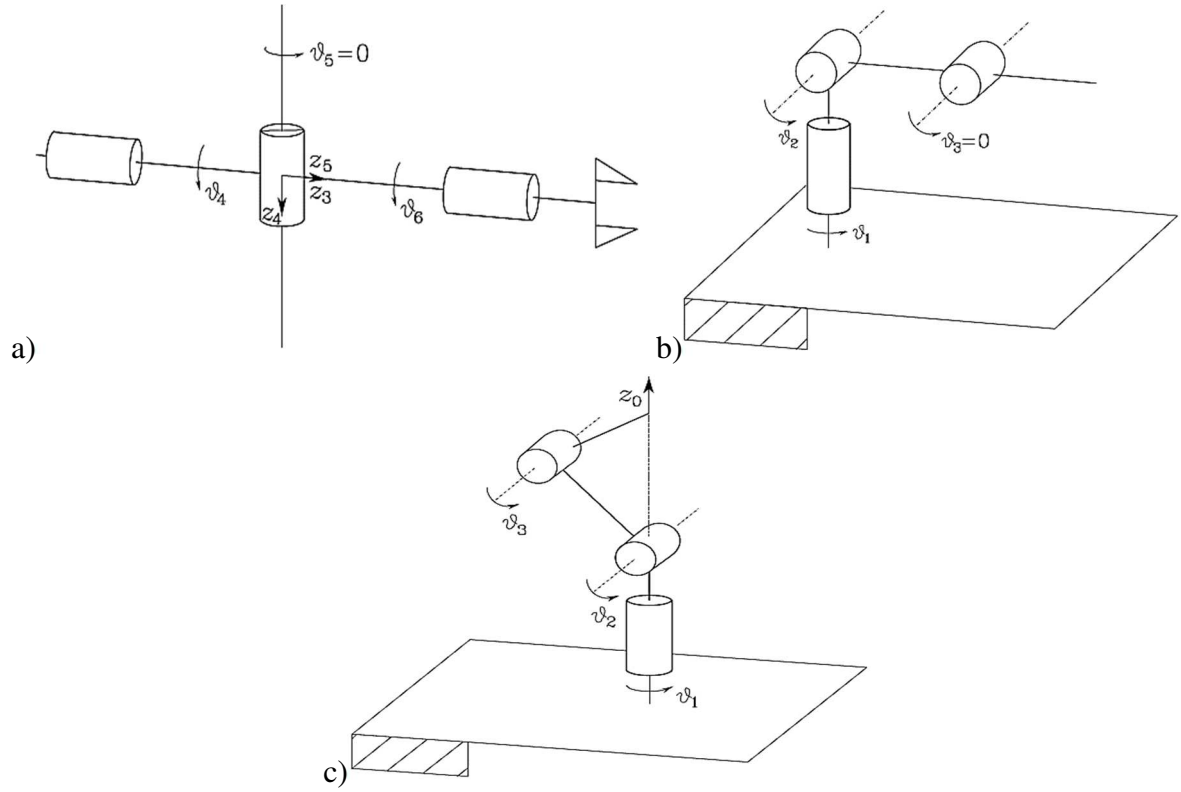


Figura 12 a) Singolarità di polso b) Singolarità di gomito c) Singolarità di spalla

1.4.2 Ridondanza

Un **manipolatore** robotico si definisce **cinematicamente ridondante** quando è dotato di un numero di gradi di libertà superiore a quello necessario per eseguire il compito assegnato ($n > r$). Pertanto, è bene sottolineare che la condizione di ridondanza è definita in funzione al task che deve essere svolto. Un manipolatore cinematicamente ridondante è solitamente più

complesso da controllare però ha il considerevole vantaggio di poter eseguire un determinato task in modi diversi e questo ne concede maggiore flessibilità nei movimenti. Si noti inoltre che un manipolatore ridondante è dotato di **self-motion** ovvero è possibile effettuare dei movimenti della struttura senza variare la posizione e l'orientazione dell'EE nello spazio sfruttando il cosiddetto spazio nullo.

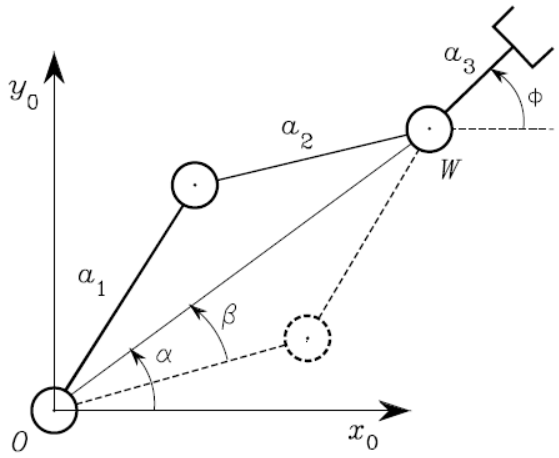


Figura 13 Esempio di manipolatore planare ridondante che acquisisce la stessa posizione dell'EE con due diverse configurazioni

Matematicamente, in base alle definizioni di immagine e spazio nullo data in precedenza, ogni qualvolta che il nullo $\mathcal{N}(\boldsymbol{J})$ di $\boldsymbol{J}$ risulta diverso dallo spazio nullo il manipolatore è dotato di gradi di ridondanza.

Si definisce pertanto **grado di ridondanza** la dimensione dello spazio nullo, ovvero, la differenza tra il numero di gradi di libertà del manipolatore e il numero di gradi di libertà necessari per l'esecuzione di un determinato task:

$$l = \dim(\mathcal{N}(\boldsymbol{J})) = n - r \tag{1.4}$$

Indicando con $\dot{\boldsymbol{q}}^*$ la soluzione dell'equazione della cinematica differenziale (1.2) e con $\boldsymbol{P}$ una matrice n x n tale che:

$$\mathcal{R}(\boldsymbol{P}) \equiv \mathcal{N}(\boldsymbol{J}) \tag{1.5}$$

Anche il vettore velocità di giunto scritto:

$$\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}^* + \boldsymbol{P}\dot{\boldsymbol{q}}_0 \tag{1.6}$$

Con $\dot{\boldsymbol{q}}_0$ vettore arbitrario di velocità nello spazio giunti e delle dimensioni opportune è soluzione dell'equazione della cinematica differenziale prima citata. Ed è prorpio questo vettore che è indicativo del fenomeno di self-motion. (1)

1.5 Algoritmi di inversione cinematica

L'operazione di inversione cinematica permette di risalire al valore che bisogna imporre alle variabili di giunto per ottenere una specifica posa del manipolatore. Il problema principale di tale operazione, che risulta molto più complessa della cinematica diretta, è che le relazioni che

esprime la posa del manipolatore nello spazio operativo e il vettore delle variabili di giunto sono non lineari. In modo concettualmente adottabile per risolvere il problema è avvalersi dell'equazione della cinematica differenziale (1.2). Se ipotizziamo che $n = r$ è possibile ricavare il vettore delle velocità di giunto invertendo la relazione della **cinematica differenziale** e pertanto calcolando l'inversa della matrice Jacobiana:

$$\dot{\mathbf{q}} = \boldsymbol{J}^{-1}(\mathbf{q})\mathbf{v_e} \tag{1.7}$$

Da tale relazione, se è noto il vettore delle variabili di giunto all'istante iniziale, mediante integrazione, è possibile calcolare lo stesso ad un qualsiasi istante temporale t_f:

$$\mathbf{q}(\mathrm{t}) = \int_0^{t_f} \dot{\mathbf{q}}(t)\, dt + \mathbf{q}(0) \tag{1.8}$$

Ovviamente, tale integrazione al calcolatore è fatta per via numerica e può pertanto essere espressa mediante l'equazione:

$$\mathbf{q}(\mathrm{t}) = \mathbf{q}(\mathrm{t} - \Delta\mathrm{t}) + \dot{\mathbf{q}}(t - \Delta\mathrm{t})\Delta\mathrm{t} \tag{1.9}$$

Indicando sempre con $\Delta\mathrm{t}$ l'intervallo di discretizzazione temporale delle variabili considerate. Il problema di queste metodo risolutivo è che, per essere matematicamente valido, la matrice Jacobiana deve essere quadrata e non singolare ovvero con determinante diverso da zero. Tali condizioni non vengono soddisfatte se:

- Il **manipolatore** è **ridondante** (in questo caso lo Jacobiano non è una matrice quadrata);
- Il **manipolatore** è in **condizioni singolari** (in questo caso la matrice Jacobiana seppur quadrata ha determinante nullo).

Inoltre, ulteriore complicazione nell'applicare questo metodo di inversione dell'equazione della cinematica differenziale è che, se matematicamente questo approccio non presenta problemi, quando lo si applica a variabili discrete, dovendo ricorrere all'operazione di integrazione numerica si introduce un fenomeno di **deriva**. Prima di analizzare le varie tecniche di inversione cinematica si ricordi ancora la differenza tra lo Jacobiano analitico e lo Jacobiano geometrico che sono due strumenti molto importanti. La matrice $\boldsymbol{J}\ (6\ x\ n)$ riportata nell'equazione della cinematica differenziale e che pertanto lega la velocità nello spazio operativo alle velocità di giunto è lo **Jacobiano geometrico**, ovvero, quella matrice così costituita:

$$\mathbf{J} = \begin{bmatrix} \boldsymbol{J}_P \\ \boldsymbol{J}_O \end{bmatrix} \tag{1.10}$$

Dove: $\boldsymbol{J}_P$ è la matrice $(3\ x\ n)$ che lega il vettore velocità di giunto alla velocità lineare $\dot{\boldsymbol{p}}_e$ mediante la relazione $\dot{\boldsymbol{p}}_e = \boldsymbol{J}_P(\boldsymbol{q})\dot{\boldsymbol{q}}$;

$\boldsymbol{J}_O$ è la matrice $(3\ x\ n)$ che lega il vettore velocità di giunto alla velocità angolare $\boldsymbol{\omega}_e$ mediante la relazione $\boldsymbol{\omega}_e = \boldsymbol{J}_O(\boldsymbol{q})\dot{\boldsymbol{q}}$;

Tale matrice è detta Jacobiano geometrico perché per calcolarla è necessario adottare un procedimento geometrico. D'altro canto, lo **Jacobiano Analitico** si può ricavare mediante

derivazione delle relazioni cinematiche rispetto alle derivate delle variabili di giunto. Pertanto, tale matrice si ricava derivando l'equazione (1.1) della cinematica diretta:

$$\dot{\boldsymbol{x}}_e = \begin{bmatrix} \dot{\boldsymbol{p}}_e \\ \dot{\boldsymbol{\Phi}}_e \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}_P(\boldsymbol{q}) \\ \boldsymbol{J}_O(\boldsymbol{q}) \end{bmatrix} \dot{\boldsymbol{q}} = \boldsymbol{J}_A(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{1.11}$$

E pertanto risulta:

$$\boldsymbol{J}_A(\boldsymbol{q}) = \frac{\partial \boldsymbol{k}(\boldsymbol{q})}{\partial \boldsymbol{q}} \tag{1.12}$$

In cui si ricorda che $\boldsymbol{k}(\boldsymbol{q})$ è la funzione vettoriale continua non lineare la cui forma è ricavabile conoscendo la struttura del manipolatore.

1.5.1 Algoritmo Closed Loop basato sulla pseudo-inversa

Come accennato il metodo di integrazione dell'equazione della cinematica differenziale non garantisce che le singolarità cinematiche siano evitate e può generare deriva numerica. Per questo motivo, uno step successivo è l'adozione di un algoritmo CLIK (Closed Loop Inverse Kinematics). Se $\boldsymbol{J}_A$ è quadrata e non singolare l'equazione:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_A^{-1}(\boldsymbol{q})(\dot{\boldsymbol{x}}_{\boldsymbol{d}} + \boldsymbol{K}\boldsymbol{e}) \tag{1.13}$$

Dove: $\boldsymbol{x}_{\boldsymbol{d}}$ è il vettore indicante la posizione desiderata dell'organo terminale;

$\boldsymbol{K}$ è una matrice definita positiva;

$\boldsymbol{e} = \boldsymbol{x}_{\boldsymbol{d}} - \boldsymbol{x}_{\boldsymbol{e}}$ è il vettore errore di posizionamento nello spazio operativo.

È equivalente all'equazione seguente:

$$\dot{\boldsymbol{e}} + \boldsymbol{K}\boldsymbol{e} = \boldsymbol{0} \tag{1.14}$$

Che rappresenta un sistema asintoticamente stabile.

L'algoritmo di inversione cinematica può essere brevemente illustrato mediante il seguente schema a blocchi. (1)

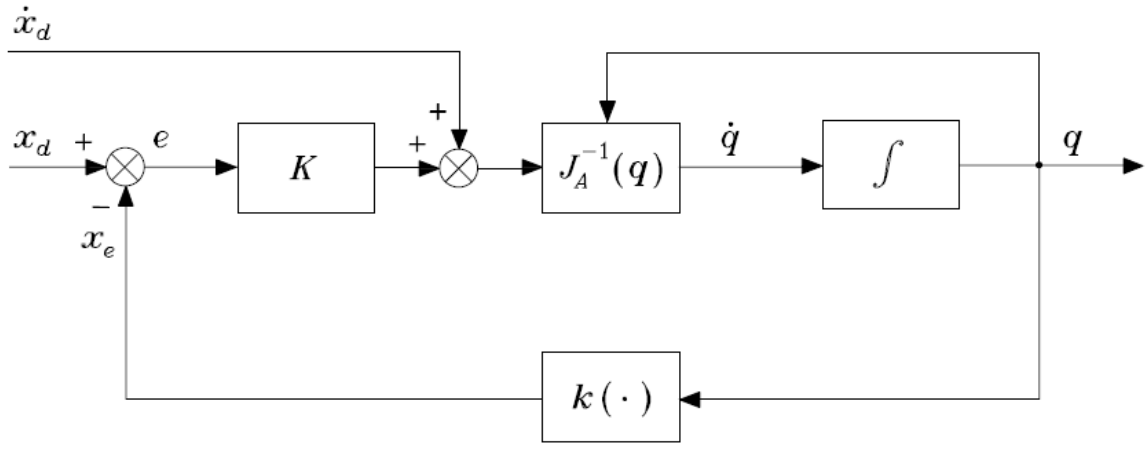


Figura 14 Schema a blocchi dell'algoritmo CLIK con inversa dello Jacobiano

La complicazione dell'algoritmo di inversione qui riportato si ha quando il manipolatore è ridondante. In tal caso occorre modificare la relazione (1.13) con la seguente:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_A^{\dagger}(\dot{\boldsymbol{x}}_{\boldsymbol{d}} + \boldsymbol{K}\boldsymbol{e}) + \left(\boldsymbol{I}_n - \boldsymbol{J}_A^{\dagger}\boldsymbol{J}_A\right)\dot{\boldsymbol{q}}_0 \tag{1.15}$$

Dove: $\boldsymbol{J}_A^{\dagger} = \boldsymbol{J}_A^T(\boldsymbol{J}_A\boldsymbol{J}_A^T)^{-1}$ la matrice **pseudo-inversa** dello Jacobiano analitico;

$\dot{\boldsymbol{q}}_0$ è il vettore delle velocità di giunto appartenenti allo spazio nullo.

Sostanzialmente, si entra con l'informazione di posizione nello spazio operativo, ci si avvale della matrice Jacobina pseudo-inversa, si esegue un'integrazione numerica ma ci chiude il Loop per una migliore efficacia utilizzando la cinematica diretta del manipolatore.

1.5.2 Algoritmi CLIK: trasposta

Un algoritmo computazionalmente meno oneroso del precedente ma anche meno efficiente in talune situazioni è quello che utilizza la **trasposta** dello Jacobiano piuttosto che la matrice pseudo-inversa. Lo schema a blocchi che descrive tale algoritmo è di seguito riportato.

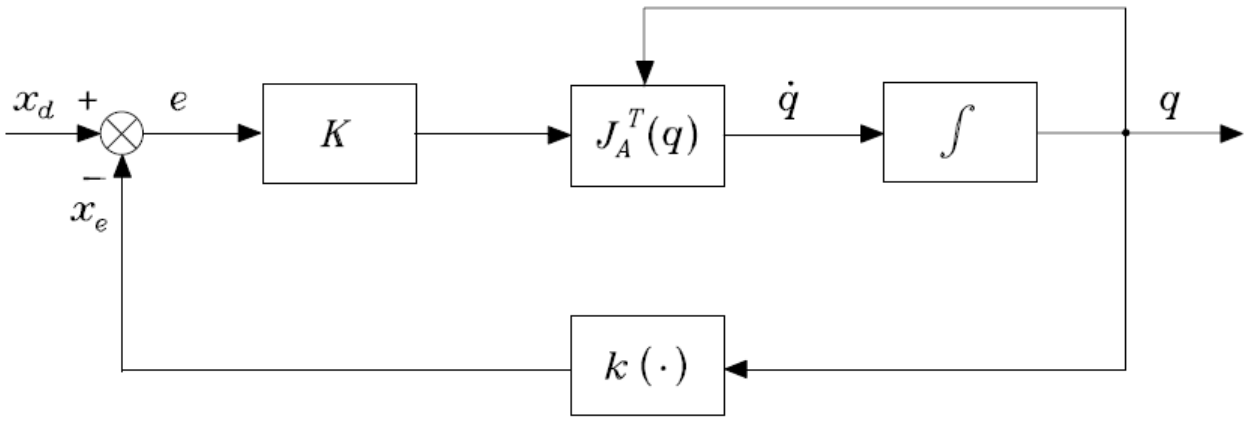


Figura 15 Schema a blocchi dell'algoritmo CLIK con trasposta dello Jacobiano

Da tale relazione si desume che il vettore delle velocità di giunto può essere espresso mediante la relazione:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_A^T(\boldsymbol{q})\boldsymbol{K}\boldsymbol{e} \tag{1.16}$$

Anche in questo caso, come nel caso precedente risulta il metodo asintoticamente stabile se la matrice $\boldsymbol{K}$ è simmetrica e definita positiva.

1.5.3 Algoritmi con ottimo vincolato

Un altro metodo di inversione cinematica, solitamente adottato per i manipolatori ridondanti, consiste nello sfruttare i gradi di libertà eccedenti del manipolatore per l'esecuzione di task ausiliari. In questo caso, per esprimere le condizioni aggiuntive si definisce una **funzione di costo** e si determina il vettore delle velocità di giunto compatibilmente con l'operazione di minimizzazione o massimizzazione di tale funzione. Ad esempio, per la minimizzazione delle velocità di giunto del manipolatore è possibile adottare la seguente funzione di costo quadratico lineare:

$$g(\dot{\boldsymbol{q}}) = \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{W}\dot{\boldsymbol{q}} \tag{1.17}$$

Dove: $\boldsymbol{W}$ è una matrice di peso $(n\ x\ n)$ simmetrica e definita positiva

Tale metodo di ottimizzazione può anche essere utilizzato per evitare degli ostacoli, ad esempio, massimizzando una funzione di costo che esprime la distanza tra il manipolatore e l'ostacolo. Inoltre, gli algoritmi che utilizzando l'ottimo vincolato, presentano il vantaggio di esprimere delle funzioni di costo anche direttamente nello spazio operativo del manipolatore. D'altro canto, possono generare singolarità algoritmiche e siccome l'ottimizzazione delle funzioni di costo coinvolge tutte le variabili di giunto e non solo i gradi di libertà di ridondanza questo può anche pregiudicare la corretta esecuzione del task principale per il quale il manipolatore è stato originariamente programmato.

1.5.4 Algoritmi con task secondari mediante proiezione nello spazio nullo

Per compensare il problema precedentemente citato e continuare a sfruttare la ridondanza del manipolatore è necessario imporre dei **task secondari** che abbiano inferiore priorità rispetto al task primario che deve essere svolto. Questo può essere fatto utilizzando, ad esempio, lo spazio nullo del manipolatore. In precedenza, è stato osservato che se $\dot{\boldsymbol{q}}$ è soluzione dell'equazione cinematica differenziale e $\boldsymbol{P}$ è una matrice $(n\ x\ n)$ tale che $\mathcal{R}(\boldsymbol{P}) \equiv \mathcal{N}(\boldsymbol{J})$ anche $\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}^* + \boldsymbol{P}\dot{\boldsymbol{q}}_0$ con vettore arbitrario di velocità nello spazio giunti e delle dimensioni opportune, risolve l'equazione della cinematica differenziale. Pertanto, come visto nel paragrafo precedente, è possibile esplicitare una funzione di costo:

$$g'(\dot{\boldsymbol{q}}) = \frac{1}{2}\left(\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_{\boldsymbol{0}}\right)^T (\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_{\boldsymbol{0}}) \tag{1.18}$$

Come detto questo metodo presenta il vantaggio di introdurre un task secondario che abbia priorità inferiore e che pertanto può essere rilassato in caso la soluzione faccia tendere a delle singolarità algoritmiche ma, al contempo, risulta anche più importante da un punto di vista computazionale

1.5.5 Algoritmi con compito aumentato

Infine, un altro metodo per affrontare il problema di risoluzione della cinematica inversa di un manipolatore ridondante che merita di essere citato è quello del compito aumentato. In tale approccio sono aggiunti $n - m$ **vincoli nello spazio operativo** in modo da eliminare virtualmente la ridondanza del manipolatore.

1.6 Manipolatori ridondanti e loro applicazioni

Come già accennato il concetto di **ridondanza cinematica** è legato al numero *n* di gradi di libertà della struttura, al numero m di variabili necessarie alla caratterizzazione dello spazio operativo ed al numero *r* di variabili dello spazio operativo necessarie per specificare il compito. Pertanto, anche nel caso in cui $m = n$ un manipolatore può risultare ridondante se $r < m$ ovvero se per l'esecuzione di un compito il manipolatore necessita di un numero minore di variabili nello spazio operativo di quelle possedute. (1)

Il vantaggio dei manipolatori ridondanti è quello di poter eseguire uno specifico task in un numero infinito di pose. Pertanto, questo si traduce nella possibilità di introdurre un task aggiuntivo o un subtask oltre l'operazione normalmente prevista. Per questa estrema versatilità e flessibilità di utilizzo i robot ridondanti sono oggi utilizzati nei più svariati campi e per l'esecuzione dei più diversi task secondari. Ciononostante, questo vantaggio si traduce in un più o meno evidente aumento della complessità in fase di controllo del manipolatore.

Di seguito si elencano alcune delle più comuni applicazioni oggetto di studio.

1.6.1 Massimizzare velocità esecuzione task

Massimizzare la velocità di esecuzione di un task comporta la **riduzione dei tempi** di produzione. Pertanto, applicando questo ragionamento ai manipolatori industriali, siccome essi svolgono principalmente attività fortemente ripetitive, comporta un aumento della produttività.

Altra osservazione alla base del criterio di massimizzazione della velocità di esecuzione di un determinato task è che essa risulta inevitabilmente limitata dalla coppia erogabile dagli attuatori dei giunti.

Fatte queste osservazioni importanti per inquadrare il problema di fondo è possibile classificare i vari approcci utilizzati per affrontarlo e risolverlo. Il problema di minimizzazione del tempo di esecuzione di un task per manipolatori ridondanti presenta avanzati studi quando il task è esplicitato nello spazio dei giunti mentre è un problema del tutto aperto quando il task è esplicitato nello spazio operativo. Questa non è però l'unica distinzione che si può fare tra i vari approcci presentati nel corso del tempo. Infatti, occorre anche distinguere in due gruppi le soluzioni proposte: approccio puramente cinematico ed approccio dinamico.

Nel caso di **percorso noto nello spazio dei giunti**, storicamente, il primo a proporre una soluzione al problema di pianificazione di una traiettoria per massimizzare la velocità di esecuzione del task è Hollerbach nel 1983. La soluzione proposta consiste nello scrivere l'equazione dinamica del manipolatore data una traiettoria nello spazio dei giunti e scalare l'intero problema rispetto ad un'unica variabile. A tal punto, calcolando le coppie necessarie ai giunti si verifica se esse rispettino i limiti di coppia ai giunti. Se le coppie richieste sono inferiori ai valori limite, opportunamente scelti, si scala l'intero problema mediante quest'unica variabile aumentando le coppie richieste progressivamente fino a raggiungere le massime coppie erogabili e conseguentemente massime velocità ottenibili dal manipolatore nell'esecuzione dell'attività prevista con il vantaggio di non dover calcolare ad ogni iterazione nuovamente la dinamica inversa del manipolatore (3).

Nel 1985, invece, sono presentati due soluzioni simili al problema del minimo tempo di controllo che riconduce sempre alla parametrizzazione del percorso nello spazio dei giunti rispetto ad un'unica variabile trasformando la dinamica non lineare ed i limiti degli attuatori in limiti sull'accelerazione dell'EE. In questo modo, si ottengono in uscita le coppie da fornire ai giunti per l'esecuzione del percorso usando un approccio open loop (4) (5).

Siccome l'approccio open loop può provocare comportamenti imprevedibili a fronte di variazione di parametri o errori di modellazione del problema nasce un approccio simile,

sempre basato sulla parametrizzazione della dinamica del sistema, ma in closed loop che risulta essere più stabile (6).

Il minimo comune denominatore di questi approcci è che il percorso è noto nello spazio dei giunti e si ricava la traiettoria nello spazio operativo. Ovviamente, molto più comune è conoscere la **traiettoria nello spazio operativo**. Questo passaggio risulta molto complesso, soprattutto nel caso di manipolatori cinematicamente ridondanti perché richiede la risoluzione della cinematica inversa. Per la risoluzione della cinematica inversa di manipolatori ridondanti ci sono diverse strade che possono essere intraprese. Una possibile soluzione è l'utilizzo di algoritmi ricorsivi basati sulla matrice Jacobiana pseudo-inversa non pesata, altra possibile soluzione è l'uso di un algoritmo di risoluzione della cinematica inversa basato sull'uso della matrice Jacobiana pseudo-inversa pesata. Nessuna delle due soluzioni dà una soluzione di ottimo quindi occorre testarle entrambe e confrontare il tempo di esecuzione dell'attività imposta al manipolatore per valutare quale dia il migliore risultato.

L'articolo (7) propone un metodo per ottenere migliore capacità di accelerazione e decelerazione nello spazio operativo definendo un percorso nello spazio dei giunti. Questo comporta una penalizzazione del moto dei giunti ed un alto rapporto inerzia/coppia. L'algoritmo descritto si basa sullo Jacobiano pseudo-inverso pesato, siccome però questo metodo garantisce solo una soluzione di ottimo locale è necessario introdurre un parametro scalare che deve essere ottimizzato mediante un'operazione di trial and error. L'equazione della dinamica del manipolatore si può risolvere mediante l'algoritmo di Shin e McKay che permette di definire la traiettoria che minimizza il tempo di esecuzione tenendo conto di generici limiti di coppia erogabile imposti con opportune modifiche che tengano conto che il manipolatore è ridondante.

Un esempio di applicazione particolare di ottimizzazione delle tempistiche di esecuzione di un task relativo alla realizzazione di un compito multiplo come può essere quello di saldatura per punti o di foratura riportato nell'articolo (8). In questo caso, in realtà, la traiettoria da seguire non risulta nota a priori ma si assumono noti solo i punti attraverso il quale l'EE deve spostarsi ed il problema si riduce alla schedulazione dell'ordine con cui trasferirsi da un punto all'altro garantendo il minimo tempo di esecuzione. Tale problema della schedulazione dell'ordine dei punti si affronta mediante un tipico Traveling Salesman Problem (TSP). Si noti che il TSP è un problema aperto che non ha soluzione ottimale e che può essere affrontato con più approcci: metodo dell'enumerazione (enumeration method), dynamic programming, branch and bound method, intelligent optimization method (GA), simulated annealing (SA), Particle Swaem Optimization (PSO).

Uno schema di controllo per manipolatori cinematicamente ridondanti noto un percorso geometrico e soggetto a limiti di giunto è suggerito, invece nell'articolo (9). L'approccio consiste nel considerare la traiettoria nello spazio operativo come una funzione di un'ascissa curvilinea s e di massimizzare, definita la traiettoria la $\dot{s}$ tenendo conto anche dei limiti di coppia erogabile dai giunti. Per l'effettiva risoluzione del problema di determinazione della traiettoria con tempo ottimale si usa la tecnica di phase-plane (rappresentazione grafica di alcuni limiti di equazioni differenziali) e la linear programming technique (metodo di ottimizzazione di una funzione obiettivo lineare soggetta a limiti).

Approccio molto simile al precedente sullo studio del problema mediante ascissa curvilinea ma imponendo una velocità di percorrenza costante sempre nel rispetto della coppia ai giunti è proposto in (10). Il metodo si articola in due step preliminari:

1) determinazione di un'area proibita dello spazio operativo;
2) valutazione del percorso a velocità costante realizzabile.

Fatto ciò si aumenta progressivamente della velocità fin quando non si trova il percorso a massima velocità e quindi a minimo tempo di esecuzione.

1.6.2 Minimizzare coppia ai giunti

Sebbene il metodo di risoluzione classico di un manipolatore ridondante è mediante la formulazione di una matrice Jacobiana pseudo-inversa. Le ricerche recenti si sono concentrate sulla risoluzione del problema mediante ottimizzazione basata sul Quadratic Programming (QP) ovvero mediante l'ottimizzazione di funzioni quadratiche con condizioni lineari. Oppure mediante GA e GPS. Altra tecnica utilizzata, con l'obiettivo di minimizzare le coppie, è completamente differente dalle precedenti e si basa su un'idea puramente meccanica e non su complessi algoritmi di controllo. Tale espediente si basa sulla rivoluzione del design meccanico del manipolatore utilizzato, spostando tutti gli attuatori alla base e utilizzando delle opportune linee di trasmissione in modo da ridurre considerevolmente massa ed inerzia della struttura meccanica del manipolatore. Si osservi, infine, che il problema di minimizzazione della coppia richiede ovviamente un indice da considerare e da minimizzare. L'approccio classico è quello di considerare la **norma 2** del vettore delle coppie dei giunti, questo però non garantisce né l'effettiva minimizzazione di una particolare coppia di giunto né il rispetto di limiti di coppia erogabile dagli attuatori. Per questo, si sviluppano altri approcci basati sulla minimizzazione della **norma infinito** del vettore delle coppie che ha un maggiore significato fisico. Minimizzare la coppia erogata dagli attuatori di un manipolatore è un problema principalmente di tipo energetico. Infatti, in letteratura si trovano indistintamente applicazioni in cui si minimizza la coppia erogata dagli attuatori ed altre in cui obiettivo finale è minimizzare il consumo di energia.

Nell'articolo (11) l'autore assume le traiettorie espresse nello spazio dei giunti individuate da un polinomio del quarto ordine ed utilizza GA e GPS come algoritmi di ottimizzazione per ottimizzare i coefficienti dei polinomi in modo da minimizzare la coppia erogata dagli attuatori. Il principale problema è calcolare traiettorie fattibili partendo da un percorso assegnato. Per la realizzazione di una traiettoria ottimale l'autore si pone l'obiettivo di minimizzare l'energia cinetica e di conseguenza la coppia erogata per la movimentazione del manipolatore. A tale scopo è possibile scrivere l'equazione dell'energia cinetica e dell'energia potenziale al fine di applicare l'equazione di Lagrange per scrivere le equazioni del moto per lo specifico manipolatore considerato. A questo punto è possibile applicare uno dei due algoritmi di minimizzazione citati. In particolare:

- Il **GA** (Genetic Algorithm) è un algoritmo di ricerca che trae ispirazione dal meccanismo della genetica naturale. Molti ricercatori applicano tale metodo soprattutto per manipolatori ridondanti esplicitando la generica funzione che esprime la variabile di giunto nel tempo con un polinomio. Ad esempio, con un polinomio del quarto ordine

si hanno 5 coefficienti da determinare. Impostando un set di primo tentativo che funge da prima generazione, l'algoritmo varia tali parametri ricercando la soluzione di ottimo con le successive generazioni.

- IL **GPS** (Generalized Pattern Search) è un algoritmo di ottimizzazione ibrido basato sul Genetic Algorithm e sul Pattern Search ed ha delle performance migliori rispetto ad un semplice GA.

È possibile confrontare i due algoritmi ed è possibile prendere come indice di ottimizzazione il consumo medio di energia per ciclo e viene constatato in (11) che con il GPS si riesce ad ottenere una traiettoria (con manipolatore planare a 3GDL) che consumi il 40% di energia rispetto a quella ottenuta mediante GA.

Un altro metodo per affrontare il problema della minimizzazione della coppia è adottare un metodo di ottimizzazione che combina la norma due della velocità dei giunti (**MVN** Minimum Velocity Norm) e la coppia dei giunti (**MTN** Minimum Torque Norm) tramite due coefficienti di peso e incorpora i limiti fisici di giunto. Il tutto viene trasformato in un QP (Quadratic Program) al livello delle accelerazioni di giunto e risolto utilizzando una rete neurale basata su variazione lineare delle disuguaglianze (LVI) (12).

Oltre ai metodi puramente matematici per ricercare la soluzione ottimale per ridurre la coppia erogata dagli attuatori elettrici di un manipolatore ridondante un'altra linea di pensiero è quella di concepire nuovamente la **struttura meccanica** del manipolatore stesso (13). L'obiettivo di questo approccio è quello di ridurre il numero dei motori elettrici utilizzati per il controllo degli angoli di giunto ed ottimizzarne la posizione ottenendo una notevole riduzione del peso del manipolatore. Ovviamente questo approccio riducendo notevolmente masse e inerzie dei vari link comporta una notevole riduzione delle coppie necessarie per la movimentazione.

Ritornando all'approccio numerico, per l'ottimizzazione della coppia erogata dagli attuatori di un manipolatore ridondante con la classica struttura meccanica, un approccio alternativo a quello classico della pseudo-inversa è quello del Minimum Torque Trajectory Planning (**MTTP**) affrontato mediante un NonLinear Programming problem (NLP). I problemi NLP solitamente necessitano di una soluzione fattibile per iniziare il processo di ottimizzazione. Utilizzando come indice da minimizzare:

$$\mathbf{J} = \sum_{i=0}^{N-1} (\boldsymbol{\tau}(i) \cdot \boldsymbol{\tau}^{\boldsymbol{T}}(i)) \cdot \Delta t \tag{1.19}$$

Dove: $\boldsymbol{\tau}$ è il vettore delle coppie ai giunti;

Δt è l'intervallo di discretizzazione temporale.

e imponendo dei limiti su angolo di giunto iniziale, velocità di giunto iniziale e limiti di coppia erogabile (minima e massima) è possibile esplicitare l'equazione della dinamica in funzione dell'accelerazione, la velocità e posizione angolare dei giunti mediante integrazione numerica. L'algoritmo di risoluzione si basa sui seguenti step:

1) fissare un Δt di discretizzazione e il numero di soluzioni fattibili da passare all'algoritmo di risoluzione NLP;

2) formulare il problema in forma discreta tenendo conto dell'equazione dinamica e dei limiti imposti;
3) adottare l'approccio di Hollerbach per trovare le accelerazioni di giunto e l'intervallo Δt di campionamento;
4) utilizzare l'approccio di Hollerbach per scalare in funzione di un'unica variabile l'intero problema partendo dal minimo valore di tale variabile;
5) trovare le coppie dei giunti corrispondenti alle relative accelerazioni trovate;
6) risolvere il problema usando un algoritmo NLP per ottenere una soluzione di minimo locale;
7) ripetere la procedura fino a raggiungere il massimo della variabile di Hollerbach e valutare il minimo assoluto come minimo dei minimi locali (14).

Un tipico algoritmo utilizzabile per la risoluzione di un problema NLP è: **Generalized Reduced Gradient Method** che usa le direzioni dei gradienti come metodo per la ricerca della migliore soluzione.

Hollerbach osserva che il metodo di risoluzione della cinematica di un robot ridondante mediante l'uso della pseudo-inversa dello jacobiano (anche conosciuta come inversa generalizzata d Moore) comporta la minimizzazione del prodotto del vettore delle velocità di giunto per il suo trasposto. Assumendo che le velocità al quadrato siano proporzionali all'energia cinetica questo comporta anche una minimizzazione dell'energia. Altro approccio usato è quello della pseudoinversa pesata. Il limite di questo approccio è che è puramente cinematico, per questo, un approccio alternativo è quello che comporta l'utilizzo di una inversa generalizzata pesata con la matrice d'inerzia. Esplicitando l'equazione dell'accelerazione di giunto in funzione della pseudo-inversa ed introducendo dei limiti di coppia ai giunti. L'approccio usato nel paper è quello di minimizzare la distanza tra la coppia istantanea e la coppia media valutata come media tra coppia massima e minima erogabile (15).

Altra soluzione, al contrario di molti altri approcci usati in cui ci si propone di minimizzare la norma due del vettore delle coppie ai giunti pone l'attenzione sulla norma infinito del vettore tenendo conto anche dei limiti di coppia erogabile (16). Tale approccio, implementato mediante una rete neurale per avere una potenziale applicazione on-line, comporta la minimizzazione della coppia più grande del vettore in modulo il che è fisicamente più sensato della riduzione della norma due. Il metodo proposto consiste in:

1) Trasformare la norma infinita in un problema di minimizzazione lineare;
2) Mappare il problema lineare mediante una rete di Proiezione e Contrazione (PC).

1.6.3 Evitare le singolarità cinematiche

I due modi più comuni per evitare il problema delle singolarità del manipolatore sono:

1) evitare la situazione in cui il rango della matrice Jacobiana si riduce;
2) pianificare la traiettoria in modo che il manipolatore non si disponga mai in una condizione di singolarità.

Un primo metodo proposto per fornire una soluzione approssimata della cinematica inversa di robot non ridondanti intorno a delle configurazioni singolari è denominato **Singularity-robust**

inverse (SRI). Tale metodo consiste nell'introdurre un termine di regolazione nella matrice Jacobina per evitare il mal condizionamento della matrice in prossimità delle singolarità. Questo metodo utilizza i minimi quadrati smorzati (Damped Least Squares DLS) per permettere un controllo del moto approssimato vicino alla traiettoria nello spazio cartesiano desiderata. Questi metodi riducono la coppia applicata ma non permettono l'effettivo raggiungimento dei punti singolari e risultano instabili perché molto sensibili ai parametri selezionati. Per compensare questi problemi è possibile adottare un approccio per il controllo del percorso ibrido che sia stabile nell'intero spazio operativo anche in prossimità delle regioni che contengono singolarità. La stabilità del metodo è verificabile utilizzano il metodo delle funzioni di Lyapunov multiple (17).

Un passo avanti è quello fatto con l'**on-line task modification method** (OTMM). Tale metodo ho l'obiettivo di evitare le singolarità di manipolatori ridondanti e non a livello della velocità (18). Con questo metodo non è necessario verificare in fase di pianificazione della traiettoria se la singolarità è evitabile nel caso di manipolatori ridondanti ma si effettua direttamente online una modifica della traiettoria impostata. Il problema nasce dal fatto che, in prossimità di una configurazione ridondante, gli algoritmi di cinematica inversa basati sulla matrice Jacobina possono non funzionare perché essa ha rango minore del massimo e pertanto il vettore di velocità (task velocity) non può essere ottenuto mediante combinazione lineare dei vettori colonna della matrice Jacobiana. Possibile soluzione è quella classica dell'SRI.

Altra soluzione, valida sia per robot ridondanti che non ridondanti, è quella proposta in (19) che sfrutta la matrice Jacobiana. Tale approccio consiste nel fissare una soglia, quando il **determinante dello Jacobiano** scende sotto questa soglia si sostituisce la riga dello Jacobiano mal condizionata con il differenziale di uno tra i non tendenti a zero determinanti. Questo metodo dipende fortemente dalla forma della cinematica diretta.

Un differente approccio si basa sullo stabilire una condizione locale sufficiente per assicurare che il rango dello Jacobiano sia preservato (20).

Il metodo definito **forma normale** presentato in (21) utilizza il classico algoritmo di Newton lontano dalle singolarità per generare la traiettoria mentre, in prossimità delle configurazioni singolari la traiettoria non è generata nello spazio operativo ma nello spazio dei giunti. Infine, le due o più parti della traiettoria vengono unite. Il problema di base di questo approccio è che è computazionalmente molto oneroso.

Per un robot ridondante si possono evitare solo alcune configurazioni singolari, quelle definite evitabili, utilizzando un approccio off-line in fase di pianificazione della traiettoria (come descritto nei papers citati). Per evitare anche le singolarità definite inevitabili occorre sviluppare un approccio on-line. L'idea di base è circoscrivere le singolarità modificando direttamente la traiettoria nello spazio operativo.

Oltre al metodo **SVD** (Singular Value Decomposition) altre strade percorribili per evitare le singolarità sono: l'algoritmo di Greville, il metodo dello spazio nullo e il metodo della perturbazione. Questi metodi hanno lo svantaggio di essere computazionalmente pesanti per risolvere la cinematica inversa. Le più recenti ricerche si concentrano, invece, sull'utilizzo delle reti neurali artificiali. Utilizzare una strategia di switch è importante se si vogliono migliorare

le performance quando si risolve il problema cinematico inverso. L'idea di fondo è fare uno switch dal metodo della pseudo-inversa al metodo della perturbazione ottimale basandosi su una misura di manipolabilità espresso mediante il modulo del determinante del prodotto dello Jacobiano per la sua trasposta che tende a zero in prossimità di una configurazione singolare (22).

Altra soluzione alternativa al problema delle singolarità si basa sulla minima **norma infinito della velocità di giunto** per permettere al manipolatore di passare attraverso le singolarità senza eccessive velocità di giunto. L'idea di fondo di questo metodo è di pesare gli errori di tracciamento della traiettoria dell'EE e la norma delle velocità di giunto. Esso permette di creare continuità nella transizione tra configurazioni singolari e non. Alcuni autori propongono di esprimere il fattore di smorzamento come una funzione della misura di manipolabilità, in quanto, la manipolabilità è definita in base allo Jacobiano. Pertanto, la manipolabilità può essere usata come indice per indicare la vicinanza ad una configurazione singolare del manipolatore.

In ogni caso la maggior parte dei ricercatori che si concentrano sulla norma della velocità di giunto si basano sulla norma due. In realtà, utilizzare la norma infinito può essere più significativo, perché in questo modo si minimizza la massima tra le velocità di giunto (23).

Altri studi (24) osservano che non esistono solo le singolarità cinematiche ma anche le **singolarità algoritmiche**. Il problema delle singolarità algoritmiche permane sia se si utilizza la ridondanza del manipolatore mediante il metodo della task-space augmentation sia se si considera il task dell'EE come primario e si aggiungano altri task con priorità inferiore. Come nel caso delle singolarità cinematiche anche quelle geometriche comportano alte velocità di giunto e soluzioni mal condizionate. La soluzione damped least-square (DLS) con filtri numerici rappresenta un buon compromesso tra l'accuratezza nel tracciare la traiettoria tipica della soluzione derivante dall'uso della pseudo-inversa e del trovare soluzioni con velocità di giunto raggiungibili tipico del DLS.

Infine, alcuni studi (25) congiungono l'ottimizzazione della coppia con l'evitare le singolarità utilizzando un approccio basato sul DLS e considerando come indice di misura della distanza dalla condizione di singolarità la manipolabilità dinamica del manipolatore.

1.6.4 Incrementare distanza dai limiti di giunto

Evitare i limiti di giunto è tra le principali applicazioni dei manipolatori ridondanti in quanto, le variabili di giunto sono limitate da limiti fisici, se essi vengono violati si possono realizzare dei danni meccanici al manipolatore a seguito dell'urto dei giunti con i loro fine corsa. L'utilizzo della ridondanza di un manipolatore per evitare limiti di giunto è spesso raggiunto attraverso una ottimizzazione globale o locale di un indice di performance mentre si soddisfano le equazioni cinematiche relative alla traiettoria dell'EE. L'**ottimizzazione globale** richiede la conoscenza di tutte le informazioni della traiettoria in anticipo pertanto non è un metodo applicabile se sono contemplabili continue modifiche della traiettoria basate sui feedback dei sensori. L'**ottimizzazione locale**, invece, determina la traiettoria nello spazio dei giunti in modo che in ogni punto lungo la traiettoria dell'EE un certo indice di performance sia ottimizzato e sono ovviamente più adatti all'implementazione on-line.

Diversi metodi possono essere messi a confronto per la pianificazione della traiettoria nel rispetto dei limiti di giunto:

- Il **Gradient Projection Methond** (GPM) è largamente utilizzato in letteratura per utilizzare la ridondanza al fine di evitare i limiti di giunto. L'indice di performance è definito come una funzione dei limiti di giunto e il suo gradiente è proiettato nello spazio nullo dello Jacobiano per ottenere il self-motion necessario per ottimizzare l'indice di performance. Questo metodo però comporta la determinazione di un coefficiente scalare che determina il modulo del self-motion, può provocare oscillazioni nella traiettoria nello spazio dei giunti e instabilità;
- **Weighted least-norm** (WLN) consiste nel considerare la norma pesata delle velocità di giunto al fine di penalizzare il moto di alcuni giunti piuttosto che altri e poi di considerare la soluzione a norma minima;
- **Logica Fuzzy** basata su un metodo che sceglie automaticamente un appropriato modulo di self-motion usando regole linguistiche per evitare i limiti di giunto;
- Altro approccio può essere quello di **bloccare** completamente il **giunto** che ha raggiunto il suo limite e continuare a muovere solo gli altri.
- Altra tecnica ha mirato a risolvere il problema formulando un problema di minimizzazione come un **Quadratic Problem** (QP).

Un altro metodo usato (26) è il Prescribed Performance Control (**PPC**). Per un generico output si definisce una funzione chiamata funzione di performance che deve essere limitata, smooth e strettamente positiva. Più o meno il valore assunto da questa funzione nel tempo definisce una regione di performance all'interno della quale l'errore di evoluzione deve essere confinato. Una comune funzione di performance è un'esponenziale negativo.

Il paper (27) è quello che per primo ha proposto il **Gradient Projection Method** applicato ad un manipolatore ridondante con 7 gradi di libertà. Questo metodo evita il calcolo della pseudo-inversa dello Jacobiano e risulta efficiente per la determinazione delle velocità di giunto. Il metodo GPM ha il fine di sfruttare il self-motion dei giunti nel caso di manipolatori ridondanti. I due principali problemi associati a questo approccio sono la scelta di un coefficiente scalare e la possibilità di avere self-motion non necessari. Per migliorare l'effetto del GPM è possibile ricorrere all'utilizzo di un fattore di scala continuo. Nel paper citato, si propone di utilizzare una matrice di coefficienti per migliorare ancora l'effetto del GPM. Se il coefficiente utilizzato per il GPM è troppo piccolo i limiti di giunto potrebbero non essere evitati in tempo, se è troppo grande potrebbe esserci self-motion non necessario ed eccessivo. Per evitare eccessivo self-motion che va contro il principio di minimizzare il consumo di energia si introduce un fattore di amplificazione che è utilizzato per incrementare il coefficiente che pesa il GPM quando ci si avvicina ai limiti di giunto al di sotto di una certa soglia prestabilita. Però, questo fattore di amplificazione incide su tutti i giunti, quindi tutti saranno affetti da un rallentamento nel momento in cui uno solo di essi si sta avvicinando ad un limite di giunto e questo provoca eccessivo self-motion. Per questo motivo lo step successivo è sostituire all'unico coefficiente del GPM una matrice diagonale di coefficienti uno per ogni variabile di giunto. Questa matrice di coefficienti, in conclusione, non solo evita non necessario self-motion ma fa anche in modo

che la penalizzazione del moto di un giunto che si avvicina ai limiti di giunto non incida su tutti gli altri giunti (28).

L'articolo (29) si propone un confronto tra i metodi GPM e WLN. Esso presenta anche alcuni degli **indici di performance** per evitare i limiti di giunto più comuni. In particolare, due importanti indici sono:

1) la sommatoria dei quadrati del rapporto tra la differenza dell'angolo di giunto istantaneo ed il valore medio e la differenza tra il valore medio e il limite di giunto. Questo criterio misura la distanza dal valore medio del range in cui la variabile di giunto può variare ma non tiene traccia del superamento del limite di giunto di nessuna variabile di giunto;
2) un indice simile al precedente ma più articolato in modo che alle l'indice sia unitario quando tutti i giunti si trovano nel mezzo del range di variabilità e vada ad infinito quando anche una sola delle variabili di giunto superi uno dei due limiti (massimo o minimo).

Per quanto concerne il **WLN**, la cosa più semplice per introdurre un peso nella norma delle velocità di giunto è quello di utilizzare una matrice diagonale. Ciascun elemento della diagonale principale della matrice di peso è studiato in modo che vada ad infinito in corrispondenza delle due estremità del range della variabile di giunto e vada a zero nel punto medio di tale range. Inoltre, bisogna valutare anche il gradiente di tale elemento della diagonale, in quanto, se un giunto si sta avvicinando al limite di giunto occorre penalizzare il movimento di quel giunto, d'altro canto, invece se si sta allontanando non c'è nessun motivo per penalizzarlo. Il vantaggio di usare il metodo WLN con una matrice di peso diagonale è che l'inversa di una matrice diagonale è facile da calcolare e quindi rispetto al GPM questo metodo risulta assai meno pesante computazionalmente.

In alternativa al GPM e al WLN c'è il metodo basato sulla logica fuzzy (30). Il **fuzzy-logic based method** (FLM) supera il problema del GPM di un eccessivo modulo del self-motion. Un set di regole linguistiche determinano gli elementi del vettore self-motion evitando self-motion non necessario e oscillazioni. Invece di ottimizzare una funzione specifica, si usa la logica fuzzy che consiste dei seguenti componenti:

1) fuzzification interface: realizza la funzione di fuzzification che consiste nel convertire i dati in input in idonei valori linguistici che possono essere viste come etichette di fuzzy sets;
2) knowledge base: comprende una base di conoscenza del dominio di applicazione e dei relativi obiettivi. Questo fornisce le definizioni necessarie in termini di etichette, che sono usate per definire un set di regole linguistiche;
3) decision making logic: è il cuore del sistema fuzzy, ha il potere di determinare le azioni;
4) defuzzification interface: a partire dalla distribuzione fuzzy di output dà in uscita un non-fuzzy output.

Questa soluzione si presenta come un'alternativa al GPM ma dipende fortemente come esso dalla scelta del coefficiente di scala tipico del GPM quindi necessita di lavoro ulteriore per essere ottimizzato al meglio.

1.6.5 Evitare collisione

È bene distinguere due diverse applicazioni:

1) generazione di una traiettoria per evitare la collisione del manipolatore con **oggetti fermi**;
2) generazione di una traiettoria e controllo real-time per evitare la collisione con **oggetti mobili**.

Occorre distinguere queste due applicazioni in quanto, in base alla metodologia utilizzata per perseguire questo scopo variano i tempi di esecuzione e conseguentemente variano anche le possibili applicazioni.

Il parametro più comunemente adottato in questo campo risulta essere la distanza di uno o più punti del manipolatore da uno o più oggetti presenti nello spazio operativo ma ci sono ulteriori approcci che valutano anche altri parametri fondamentali come: inerzia, velocità relativa, parte del corpo urtata ecc.

Un approccio molto analizzato in letteratura è quello che fa uso degli **algoritmi genetici** (GA). Tale approccio prevede prima di tutto la determinazione di uno spazio operativo limitato all'operazione richiesta che generi un parallelepipedo avente diagonale il punto iniziale ed il punto finale della traiettoria che si vuole determinare. Determinato tale volume si traccia una griglia di possibili configurazioni in cui si può disporre il manipolatore. La griglia deve essere sufficientemente fitta da eliminare la possibilità che un oggetto di possa disporre tra una configurazione e l'altra. L'approccio utilizzato consiste nel minimizzare la somma della distanza tra due configurazioni adiacenti e la distanza tra la configurazione intermedia da raggiungere allo step successivo rispetto alla configurazione finale del manipolatore. L'algoritmo genetico spesso utilizza l'Hermen cubic interpolation method che permette il passaggio da una configurazione intermedia alla successiva con un polinomio del terzo ordine. Il tempo di esecuzione dell'algoritmo è di alcune decine di secondi pertanto presumibilmente non è adatto per operazioni real-time ma studia una traiettoria che ottimizza il tempo di esecuzione del task evitando gli ostacoli presenti nello spazio operativo. Tale algoritmo può essere adottato per qualsiasi configurazione cinematica del robot, non importa che esso sia ridondante (31).

Altra soluzione è quella del **QPSO** (Quantum Behaved Particicle Swarm Optimization) un algoritmo più rapido del GA ed ha potenziale applicazione online. Un metodo per procedere con tale tecnica consiste nel considerare una funzione di costo costituita da 4 indici, ognuno con il proprio peso, che deve essere minimizzata (32).

Il metodo **SQP** (Sequential Quadratic Programming) è un metodo iterativo per l'ottimizzazione di vincoli non lineari. Tale metodo può essere adottato al fine di ottimizzare diversi obiettivi come ad esempio: massimizzare la distanza dai limiti di giunto, mantenere in un range prefissato la velocità dei giunti e massimizzare la distanza di un punto del manipolatore da un ostacolo esterno. Tale metodo, come i precedenti, evita l'utilizzo del metodo della pseudo-inversa ma presenta alcuni svantaggi non trascurabili: difficoltà di bilanciare i costi e evitare la collisione, in alcuni casi la collisione non può essere evitata, in certe situazioni il robot deve avvicinarsi troppo all'ostacolo (33).

Il metodo **Deapth Space Approach** è quello adottato all'Università Sapienza di Roma, al contrario dei precedenti i quali presuppongono nota la posizione (fissa o mobile) degli ostacoli da evitare e si concentrano sulla generazione di una opportuna traiettoria integra le due cose. Infatti, tale metodo utilizza un Kinect che da una informazione 2-3 dimensionale. Lo studio è stato avanzato su un KUKA a 7 gradi di libertà controllato in posizione e pertanto 4 volte ridondante utilizzando un algoritmo di potenziale che genera un campo attrattivo verso il target e un campo repulsivo verso gli ostacoli. L'algoritmo di generazione della traiettoria utilizzato è tra i più rapidi di quelli elencati perché genera un controllo ogni pochi millisecondi (34) permettendo una reale implementazione online.

Altro metodo che si distingue dai precedenti perché non valuta esclusivamente la distanza del manipolatore dall'ostacolo ma ne considera anche la velocità relativa è quella del **CDF** (Cumulative Danger Field) tale approccio è stato ottimizzato nel corso degli anni da studi condotti da ricercatori del Politecnico di Milano.

1.6.6 Massimizzare / Minimizzare Forza

Esistono un gran numero di indici per misurare le performance dinamiche di un manipolatore. In particolare:

- **ME** (Manipulability Ellipsoid): l'ellissoide di manipolabilità è introdotto per valutare le performance statiche di un manipolatore come un indice che lega le velocità angolari dei vari giunti alla velocità dell'EE;
- **MFE** (Manipulability Force Ellipsoid): l'ellissoide di manipolabilità di forza lega invece la trasmissione di forza-coppia statica dai giunti all'EE;
- **DME** (Dynamic Manipulability Ellipsoid): l'ellissoide di manipolabilità dinamica è una misura della performance dinamica del manipolatore basata sulla massima accelerazione dell'EE;
- Il **Compatibility Index** ed il numero di condizionamento dello Jacobiano sono proposti per scegliere posture ottimali per un certo task;
- **IME** (Inertial Manipulability Ellipsoid): l'ellissoide di manipolabilità inerziale caratterizza l'efficienza di trasmissione di forza-coppia dagli attuatori dei giunti all'EE e da esso si possono ricavare il DME e il MFE.

Questi sono alcuni dei principali strumenti di cui si dispone per l'analisi della manipolabilità ed in particolare della forza di un manipolatore. Poi però ci sono una gran varietà di possibili applicazioni come, ad esempio, la minimizzazione della forza di impatto del manipolatore con un oggetto esterno, il controllo della forza di contatto ecc.

Nell'articolo (35) si introduce il concetto di IME come indice di misura dell'efficienza di trasmissione di forza-coppia dagli attuatori dei giunti all'EE come estensione dei classici concetti di ME, MFE e DME largamente utilizzati e noti. L'autore confronta l'accoppiamento inerziale del manipolatore seriale all'accoppiamento di un attuatore o di un sistema di ingranaggi. In un sistema di ingranaggi il concetto di accoppiamento inerziale è ampiamente diffuso per individuare l'ottimale rapporto di trasmissione basandosi sulla performance di trasmissione della coppia prodotta dall'attuatore e quella trasmessa al carico. In modo analogo lo Jacobiano può essere usato come il rapporto di trasmissione negli ingranaggi per misurare la

trasformazione tra la velocità di giunto e quella dell'EE. Siccome lo Jacobiano per i manipolatori è analogo al rapporto di trasmissione per gli ingranaggi, il concetto di accoppiamento inerziale può essere esteso al caso di manipolatore seriale come un indice dell'efficienza di trasmissione tra la coppia prodotta dagli attuatori e la forza applicata dall'EE al carico. A fronte di questa idea si esplicita quindi l'equazione del moto di un manipolatore e l'equazione del moto del carico tenuto dall'EE e si ricava l'accelerazione dell'EE stesso. Con una serie di passaggi riportati nel paper si giunge all'indice che misura l'efficienza tra la coppia prodotta da ogni giunto e la forza e il momento applicati al carico dall'EE che possiamo definire l'indice di accoppiamento inerziale. A questo punto si può anche introdurre i limiti di coppia per ogni attuatore (ipotizzati simmetrici) e si arriva all'Inertial Manipulability Ellipsoid (IME).

Molto utile nel campo della robotica collaborativa può essere un articolo di Walker, Rice University, Texas (36). L'articolo analizza il problema di ridurre i potenziali effetti dannosi causati dall'interazione di un manipolatore robotico con l'ambiente che li circonda. In particolare, si concentra sull'utilizzo della ridondanza del manipolatore per configurare il meccanismo in modo da ridurre il modulo delle **forze impulsive** create all'istante del **contatto**. Queste forze, infatti, oltre a danneggiare eventuali oggetti/persone presenti nell'ambiente in cui opera il manipolatore si trasmettono sul manipolatore stesso e possono pregiudicarne il funzionamento. Tale ricercatore ha introdotto un modello per analizzare il fenomeno dell'impatto tra manipolatore ed oggetto. (37) Ha proposto un primissimo modello di impatto per un sistema guidato su un singolo asse in cui si effettuano delle analisi di vibrazione. (38) Presenta un metodo geometrico di impatto nel caso planare. Il metodo include attrito, effetti inerziali ed elasticità di due soli oggetti in collisione. Questi due metodi però non sono di validità generale per un manipolatore robotico ben più complesso. (39) Riporta, invece, un primo modello di impatto che incorpora l'intera dinamica di un manipolatore spaziale poi anche sviluppato nel caso di due manipolatori da altri autori. Successivamente, è stato sviluppato anche un modello di collisione tra due corpi nel caso più complesso in cui c'è l'interazione tra due diversi manipolatori. In tal caso, si ipotizzano entrambi i corpi rigidi, si ipotizza nota la velocità prima dell'impatto e la variazione di velocità a seguito dello stesso. Per considerare la possibilità di un urto plastico o elastico si introduce un semplice coefficiente e variabile tra 0 ed 1. In seguito, si calcola la forza impulsiva che si sviluppa nel contatto tra i due corpi ipotizzando l'urto in un intervallo di tempo infinitesimo. Con una serie di relazioni che tengono conto della dinamica del manipolatore e alcune approssimazioni si riesce a calcolare la forza impulsiva che agisce sul manipolatore nel momento del contatto. Ovviamente la forza impulsiva risulta in ogni caso essere funzione delle velocità precedenti all'urto, della geometria degli oggetti e delle configurazioni assunte dal/dai manipolatori all'istante dell'impatto. Quindi, considerando come indice di riferimento la forza impulsiva di impatto è possibile interrogarsi su quale configurazione è meglio far assumere al robot ridondante per minimizzare tale parametro. Tale applicazione è valida tanto per impatto con oggetti inanimati (ad es. pareti o altri robot) tanto per urto con persone.

Un'altra applicazione dei manipolatori ridondanti è nel controllo della **forza di contatto**. Per fare un efficace controllo della forza del manipolatore si può utilizzare un sensore di coppia/forza a 6 assi. I dati raccolti possono essere inviati ad un controllore dell'anello esterno che implementa un controllo di impedenza o di forza per ognuno dei sei assi del sistema di

riferimento del tool. Il controllore di impedenza del secondo ordine fa agire l'EE come un sistema massa-molla-smorzatore con parametri modificabili e fornisce una forza di set, una inerzia e uno smorzamento desiderati per migliorare le performance in transitorio. Nel loop più interno c'è un controllore PD, un modulo per risolvere la ridondanza al livello dell'accelerazione e un modulo per la dinamica inversa. L'utilizzo di un controllore PD nel loop di controllo più interno migliora le performance del controllo forza del manipolatore. Questo filone di ricerca basato sul controllo della forza del manipolatore non è molto noto ed utilizzato ad oggi per i seguenti motivi:

1) complessità degli schemi che richiede complessi hardware di controllo;
2) piccoli ritardi nel loop di controllo forza possono causare instabilità quindi occorrono frequenze di campionamento elevate;
3) molti schemi di controllo si basano sulla dinamica inversa che si basa sull'accuratezza dei parametri dinamici del manipolatore;
4) effetti di accoppiamento tra posizione e forza che possono portare ad instabilità nella pianificazione della traiettoria.

Ciononostante, è stato sviluppato descrive un controllore di forza adattativo applicato ad un manipolatore ridondante (40). Per adottarlo, prima di tutto occorre definire la traiettoria nello spazio operativo poi si applica un controllore di accelerazione ad ogni giunto e si determina la massa del manipolatore nello spazio dei giunti arbitrariamente.

Nello specifico della robotica collaborativa, esiste uno studio mirato a limitare la forza di contatto ed evitare il danneggiamento dell'ambiente circostante e soprattutto ferire persone (41). Diversi metodi sono stati sviluppati ma questi presuppongono nota la dinamica e questo comporta anche una stima degli effetti di attrito difficili da determinare.

1.6.7 Minimizzare reazioni vincolari

È un tipico problema delle applicazioni spaziali. Infatti, in questo campo le reazioni vincolari del manipolatore sono scaricate sulla base e possono comportare una rotazione non facilmente arrestabile dell'intera base spaziale.

Nel tempo sono stati sviluppati un **approccio globale** ed un **approccio locale** per la minimizzazione del modulo della forza e della coppia delle reazioni alla base. Alcune ricerche dimostrano che l'approccio globale è più efficace nel ridurre e nel rendere più smooth la forza alla base mentre l'approccio locale è migliore per la riduzione del momento applicato alla base del manipolatore.

Se si adotta un approccio locale a questo problema (42) si definisce un indice di performance o una funzione di costo che viene minimizzata ad ogni intervallo di tempo. Sebbene il vantaggio evidente è quello di possedere un algoritmo di controllo che funzione in real time, un problema dell'approccio locale è che ci sono dei picchi indesiderati della reazione alla base. D'altro canto, un metodo di ottimizzazione globale (43) consiste nell'ottimizzazione di un indice di performance lungo tutta la traiettoria compiuta dal manipolatore. Pertanto, esso è adottabile solo se la traiettoria è pianificata a priori e successivamente eseguita.

La minimizzazione delle reazioni vincolari è solo una delle infinite soluzioni che possono essere adottate per compensare eventuali movimenti e rotazioni della base spaziale a fronte del movimento del robot (44). È possibile, infatti, procedere almeno in altri due modi diversi: uso di getti a reazione oppure movimentare il manipolatore in modo che i suoi movimenti compiano il task desiderato ma contemporaneamente mantengano la stabilità dell'intero sistema. Per questa seconda opzione è presentato un elenco di soluzioni esaustivo: conservazione del momento angolare e lineare, uso di un volano di compensazione alimentato da energia fotovoltaica ecc. È possibile anche utilizzare un manipolatore ridondante ed il suo self-motion per la realizzazione di un percorso ottimale usando un approccio variazionale basato sulla conservazione del momento.

L'ottimizzazione globale di una funzione di costo lungo tutta la traiettoria pianificata dall'EE può essere eseguita con un approccio che integra il metodo dello Jacobiano Partizionato per la risoluzione di un manipolatore ridondante e l'uso di una 4-3-4 traiettoria di giunti per minimizzare una funzione di costo che rappresenta l'integrale nel tempo del modulo delle reazioni alla base (45). Al contrario di molti dei problemi di ottimizzazione presentati in precedenza questo è un problema dinamico e non puramente cinematico. Il vettore di forze/coppie alla base può essere espresso mediante la nota equazione dinamica di equilibrio del manipolatore:

$$\mathbf{M}(\mathbf{q})\ddot{\boldsymbol{q}} + \boldsymbol{b}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{T} \tag{1.20}$$

Dove: $\boldsymbol{q}$ è il vettore delle coordinate di giunto;

$\mathbf{M}(\mathbf{q})$ è la matrice d'inerzia simmetrica non singolare;

$\boldsymbol{b}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ è il vettore delle coppie dovute all'azione centripeta, Coriolis e attrito;

$\boldsymbol{g}(\boldsymbol{q})$ è il vettore delle coppie dovute alla gravità:

$\boldsymbol{T}$ è il vettore delle coppie di giunto.

Al fine di minimizzare le reazioni vincolari occorre definire una funzione di costo del vettore delle forze che è una funzione di posizione, velocità e accelerazione di giunto. Pertanto, la minimizzazione della funzione di costo si traduce nella determinazione delle variabili di giunto che minimizzino questa funzione di costo.

L'approccio locale si basa innanzitutto sulla descrizione delle velocità di giunto dei giunti ridondanti mediante delle relazioni polinomiali nel tempo. In particolare, in questo approccio si considera la variabile tempo discretizzata in intervalli uguali. Fatto questo è possibile racchiudere in una matrice di coefficienti del polinomio caratterizzanti le velocità dei giunti ridondanti e sostituire il tutto nelle equazioni della velocità e dell'accelerazione di giunto. In questo modo, per ogni intervallino di tempo si possono conteggiare posizione, velocità ed accelerazione di giunto in funzione degli elementi della matrice C che esprimono i coefficienti dei polinomi indicanti le velocità di giunto. E pertanto per ogni configurazione del manipolatore le reazioni vincolari sono una funzione degli elementi di questa matrice.

Nell'approccio globale non c'è nessun campionamento temporale e non è possibile usare un polinomio che descriva l'intera traiettoria perché occorrerebbe avere un grado troppo alto

pertanto si usa la tecnica 4-3-4. Ovvero, si divide la traiettoria dei giunti in 3 segmenti in cui il primo è caratterizzato da un polinomio del 4 ordine, il secondo da un polinomio del terzo ordine e il terzo da un altro polinomio del 4 ordine. In questo modo è possibile soddisfare, con polinomi di basso ordine, tutti i vincoli di velocità ed accelerazione che tipicamente si hanno nei punti iniziale e finale. Inoltre, il metodo delle traiettorie di giunto 4-3-4 può essere semplificato riducendolo a 3 parametri caratteristici: il fattore temporale, il fattore di spostamento angolare, l'angolo di giunto finale. Conoscendo 3 parametri per ogni grado di ridondanza si può esprimere una rappresentazione dei giunti basata sul metodo 4-3-4. Fatto questo, si può costruire una funzione di costo, dipendente da questi 3 parametri per ottimizzare globalmente le reazioni vincolari alla base del manipolatore imponendo anche dei limiti per ognuno di questi 3 parametri.

Per la determinazione della funzione di costo si può considerare l'integrale nel tempo della somma dei moduli della forza e della coppia alla base del manipolatore pesati con due opportuni coefficienti per l'approccio globale. Mentre per l'approccio locale basta considerare la somma pesata dei moduli senza integrare.

1.6.8 Biomimetica

Negli ultimi due decenni i robot sono diventati sempre più simili agli uomini, introducendo così la necessità di avere dei **movimenti antropomorfici** dei manipolatori per permettere una migliore interazione tra uomo e macchina.

I primi studi che sono stati condotti in questo campo riguardavano l'osservazione di operazioni quotidiana degli uomini e la riproduzione con un manipolatore (46) (47). Nella maggior parte di questi studi avanzati per imitare i movimenti di un braccio umano si usa un sistema per catturare i movimenti e lo si riproduce. Il problema di questo metodo è che non permette di generare ex novo dei movimenti simili a quelli di un braccio umano con il manipolatore ma solo di riprodurre quanto acquisito in opportune banche dati. Altro approccio è quello delle funzioni di costo che sono state proposte per modellare i principi di movimento di un braccio umano, poi sono stati introdotti altri criteri cinematici o dinamici ma che non permettono il moto del braccio in uno spazio 3D e ancora molti altri metodi sono stati presentati: HMM (Hidden Markow Models) sono stati usati per modellare il movimento di un braccio attraverso imitazione. Possibile interessante applicazione della biomimetica è quello della **robotica riabilitativa**. È possibile, ad esempio, pensare ad esoscheletri che possono essere applicati ad arti feriti per aiutarli a compiere i corretti movimenti. Inoltre, la biomimetica può farci comprendere quale sia il principio di ottimizzazione del cervello umano nel posizionamento del nostro corpo e soprattutto dei nostri arti. Altra comune applicazione di robot che si basano sulla biomimetica sono quelli disponibili nei parchi a tema per fini di **intrattenimento**.

Uno dei primi studi (48) si propone di **riprodurre** una **persona che compie un disegno** o sta scrivendo mediante un manipolatore a 7 DOF. Nell'esecuzione di un'operazione di questo tipo occorrono 3 gradi di libertà pertanto il manipolatore (così come il braccio umano) risulta 4 volte ridondante. Quindi, per risolvere la cinematica inversa, si adotta un algoritmo ad anello chiuso per la risoluzione delle ridondanze mediante task imposti al moto interno con l'approccio del "Virtual End-effector" per mimare i movimenti del braccio umano. Poi, le traiettorie del

manipolatore sono confrontate con quelle compiute dall'uomo ed acquisite mediante un opportuno sistema di telecamere. In particolare, le traiettorie di polso e gomito per minimizzare l'azione della forza di gravità devono essere vicino al piano di scrittura/disegno, per sfruttare i gradi di ridondanza si impongono le traiettorie di polso e gomito per simulare quelle di uno uomo mediante il principio dell'EE virtuale con traiettorie pianificate nello spazio operativo.

Il primo step per realizzare quanto proposto è quello di effettuare una pianificazione della traiettoria basata sulla visione attiva. L'occhio umano si muove acquisendo immagini 3 o 4 volte al secondo concentrandosi in un disegno su intersezioni di linee e rapide variazioni dell'andamento della linea stessa (spigoli). Per scegliere la traiettoria in base alle informazioni visive si utilizza una tecnica di processamento delle immagini derivante dal filtro di Gabo. In seguito, occorre risolvere la cinematica inversa tenendo conto dei gradi di ridondanza. Si utilizza un algoritmo CLIK (Closed Loop Inverse Kinematic) e si implementa un algoritmo di priorità a tre strati. Nel primo si considera la corrispondenza tra le velocità di giunto e la velocità del gomito calcolando la cinematica inversa con la matrice Jacobiana pseudo-inversa. Nello strato centrale si fa la stessa cosa per la velocità di polso imponendo una scelta delle variabili di giunto tanto più vicino possibile a quelle trovate allo step 1. Infine, si definisce una funzione obiettivo considerando l'intero problema CLIK risolto mediante lo spazio nullo. In questo modo, con alcune simulazioni si arriva a dimostrare una vicinanza tra le variabili di giunto del manipolatore e gli effettivi movimenti fatti dal braccio di una persona durante l'esecuzione di alcuni disegni.

Altro utilizzo della biomimetica si propone di analizzare il coordinamento simultaneo di diversi movimenti e la dipendenza tra certi angoli di giunto che esistono negli arti umani per emularli con i manipolatori e migliorare la sempre crescente interazione uomo-macchina (49). Lo studio si distingue da molte ricerche nel campo analizzato per un motivo principale ovvero quello di non imitare un predeterminato task di un braccio umano minimizzando la differenza nel movimento tra braccio umano e braccio robotico ma il metodo presentato tenta di **modellizzare** matematicamente le **caratteristiche antropomorfiche** di un **braccio umano** definendo una funzione obiettivo basata sulla dipendenza tra i vari giunti e usando un metodo probabilistico. E si testa il modello costruito su alcuni movimenti "nuovi". La prima fase di questa ricerca consiste nell'eseguire una campagna sperimentale per acquisire dati relativi al movimento di un braccio umano nello spazio al fine di poter sviluppare un modello che tenga conto della dipendenza tra i vari moti delle variabili di giunto delle nostre braccia considerando l'arto superiore con 5 DOF (3 nella spalla e 2 nel gomito). Per registrare gli angoli di giunto si è usato un sistema di tracciamento della posizione magnetico. Si è posizionato un tracciatore di posizione sul gomito e uno sul polso. È possibile costruire un modello della dipendenza delle variabili di giunto usando un Gaussian Mixture Model (**GMM**) che sarebbe una somma pesata di funzioni di distribuzioni gaussiane che descrive in modo abbastanza efficiente delle complesse funzioni di distribuzione di probabilità che legano tra loro le variabili di giunto. Queste dipendenze tra le variabili di giunto possono essere formulate come funzioni obiettivo per uno schema di cinematica inversa di un manipolatore. Usando la pseudo-inversa scelgo le velocità di giunto dello spazio nullo in modo da ottimizzare una funzione obiettivo che tenga conto delle configurazioni antropomorfiche più probabili valutate mediante un vettore delle funzioni densità di probabilità di ogni giunto.

Il **Bio-Inspired Approach** (BIA), al contrario dei classici metodi basati sulla riproduzione di uno specifico task (e pertanto molto limitati) pone maggiore enfasi su delle primitive di movimento umano. Questi metodi BIA cercano di risolvere la ridondanza cinematica soddisfacendo limiti e funzioni di costo basati su osservazioni biologiche. In particolare, il consumo dell'energia metabolica e la stabilità posturale sono le due più comuni funzioni di costo che vengono minimizzate passandole ad algoritmi di pianificazione della traiettoria. Un limite però di questo metodo è nell'affidabilità di una risoluzione cinematica basata sullo Jacobiano che non funziona in corrispondenza o in prossimità di configurazioni singolari. Strada alternativa è quella della **sinergia dei motori**. Questo metodo è stato usato per spiegare come il CNS (Central Nervous System) umano controlla una struttura complessa e ridondante come le nostre mani. I movimenti del braccio umano sono considerati come combinazione di due sinergie. Ogni sinergia segue un moto anatomicamente definito pertanto disponendo di due sinergie ognuna delle quali è controllata da una parabola indipendente caratterizzata da una certa curvatura si hanno solo due variabili (le due curvature) per controllare un manipolatore a 7 DOF. Quindi, applicare il concetto delle sinergie dei giunti, causa una semplificazione nel controllo di un manipolatore ridondante (50).

Anche alcune delle più comuni tecniche adottate per la movimentazione di **robot umanoidi** per intrattenimento, usati per riprodurne i movimenti secondo un approccio biomimetico, si basano sulla riproduzione dei movimenti (51). In queste tecniche il primo step rimane quello di catturare il moto di uno o più persone e questo può essere fatto, ad esempio, con un sistema che cattura dati sul movimento di persone mediante un sistema chiamato Vicon. Questo sistema costituito da 8 telecamere permette di misurare il movimento di tutto il corpo come gli angoli del polso e della testa ma non movimenti delle dita ed espressioni facciali. In seguito, le informazioni acquisite sono modificate tenendo conto dei limiti delle variabili di giunto e delle loro velocità al fine di permettere la riproduzione dei movimenti da parte dei robot umanoidi. Infine, si usa un sistema di controllo della traiettoria per seguire i movimenti registrati mediante il manipolatore. L'indice che viene utilizzato per misurare la corrispondenza tra i movimenti umani e quelli del robot è legato all'errore nel posizionamento spaziale dei vari segmenti corporei. Interessante aspetto trattato nell'articolo presentato è l'implementazione dei limiti del robot in alcune configurazioni in termini di variabili di giunto e di angoli raggiunti. Per permettere la corretta esecuzione dei vari movimenti si scalano posizioni e velocità di ogni giunto al fine di farli rientrare nei rispettivi limiti preservando però una corretta mimica del movimento umano.

Altri ricercatori (52), trattano quali siano le tipiche caratteristiche del movimento umano. Una tra le principali caratteristiche è la morbidezza nel movimento (smoothness), un'altra tipica caratteristica è la correlazione tra la velocità e l'accuratezza, più veloce viene eseguito un certo task meno accurata risulta la posizione finale della mano. Per simulare la morbidezza nei movimenti umani come indice di ottimizzazione nella scelta dei possibili movimenti da attuare c'è la minimizzazione del jerk. Un modello di questo tipo, puramente cinematico, è applicabile però non considera le asimmetrie nel profilo di velocità che in realtà si registrano nel braccio umano. Altro modello proposto definisce una funzione obiettivo the minimizza la variazione di coppia ai giunti. Il modello in questione nelle aree di fronte al corpo e in sua vicinanza porta a delle traiettorie simili a quelle proposte dal precedente metodo basato sul jerk ma in posizioni

laterali rispetto al corpo e alle estremità del campo di mobilità dà risultati molto differenti. Pertanto, si dimostra una forte correlazione dei movimenti prodotti dal braccio umano dalle zone dello spazio operativo in cui opera. Inoltre, i due metodi citati non tengono conto dei limiti dei movimenti realizzabili. Altri ricercatori suggeriscono che uno dei parametri con maggiori impatto è l'effetto di un **rumore neurale** sull'accuratezza dei movimenti. In sostanza, la deviazione standard del rumore aumenta proporzionalmente con il valore assoluto del segnale sul quale si somma, per questo, un segnale di movimento rapido (che risulta in modulo maggiore) ha un rumore più alto e quindi una minore accuratezza. Sulla base di questa osservazione è applicabile una minimizzazione della variazione della posizione dell'EE per raggiungere la posizione finale con buona accuratezza. In questo modo, anche se non si vincolano i movimenti del braccio ad essere morbidi si ottengono lo stesso movimenti regolari e una distribuzione di velocità a campana che assomiglia a quella del nostro arto superiore. Per implementare questa minimizzazione della variazione della posizione dell'EE possibile usare il quadratic programming e risolverlo con Matlab. In seguito, è stato proposto di usare il filtro di Kalman, altri autori hanno usato le reti neurali. Questi approcci sono complessi ma applicabili a configurazioni complesse e ridondanti.

1.6.9 Incrementare affidabilità rispetto ai guasti

Quella che viene definita come **fault-tolerance** è una vasta ed importante applicazione dei manipolatori ridondanti. L'obiettivo è quello di permettere al manipolatore di realizzare il percorso richiesto anche a seguito del guasto di uno o più giunti sfruttando i giunti funzionanti per mantenere comunque la traiettoria desiderata dell'EE. Il verificarsi di un guasto anche su un solo giunto può mettere a rischio l'esecuzione dell'intero task ed addirittura causare il danneggiamento meccanico del robot stesso.

L'utilizzo di un manipolatore ridondante tollerante ai guasti risulta essenziale in alcune applicazioni in cui la sicurezza è cruciale come ad esempio: aeroplani, velivoli spaziali, esplorazione di profondità oceaniche, impianti chimici in cui si opera con sostanze pericolose ed in tutte quelle applicazioni in cui la continuità e la sicurezza delle operazioni sia fondamentale.

La maggior parte delle ricerche effettuate in passato si basano sul metodo della pseudo-inversa che può incontrare il problema delle singolarità cinematiche quando avviene il guasto di uno o più giunti (53). Un algoritmo di pianificazione della traiettoria che evita configurazioni sfavorevoli del manipolatore prima che si verifichi un guasto è stato sviluppato. Siccome però, l'ottimizzazione globale è computazionalmente onerosa non può essere utilizzata per il controllo real-time del movimento che solitamente richiede modifiche della traiettoria in base alle informazioni ricevute come feed-back dai sensori.

Per compensare al problema dei guasti sono principalmente due le strade percorribili:

1) intervento a livello di **design del manipolatore**;
2) intervento a livello di **controllo**.

Alla prima categoria fanno capo le proposte di utilizzo di particolari strutture seriali dei manipolatori o all'uso di manipolatori paralleli. A livello di controllo si possono usare approcci di Fault Detection, Fault Isolation and Identification e Safety Issues.

Un algoritmo per l'esecuzione di un task pianificato anche a seguito del gusto di uno o più giunti (fino a 3) si può basare sullo sviluppo di un **quadratic program** (QP) che incorpora anche i limiti sulle variabili di giunto e sulle velocità di giunto (54). Per la risoluzione real-time del quadratic program è possibile utilizzare un risolutore numerico basato su Linear Variational Inequalities (LVI). L'approccio QP è utilizzato per la risoluzione della cinematica inversa dei manipolatori ridondanti tenendo conto anche sia di limiti simmetrici che asimmetrici per le variabili di giunto e le sue velocità. Un Quadratic Program è equivalente a un sistema LVI che può essere efficientemente risolto mediante diversi metodi: numerico o reti neurali. L'uso di una formulazione QP è molto interessante a causa di diversi motivi:

1) i limiti fisici dei giunti possono essere direttamente incorporati nella formulazione QP;
2) in una formulazione QP si possono introdurre diversi indici di performance con diversi scopi di ottimizzazione;
3) il processo di risoluzione di un QP gode di proprietà asintotiche, adattative ed iterative.

Con tale tecnica si può conservare una traiettoria predefinita anche con un basso errore di scostamento dalla traiettoria programmata anche nel caso di guasto di 3 giunti per un manipolatore 6 DOF cui è assegnato il task di seguire una traiettoria nel piano. Si osservi che la tecnica proposta è quella di implementare un problema QP che possa risolvere la cinematica inversa di un manipolatore ridondante e permettere l'esecuzione di un task anche a seguito del guasto di uno o più giunti e di risolvere il problema così formulato mediante un algoritmo LVI basato sull'E47 che permette l'implementazione on-line del problema.

Un perfezionamento di questo filone di studi si ha con delle ricerche in cui ci si concentra sul problema di **mantenimento**, non solo della traiettoria ma anche della **forza** richiesta all'EE in caso di guasto (55). Per raggiungere questo obiettivo, occorre effettuare una mappatura della coppia che i giunti funzionanti devono applicare per compensare quella non applicabile dai giunti guasti. Fatto questo l'obiettivo del paper è minimizzare il salto di forza che si verifica all'EE a causa dell'avvenuto guasto. Questo obiettivo di preservare la forza fornita all'EE viene presentato usando il modello della matrice di perturbazione. Considerando un generico giunto bloccato per effetto di un guasto l'approccio del modello di perturbazione consiste nel considerare prima di tutto quale sia la variazione di forza all'EE e scrivendo la relazione:

$$\boldsymbol{\tau} + \Delta\boldsymbol{\tau} = (\boldsymbol{J} + \Delta\boldsymbol{J})^{\boldsymbol{T}}(\boldsymbol{F} + \Delta\boldsymbol{F}) \tag{1.21}$$

A questo punto, ci si pone l'obiettivo di minimizzare il salto di forza e per farlo bisogna utilizzare lo spazio nullo della matrice Jacobiana. Il paper mostra anche un esempio di applicazione del problema presentato applicandolo ad un manipolatore 5 DOF.

Un'altra ricerca (56) pone l'obiettivo di **ridisegnare** un manipolatore ridondante per essere tollerante a guasti dei giunti. Ottimo qui viene inteso in termini di indice di manipolabilità relativo valutato nel caso peggiore. Sebbene questa tecnica è applicabile sia per manipolatori seriali che paralleli viene usata tipicamente per meccanismi paralleli e con limitato spazio operativo. L'indice di manipolabilità relativa, ad esempio, ha l'obiettivo di quantificare la tolleranza ai guasti di un manipolatore ridondante. Tale parametro, quindi, è la misura della destrezza del manipolatore a seguito di uno o più guasti ai giunti e varia tra 0 e 1, esso vale 0

quando a seguito del/dei guasto/i l'EE perde completamente mobilità, 1 se continua a muoversi allo stesso modo. Usando questo indice è possibile definire quale sia la forma che deve assumere la matrice Jacobiana per rispondere in modo ottimale al guasto di uno o più giunti e quindi il design più opportuno di un manipolatore ridondante.

1.6.10 Minimizzare le vibrazioni

È un problema tipicamente sviluppato nel campo di **manipolatori ridondanti flessibili**. Il problema (57) non si presenta in manipolatori in cui si fa l'ipotesi di link rigidi perché si basa sul controllo di un manipolatore flessibile in modo da evitare le vibrazioni causate dai movimenti nello spazio nullo che, nel caso di manipolatore flessibile, si ripercuotono sul movimento dell'EE.

Altra **applicazione** è quella dei manipolatori in ambiente **spaziale** (58). Interessante è lo studio del problema di riduzione delle vibrazioni causate dal movimento del manipolatore (rigido) o dall'impatto di piccoli detriti spaziali sulle appendici flessibili (ad esempio antenne o pannelli solari). In questa applicazione di fondamentale importanza è la modellazione dell'appendice flessibili che viene fatta solitamente mediante modelli agli elementi finiti che non sono però utilizzabili per controllo on-line. Per un controllo online si utilizza un modello proposto da Yoshikawa basato sulla modellazione di un manipolatore con appendici flessibili mediante una serie di link rigidi e molle. Questo metodo è basato sul controllo delle vibrazioni di un'appendice modellizzata mediante la presenza di giunti virtuali che permette l'alternanza di link rigidi e molle. Per il controllo di questo sistema è necessario un feedback dai sensori dello stato dell'appendice flessibile. Siccome questo è di difficile implementazione gli autori propongono l'uso di un indice che stimi lo stato dell'appendice flessibile basandosi sul feedback di un sensore di forza/coppia posizionato alla base dell'appendice stessa. Per eliminare l'effetto delle vibrazioni introdotte dall'appendice si usa un controllo che impone l'accelerazione ai giunti del manipolatore tali da annullare l'effetto vibratorio generato dall'appendice con una trattazione analitica ben spiegata nel paper citato.

2 CAPITOLO 2: SVILUPPO DEL MODELLO CINEMATICO/DINAMICO DI UN MANIPOLATORE A 7 GDL

Per eseguire delle prove al computer che simulino il reale comportamento di un manipolatore robotico è possibile adottare un modello cinematico di un manipolatore ridondante a 7 gradi di libertà avente le caratteristiche del manipolatore **KUKA LBR iiwa R820**. È possibile sviluppare tale modello secondo le convenzioni tipiche della robotica che fanno ricorso ai parametri di Denavit-Hartenberg e pianificarne la traiettoria mediante i principi ricavati dallo studio del manuale (1). Infine, per una stima delle coppie erogate dai motori dello stesso è possibile aggiungere anche i parametri inerziali del manipolatore e ricorrere all'utilizzo di un algoritmo ricorsivo basato sulle equazioni del moto con la formulazione di Newton Eulero. In tale capitolo, si ripercorre tale processo di modellizzazione eseguito per il manipolatore KUKA ma ripetibile per qualsiasi manipolatore robotico.

2.1 Parametri di Denavit-Hartenberg

Un manipolatore robotico può essere descritto come una catena cinematica aperta costituita dall'alternanza di link e giunti che ne permettono il movimento relativo. Per risolvere l'equazione cinematica diretta di un manipolatore occorre definire la posizione e orientazione relativa tra due link consecutivi. Per farlo si utilizzano i **parametri di Denavit-Hartenberg**. Per descrivere tali parametri fondamentali di riferimento si può utilizzare la seguente figura.

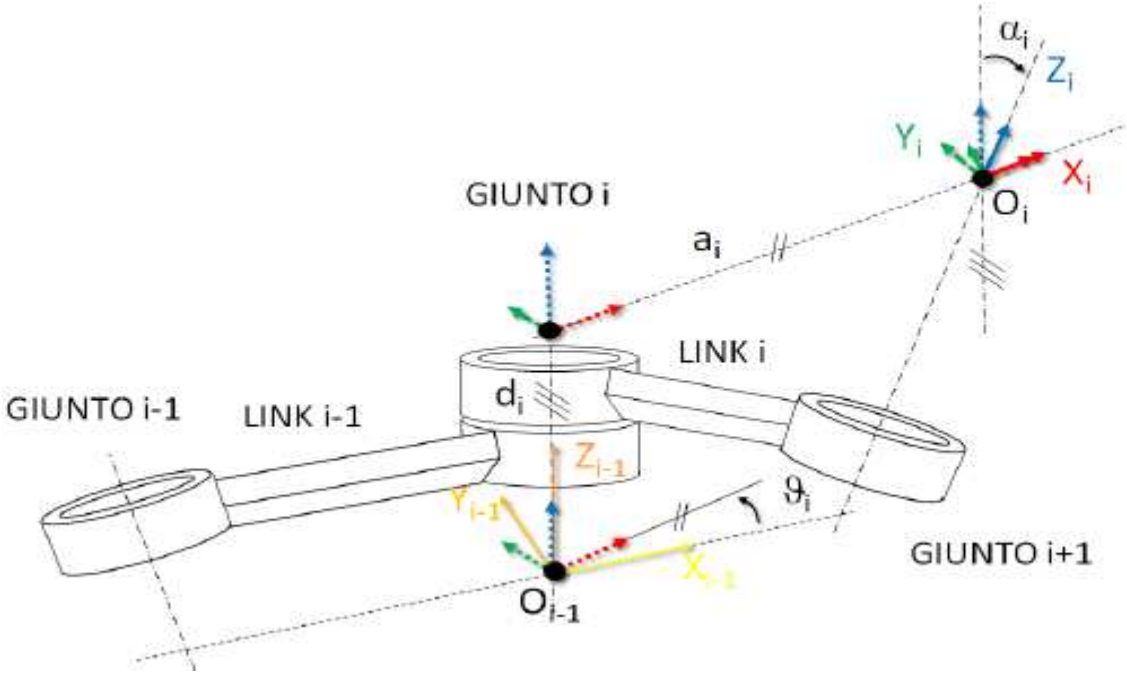


Figura 16 Parametri di Denavit-Hartenberg, convenzione standard

Si indichi con i l'asse del giunto che connette i link i e $i-1$. È necessario definire una terna destrorsa per determinare i quattro parametri di Denavit-Hartenberg.

Per il generico link i si deve definire l'asse z_i di tale terna in modo che esso giaccia lungo l'asse del giunto di rotazione $i+1$ definendo il verso in base al verso di rotazione ipotizzato positivo per il giunto di rotazione. Successivamente, occorre definire l'asse x_i, per farlo, è necessario tracciare la normale comune (ovvero la retta cui appartiene il segmento di minima distanza tra due rette sghembe) tra gli assi dei giunti i e $i+1$. Fatto ciò, l'asse x_i ha direzione coincidente con quella della normale comune e verso positivo da i verso $i+1$. Infine, si definisce l'asse mancante y_i in modo da costituire una terna levogira. Ovviamente, l'origine O_i di tali terne di riferimento coincide con il punto di intersezione tra l'asse z_i e x_i. A questo punto è possibile determinare i parametri di Denavit-Hartenberg. Per determinare queste grandezze ci si avvale di una terna di riferimento ausiliaria $x_{i'} - y_{i'} - z_{i'}$. Si comincia disponendo questa terna ausiliaria coincidente con il sistema di riferimento $x_{i-1} - y_{i-1} - z_{i-1}$. Il primo parametro che si può definire è θ_i indicando con tale denominazione l'angolo intorno all'asse z_{i-1} di cui bisogna ruotare l'asse $x_{i'}$ per coincidere con x_i. Poi è possibile definire il parametro d_i come la distanza di cui bisogna traslare l'origine $O_{i'}$ del sistema di riferimento $x_{i'} - y_{i'} - z_{i'}$ lungo $z_{i'}$ affinché l'asse $x_{i'}$ giaccia sulla stessa retta su cui giace x_i. Indicando con $x_{i''} - y_{i''} - z_{i''}$ il nuovo sistema di riferimento, ruotato e traslato, è possibile indicare con a_i la traslazione lungo l'asse $x_{i''}$ che bisogna compiere affinché l'origine $O_{i''}$ coincida con O_i. Indicando ora con $x_{i'''} - y_{i'''} - z_{i'''}$ il nuovo sistema di riferimento ottenuto mediante questa traslazione possiamo definire l'ultimo dei quattro parametri di Denavit-Hartenberg come α_i ovvero l'angolo, intorno a $x_{i'''}$, di cui bisogna ruotare la terna $x_{i'''} - y_{i'''} - z_{i'''}$ affinché l'asse $z_{i'''}$ coincida con l'asse z_i. A questo punto per indicare posizione e orientazione del sistema di riferimento i rispetto al sistema di riferimento $i-1$ è possibile utilizzare la **matrice di trasformazione omogenea**:

$$\boldsymbol{A}_i^{i-1}(q_i) = \boldsymbol{A}_{i'}^{i-1}\boldsymbol{A}_i^{i'} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i)\cos(\alpha_i) & \sin(\theta_i)\sin(\alpha_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i)\cos(\alpha_i) & -\cos(\theta_i)\sin(\alpha_i) & a_i\cos(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

In cui le matrici $\boldsymbol{A}_{i'}^{i-1}$ e $\boldsymbol{A}_i^{i'}$ sono rispettivamente le matrici di trasformazione omogenea dal sistema di riferimento i' al sistema di riferimento $i-1$ e dal sistema di riferimento i al sistema di riferimento i'. Oltre a queste matrici di trasformazione omogenea tra un link ed il successivo è ovviamente anche necessario esplicitare le matrici di trasformazione della base rispetto al sistema di riferimento world (sistema di riferimento assoluto) e dell'EE rispetto al sistema di riferimento dell'ultimo giunto. Pertanto, in conclusione, utilizzando il manipolatore KUKA LBR iiwa R820 come riferimento per la nostra trattazione è possibile esplicitare la posizione ed orientazione dell'EE nel sistema di riferimento assoluto mediante la relazione:

$$\boldsymbol{A}_{EE}^0(q_i) = \boldsymbol{A}_B^0\boldsymbol{A}_1^B\boldsymbol{A}_2^1\boldsymbol{A}_3^2\boldsymbol{A}_4^3\boldsymbol{A}_5^4\boldsymbol{A}_6^5\boldsymbol{A}_7^6\boldsymbol{A}_{EE}^7 \tag{2.2}$$

Solitamente è possibile considerare il sistema di riferimento della base coincidente con quello assoluto, a meno di particolari necessità, mentre, per la determinazione della matrice di trasformazione omogenea $\boldsymbol{A}_{EE}^7$ è possibile usare la **convenzione RPY** (Roll Pitch Yaw) che

consiste nel ricostruire l'orientazione dell'EE rispetto al sistema di riferimento dell'ultimo link mediante composizione di tre matrici di rotazione intorno agli assi x_0, y_0 e z_0 rispettivamente. Indicando gli angoli di rotazione intorno a tali assi, rispettivamente con ψ, χ e φ. La matrice che esprime l'orientazione dell'EE rispetto al sistema di riferimento del link 7 risulta:

$$\boldsymbol{R}(\psi,\chi,\varphi) = \boldsymbol{Rot}(z,\varphi)\boldsymbol{Rot}(\mathrm{y},\chi)\boldsymbol{Rot}(x,\psi) = \\ = \begin{bmatrix} c_\varphi c_\chi & c_\varphi s_\chi s_\psi - s_\varphi c_\psi & c_\varphi s_\chi c_\psi + s_\varphi s_\psi \\ s_\varphi c_\chi & s_\varphi s_\chi s_\psi + c_\varphi c_\psi & s_\varphi s_\chi c_\psi - c_\varphi s_\psi \\ -s_\chi & c_\chi s_\psi & c_\chi c_\psi \end{bmatrix} \tag{2.3}$$

Utilizzando una notazione per cui $c_\psi = \cos(\psi)$ e $s_\psi = \sin(\psi)$.

Volendo costruire la matrice di trasformazione omogenea ed avendo per ora fissato l'orientazione dell'EE occorre anche definire la posizione dell'EE rispetto al sistema di riferimento assoluto e ciò è possibile mediante il vettore posizione:

$$\boldsymbol{p}_{EE}^0 = [p_{EE}(x_0) \quad p_{EE}(y_0) \quad p_{EE}(z_0)]^T \tag{2.4}$$

Pertanto, la matrice di trasformazione omogenea che identifica posizione ed orientazione dell'EE nel sistema di riferimento assoluto risulta:

$$\boldsymbol{A}_{EE}^0 = \begin{bmatrix} \boldsymbol{R}(\psi,\chi,\varphi) & \boldsymbol{p}_{EE}^0 \\ \boldsymbol{0} & 1 \end{bmatrix} \tag{2.5}$$

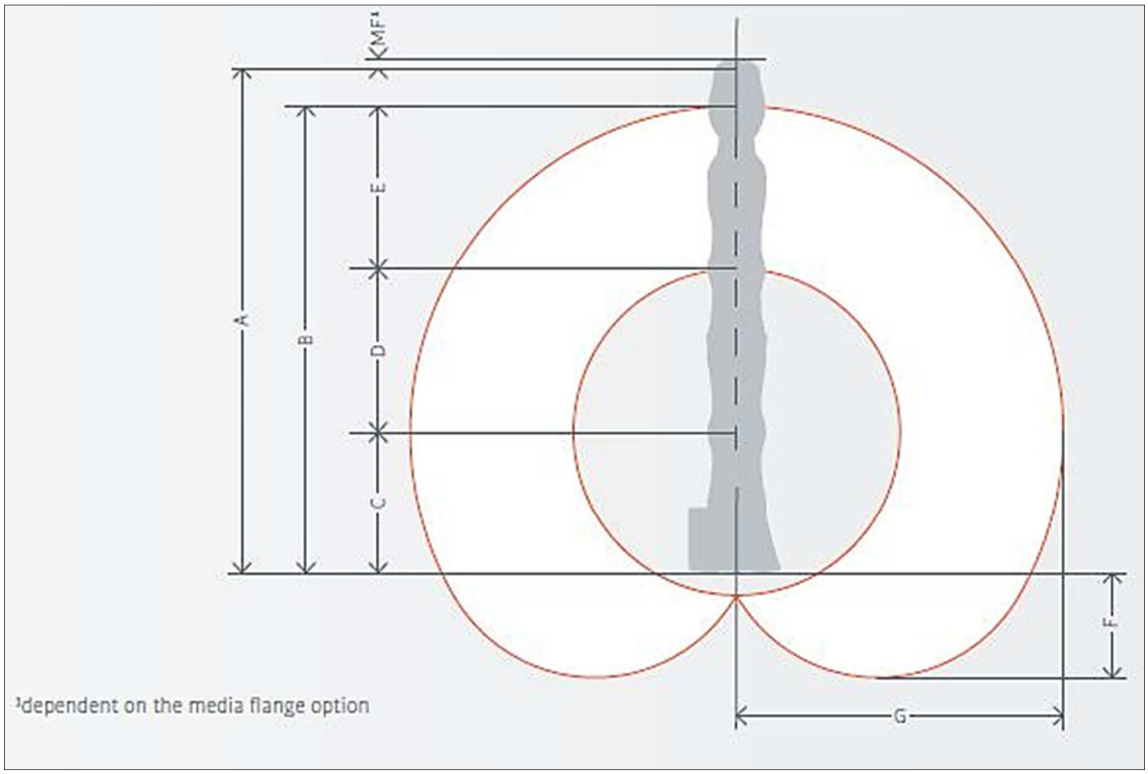
2.2 Informazioni caratteristiche KUKA LBR iiwa R820

Per costruire un modello analitico di un manipolatore robotico in ambiente virtuale è necessario raccogliere una serie importante di dati geometrici ed inerziali.

Il manipolatore KUKA LBR iiwa R820 è un braccio robotico dotato di 7 giunti rotoidali e quindi una volta ridondante nello spazio. Per ricavare i **dati geometrici** di tale manipolatore si è fatto riferimento al sito ufficiale del produttore.

Portata	14 kg
Carico totale nominale	14 kg
Max. raggio d'azione	820 mm
Numero di assi comandabili	7
Ripetibilità di posizionamento (ISO 9283)	±0,15 mm
Peso	30 kg
Posizione(i) di montaggio	Parete / Pavimento / Soffitto
Temperatura ambiente	Da 5 °C a + 45 °C

Tabella 1 Informazioni generali sul manipolatore KUKA LBR iiwa R820



Working envelope	Dimensions A	Dimensions B	Dimensions C	Dimensions D	Dimensions E	Dimensions F	Dimensions G	Volume
LBR iiwa 7 R800	1,266 mm	1,140 mm	340 mm	400 mm	400 mm	260 mm	800 mm	1.7 m³
LBR iiwa 14 R820	1,306 mm	1,180 mm	360 mm	420 mm	400 mm	255 mm	820 mm	1.8 m³

Figura 17 Dimensioni caratteristiche KUKA LBR iiwa R820

Attraverso questi dati geometrici è possibile ricavare tutte le matrici di trasformazione omogenea citate precedentemente applicando la convenzione di Denavit-Hartenberg.

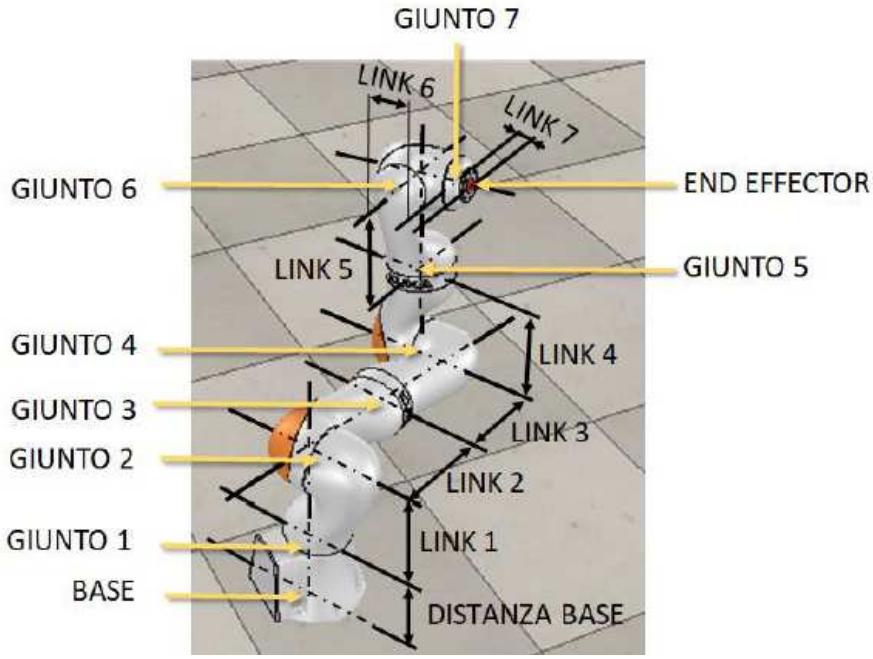


Figura 18 Schematizzazione dei giunti e dei link del manipolatore KUKA LBR iiwa R820

Dalla Figura 18 è possibile vedere i giunti caratteristici del manipolatore. In particolare, si distinguano alcuni dei giunti caratteristici del manipolare. Oltre la base, vale la pena notare che il giunto 2 del manipolatore costituisce quello che comunemente è chiamata spalla, il giunto 4, invece, è detto gomito mentre i giunti 5, 6 e 7 costituiscono il polso del robot. Infine, il giunto 7 costituisce l'EE del manipolatore al quale può essere collegato un eventuale tool finale mediante apposita flangia di collegamento (gripper, pistola di verniciatura, ventosa ecc.).

Oltre a tali informazioni geometriche per sviluppare un modello corretto del manipolatore analizzato occorre anche valutare quali siano i **limiti** di escursione angolare dei giunti del manipolatore e conseguentemente anche i limiti di velocità angolare massima raggiungibile e coppia massima erogabile.

		LBR IIwa 7 kg		LBR IIwa 14 kg	
Axis data / range of motion		Maximum torque	Maximum velocity	Maximum torque	Maximum velocity
Axis 1 (A1)	± 170°	176 Nm	98°/s	320 Nm	85°/s
Axis 2 (A2)	± 120°	176 Nm	98°/s	320 Nm	85°/s
Axis 3 (A3)	± 170°	110 Nm	100°/s	176 Nm	100°/s
Axis 4 (A4)	± 120°	110 Nm	130°/s	176 Nm	75°/s
Axis 5 (A5)	± 170°	110 Nm	140°/s	110 Nm	130°/s
Axis 6 (A6)	± 120°	40 Nm	180°/s	40 Nm	135°/s
Axis 7 (A7)	± 175°	40 Nm	180°/s	40 Nm	135°/s

Tabella 2 Limiti angolari, di velocità e coppia KUKA LBR iiwa R820

Quello che manca sono le **informazioni inerziali** del manipolatore che non vengono direttamente fornite sul catalogo del produttore e che pertanto devono essere ricavate in modo alternativo come verrà analizzato in seguito.

2.3 Trascrizione dei parametri di Denavit-Hartenberg

Come accennato nel paragrafo precedente conoscendo quali siano le dimensioni caratteristiche del manipolatore di riferimento utilizzato è possibile costruirne un modello matematico adottando la convezione di Denavit-Hartenberg.

Link	**i-1 rispetto i**	$\boldsymbol{\alpha_i}$ [°]	$\boldsymbol{a_i}$ $[\boldsymbol{m}]$	$\boldsymbol{d_i}[\boldsymbol{m}]$	$\boldsymbol{\theta_i = \theta_{i_0} + q_i}$ [°]
1	B-1	0	0	0.360	q_1
2	1-2	-90	0	0	q_2
3	2-3	90	0	0.420	q_3
4	3-4	90	0	0	q_4
5	4-5	-90	0	0.400	q_5
6	5-6	-90	0	0	q_6
7	6-7	90	0	0.163	q_6

Tabella 3 Parametri di Denavit-Hartenberg (convenzione standard) KUKA LBR iiwa R820

Link	**i-1 rispetto i**	$\boldsymbol{\alpha_i}$ [°]	$\boldsymbol{a_i}$ $[\boldsymbol{m}]$	$\boldsymbol{d_i}[\boldsymbol{m}]$	$\boldsymbol{\theta_i}$ [°]
Base	0-B	0	0	d_B	0
Tool	7-EE	0	0	d_T	0

Tabella 4 Parametri di Denavit-Hartenberg (convenzione standard) per la base e tool

Si noti che nella Tabella 4 si riportano con la dicitura d_B e d_T due dimensioni indicative della base dell'eventuale tool applicato al manipolatore.

Utilizzando il tool Matlab di Peter Corke per i robot è possibile riportare una rappresentazione schematica ma efficace del manipolatore descritto corredato dei **sistemi di riferimento** adottati di qui in avanti per la trattazione eseguita.

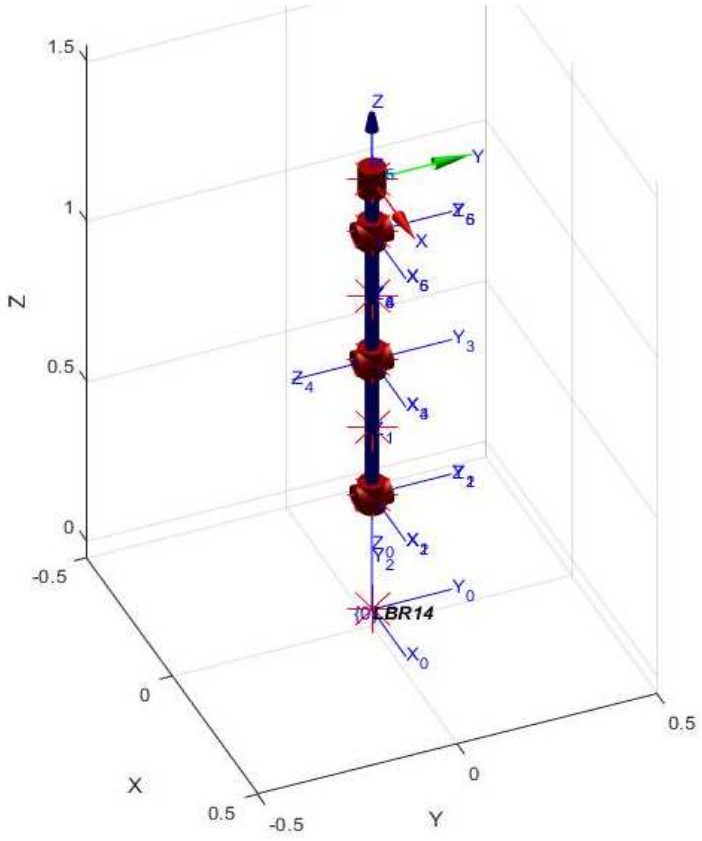


Figura 19 Sistemi di riferimento dei link utilizzati per robot KUKA LBR iiwa R820

Infine, dai parametri di Denavit-Hartenberg individuati è anche possibile ricavare le **matrici di trasformazione omogenea** necessarie per proseguire con la presente trattazione e con la successiva pianificazione della traiettoria del manipolatore. Si riportano ora le matrici di trasformazione omogenea ricavate:

$$\boldsymbol{A_1^B} = \begin{bmatrix} \cos(q_1) & -\sin(q_1) & 0 & 0 \\ \sin(q_1) & \cos(q_1) & 0 & 0 \\ 0 & 0 & 1 & 0.360 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

$$\boldsymbol{A}_2^1 = \begin{bmatrix} \cos(q_2) & 0 & -\sin(q_2) & 0 \\ \sin(q_2) & 0 & cos(q_2) & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

$$\boldsymbol{A}_3^2 = \begin{bmatrix} \cos(q_3) & 0 & \sin(q_3) & 0 \\ \sin(q_3) & 0 & -\cos(q_3) & 0 \\ 0 & 1 & 0 & 0.420 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

$$\boldsymbol{A}_4^3 = \begin{bmatrix} \cos(q_4) & \sin(q_4) & 0 & 0 \\ \sin(q_4) & -\cos(q_4) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.9}$$

$$\boldsymbol{A}_5^4 = \begin{bmatrix} \cos(q_5) & 0 & -\sin(q_5) & 0 \\ \sin(q_5) & 0 & cos(q_5) & 0 \\ 0 & 0 & 1 & 0.400 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

$$\boldsymbol{A}_6^5 = \begin{bmatrix} \cos(q_6) & 0 & -\sin(q_6) & 0 \\ \sin(q_6) & 0 & cos(q_6) & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

$$\boldsymbol{A}_7^6 = \begin{bmatrix} \cos(q_7) & \sin(q_7) & 0 & 0 \\ \sin(q_7) & -\cos(q_7) & 0 & 0 \\ 0 & 0 & 1 & 0.163 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.12}$$

Infine, le matrici di trasformazione omogenea della base rispetto al sistema di riferimento world e quella dell'EE rispetto al sistema di riferimento del giunto 7 risultano:

$$\boldsymbol{A}_B^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_B \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.13}$$

$$\boldsymbol{A}_{EE}^7 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_T \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

2.4 Pianificazione della traiettoria

Per i test condotti nel corso del presente lavoro di tesi si utilizzano due semplici traiettorie del manipolare pianificate nello spazio operativo. La prima consiste nel mantenere il manipolatore in una certa posa ed intervenire mediante lo spazio nullo per compensare il grado di ridondanza del manipolatore nello spazio. La seconda traiettoria pianificata è un movimento lineare dell'EE nello spazio operativo lungo uno specifico asse.

2.4.1 Mantenimento posa prefissata del manipolatore

Per il **mantenimento** della **posa** del manipolatore occorre definire una certa posizione e orientazione dell'EE. Fissando un certo vettore di angoli di giunto ed adottando l'equazione cinematica diretta è possibile ricavare la conseguente posizione dell'EE, inoltre, adottando gli angoli di Eulero e la convenzione RPY è possibile anche definirne l'orientazione. In questo modo, come descritto nel paragrafo 2.1 si può determinare la matrice di trasformazione omogenea che indica la posizione e orientazione dell'EE rispetto al sistema di riferimento assoluto. E pertanto, ad ogni istante temporale vengono imposte come variabili di giunto di set al manipolatore le medesime variabili senza particolari complicazioni.

2.4.2 Esecuzione traiettoria lineare

Per realizzare con il manipolatore un **segmento nello spazio**, invece, occorre definire il punto iniziale di partenza ed il punto finale, rispettivamente, tali punti nello spazio saranno indicati dai vettori $\boldsymbol{p}_i$ e $\boldsymbol{p}_f$. A questo punto, occorre dare una **rappresentazione parametrica della curva**. Questo può essere fatto con un'ascissa curvilinea s variabile da 0 ad 1. Utilizzando questo espediente, un segmento nello spazio può essere espresso mediante la relazione seguente:

$$\boldsymbol{p}(s) = \boldsymbol{p}_i + \frac{s}{\|\boldsymbol{p}_f - \boldsymbol{p}_i\|}(\boldsymbol{p}_f - \boldsymbol{p}_i) \tag{2.15}$$

Se si desidera percorrere questa traiettoria è necessario imporre dei vincoli sul vettore posizione e sulla matrice di orientazione. In particolare, indicando con t_i l'istante iniziale e t_f l'istante finale:

- Vincoli di posizione: $\boldsymbol{p}(t_i) = \boldsymbol{p}_i\ ,\ \boldsymbol{p}(t_f) = \boldsymbol{p}_f$
- Matrice di orientazione: $\boldsymbol{A}(t_i) = \boldsymbol{A}_i\ ,\ \boldsymbol{A}(t_f) = \boldsymbol{A}$

Il parametro s deve invece rispettare i seguenti vincoli: $s(t_i) = 0\ ,\ s(t_f) = 1$. Comunemente, si pianificano le traiettorie lineari con un profilo di velocità variabile secondo un trapezio. Pertanto, l'ascissa curvilinea deve variare in modo da rispettare i seguenti andamenti:

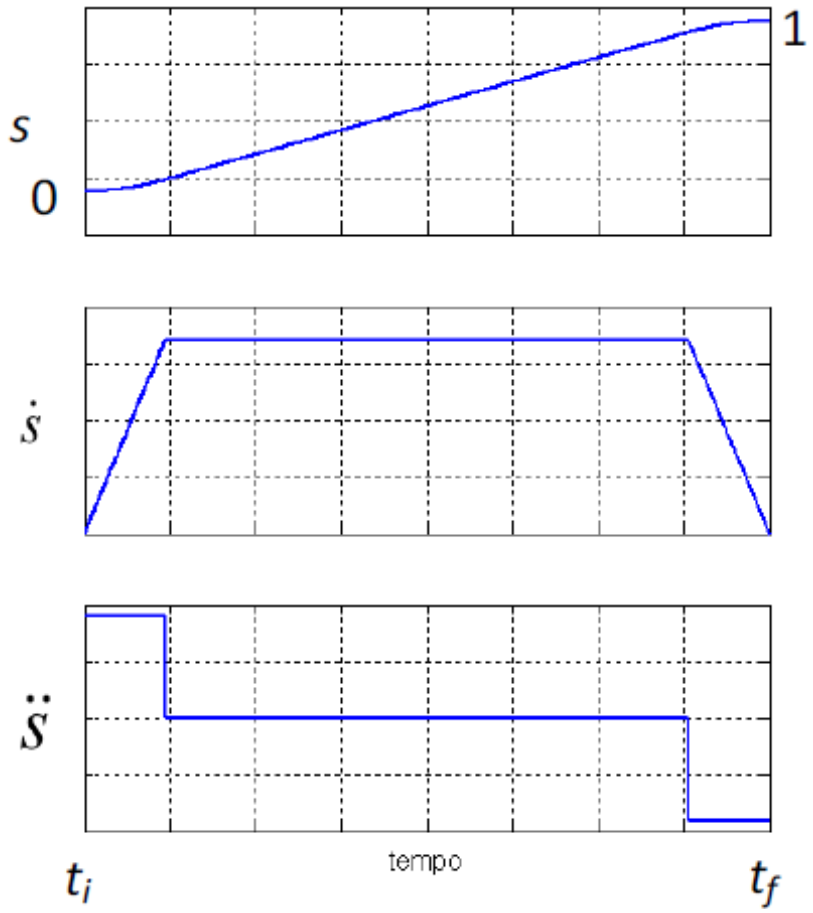


Figura 20 Andamenti caratteristici di un parametro di traiettoria con trapezio di velocità

Questa particolare traiettoria è sostanzialmente costituita da tre tratti, nel primo l'EE accelera, nel secondo viene mantenuta una velocità costante, ed infine, nell'ultimo si ha una decelerazione.

2.5 Metodo ricorsivo per il calcolo delle coppie erogate dai giunti

L'obiettivo principale della determinazione dei **parametri inerziali** del robot è quello di utilizzare tali dati per effettuare una stima delle coppie erogate dai giunti e quindi dall'eventuale dispendio energetico necessario per la sua alimentazione. Per effettuare tale stima è necessario utilizzare un algoritmo ricorsivo. Tale algoritmo si basa sulle equazioni di Newton-Eulero e viene utilizzato per determinare le forze ed i momenti agenti su ogni link del manipolatore semplicemente partendo dai valori di forze e momenti applicati sull'EE. Si riporti uno schema rappresentativo dell'algoritmo ricorsivo:

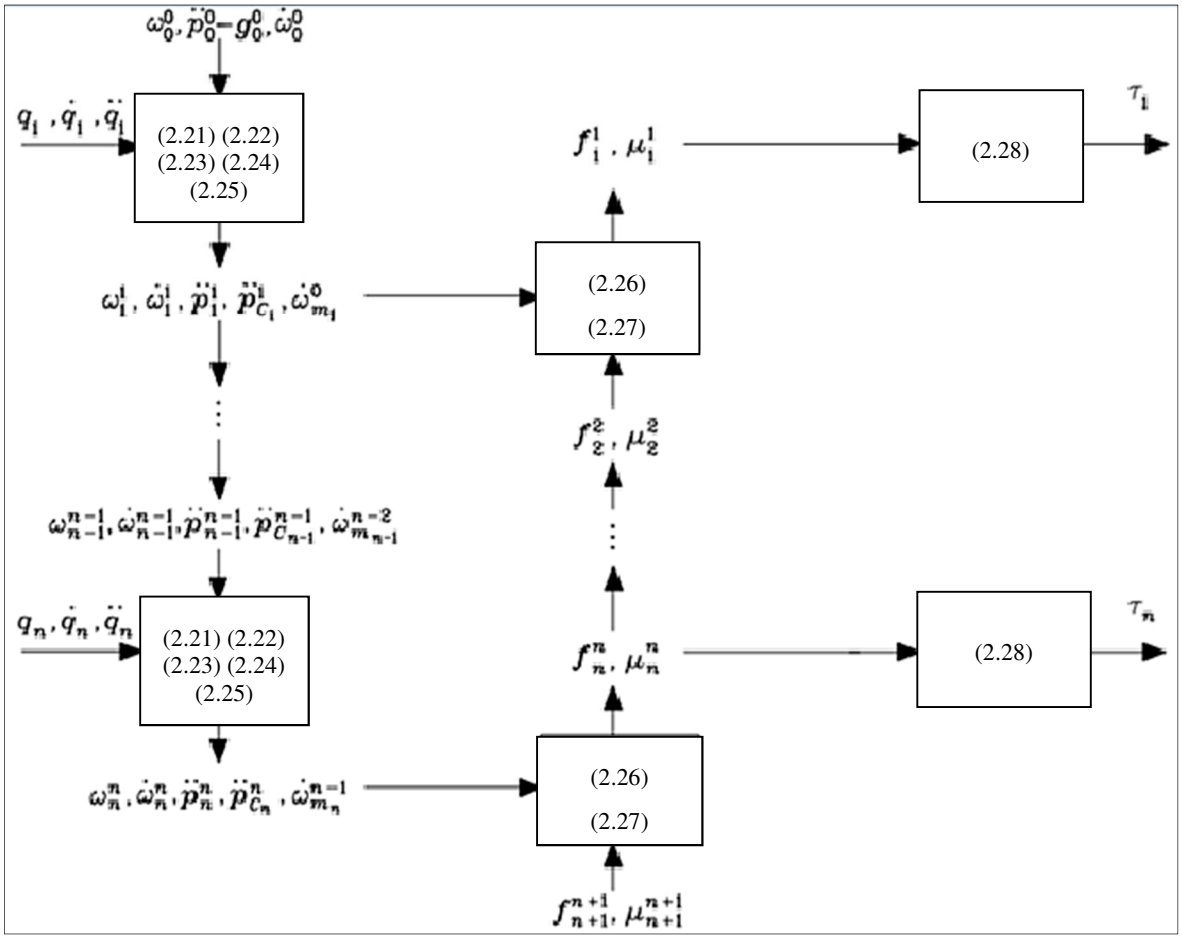


Figura 21 Schema concettuale algoritmo ricorsivo

Si indichi con:

- $q_i, \dot{q}_i, \ddot{q}_i$ la i-esima variabile di giunto e le sue derivate prima e seconda;
- $\boldsymbol{\omega}_i^i, \dot{\boldsymbol{\omega}}_i^i$ velocità ed accelerazione angolare dell'i-esimo link nell'i-esimo sistema di riferimento;
- $\boldsymbol{p}_i, \dot{\boldsymbol{p}}_i, \ddot{\boldsymbol{p}}_i$ posizione, velocità ed accelerazioni lineari dell'origine dell'i-esima terna;
- $\dot{\boldsymbol{p}}_{C_i}, \ddot{\boldsymbol{p}}_{C_i}$ velocità ed accelerazioni lineari del baricentro dell'i-esimo link;
- $\boldsymbol{\omega}_{m_i}^i, \dot{\boldsymbol{\omega}}_{m_i}^i$ velocità ed accelerazione angolare dell'i-esimo rotore nell'i-esimo sistema di riferimento;
- $\boldsymbol{f}_i^i$ forza esercitata dal braccio $i-1$ sul braccio i nel sistema di riferimento i-esimo;
- $\boldsymbol{\mu}_i^i$ momento esercitato dal braccio $i-1$ sul braccio i nel sistema di riferimento i-esimo;
- τ_i forza generalizzata al giunto i.

Ora si descrive la procedura iterativa adottata. Si inizia (nella parte in alto a sinistra dello schema) imponendo le accelerazioni e velocità del braccio di base, solitamente assumibili pari a:

$$\ddot{\boldsymbol{p}}_0^0 - \boldsymbol{g}_0^0 = [0\ g\ 0]^T \tag{2.16}$$

$$\boldsymbol{\omega}_0^0 = \dot{\boldsymbol{\omega}}_0^0 = \mathbf{0} \tag{2.17}$$

E si impongono i valori delle forze e momenti applicati all'EE. Che nel caso di assenza di contatto con elementi esterni risultano pari a:

$$\boldsymbol{f}_{n+1}^{n+1} = \mathbf{0} \tag{2.18}$$

$$\boldsymbol{\mu}_{n+1}^{n+1} = \mathbf{0} \tag{2.19}$$

A questo punto è possibile calcolare dal giunto 0 fino all'EE velocità ed accelerazioni angolari del link corrente, accelerazioni lineari dell'origine del sistema di riferimento del link corrente, accelerazione lineare del baricentro link corrente ed accelerazione angolare del rotore. Questo viene eseguito mediante le seguenti relazioni che valgono per giunti rotoidali:

$$\tau_i = \boldsymbol{\mu}_i^T \boldsymbol{z}_{i-1} + k_{ri} I_{mi}\, \dot{\boldsymbol{\omega}}_{m_\iota}^T \boldsymbol{z}_{mi} \tag{2.20}$$

Dove: I_{mi} è il momento d'inerzia del rotore;

k_{ri} è il rapporto di trasmissione meccanica del rotore;

$\boldsymbol{z}_{i-1}$ è versore dell'asse z della terna $i-1$;

$\boldsymbol{z}_{mi}$ è il versore dell'asse del rotore i-esimo.

$$\boldsymbol{\omega}_i^i = \boldsymbol{R}_i^{i-1^T}(\boldsymbol{\omega}_{i-1}^{i-1} + \dot{\theta}_\iota \boldsymbol{z}_0) \tag{2.21}$$

$$\dot{\boldsymbol{\omega}}_i^i = \boldsymbol{R}_i^{i-1^T}(\dot{\boldsymbol{\omega}}_{i-1}^{i-1} + \ddot{\theta}_\iota \boldsymbol{z}_0 + \dot{\theta}_\iota \boldsymbol{\omega}_{i-1}^{i-1} \text{ x } \boldsymbol{z}_0) \tag{2.22}$$

$$\ddot{\boldsymbol{p}}_\iota^\iota = \boldsymbol{R}_i^{i-1^T} \ddot{\boldsymbol{p}}_{\iota-1}^{i-1} + \dot{\boldsymbol{\omega}}_i^i \text{ x } \boldsymbol{r}_{i-1,i}^i + \boldsymbol{\omega}_i^i \text{ x } (\boldsymbol{\omega}_i^i \text{ x } \boldsymbol{r}_{i-1,i}^i) \tag{2.23}$$

Dove: $\boldsymbol{r}_{i-1,i}^i$ il vettore posizione della terna $i-1$ rispetto all'origine della terna i nel sistema di riferimento i.

$$\ddot{\boldsymbol{p}}_{C_\iota}^\iota = \ddot{\boldsymbol{p}}_\iota^\iota + \dot{\boldsymbol{\omega}}_i^i \text{ x } \boldsymbol{r}_{i,C_i}^i + \boldsymbol{\omega}_i^i \text{ x } (\boldsymbol{\omega}_i^i \text{ x } \boldsymbol{r}_{i,C_i}^i) \tag{2.24}$$

$$\dot{\boldsymbol{\omega}}_{m_\iota}^{i-1} = \boldsymbol{\omega}_{i-1}^{i-1} + k_{ri}\, \ddot{q}_\iota \boldsymbol{z}_{mi}^{i-1} + k_{ri}\, \dot{q}_\iota \boldsymbol{\omega}_{i-1}^{i-1} \text{ x } \boldsymbol{z}_{mi}^{i-1} \tag{2.25}$$

Utilizzando tali relazioni si prosegue dal link 1 fino all'ultimo link in modo iterativo. In seguito, conoscendo tutti i parametri cinematici dell'ultimo link è possibile adottare le seguenti relazioni per calcolare, note le forze e coppie applicate all'EE, le forze e coppie applicate all'n-esimo link.

$$\dot{\boldsymbol{f}}_\iota^i = \boldsymbol{R}_{i+1}^{i\ T} \dot{\boldsymbol{f}}_{\iota+1}^{\iota+1} + m_i \ddot{\boldsymbol{p}}_{C_\iota}^i \tag{2.26}$$

$$\boldsymbol{\mu}_i^i = -\boldsymbol{f}_i^i \text{ x } \left(\boldsymbol{r}_{i-1,i}^i + \boldsymbol{r}_{i,C_i}^i\right) + \boldsymbol{R}_{i+1}^i \boldsymbol{\mu}_{i+1}^{i+1} + \boldsymbol{R}_{i+1}^i \boldsymbol{f}_{i+1}^{i+1} \text{ x } \boldsymbol{r}_{i,C_i}^i + \boldsymbol{I}_i^i \dot{\boldsymbol{\omega}}_i^i + \boldsymbol{\omega}_i^i \text{ x } \left(\boldsymbol{I}_i^i \boldsymbol{\omega}_i^i\right) + k_{r,i+1} \ddot{q_{i+1}} I_{m_{i+1}} \boldsymbol{z}_{m_{i+1}}^i + k_{r,i+1} \dot{q_{i+1}} I_{m_{i+1}} \dot{\boldsymbol{\omega}}_i^i \text{ x } \boldsymbol{z}_{m_{i+1}}^i \tag{2.27}$$

Dove: $\boldsymbol{I}_i^i$ è il tensore d'inerzia di link più motore i-esimo.

Con queste relazioni è possibile ricavare il vettore delle forze e dei momenti applicati a ciascun link proseguendo in senso inverso rispetto a prima, ovvero partendo dall'ultimo link e ritornando verso la base del manipolare. Inoltre, ad ogni iterazione è possibile estrapolare il vettore di forze generiche mediante la relazione:

$$\tau_i = \boldsymbol{\mu}_i^{i^T} \boldsymbol{R}_i^{i-1^T} \boldsymbol{z}_0 + k_{ri} I_{mi} \, \boldsymbol{\omega}^{\dot{\iota-1}^T}_{m_\iota} \boldsymbol{z}_{mi}^{i-1} \tag{2.28}$$

3 CAPITOLO 3: ALGORITMI DI COLLISION AVOIDANCE

Tra i molteplici utilizzi della ridondanza cinematica dei manipolatori robotici dopo una prima valutazione generale si è deciso di focalizzare l'attenzione sull'utilizzo per **anticollisione** al fine di sfruttare questa particolarità cinematica nell'ambito della robotica collaborativa, macro-argomento nel quale si vuole integrare questo lavoro di tesi.

A questo scopo, prima di poter realizzare ex novo o modificare alcuni algoritmi di collision avoidance già esistenti è opportuno implementare alcuni interessanti approcci ritrovati a livello bibliografico. L'implementazione di due diversi algoritmi di anticollisione ha permesso il passaggio da conoscenze puramente teoriche ricavate dai più noti testi di robotica come (1) a competenze di tipo informatico per il controllo software di un particolare dispositivo robotico. In ambito di programmazione si utilizza il software Matlab. Si riportano, di seguito, due importanti algoritmi di anticollisione analizzati a fondo e implementati a partire dalla loro base teorica presentata in pubblicazioni scientifiche.

3.1 Depth space approach

Un primo utile algoritmo di collision avoidance è quello proposto da un gruppo di ricerca dell'Università sapienza di Roma cui fanno parte: De Luca A., Flacco F., Kröger T. e Khatib O. (34). In un articolo scientifico questi ricercatori propongono l'integrazione di un metodo di collision avoidance basato su **potenziali** attrattivi e repulsivi applicati all'EE. In modo semplificato l'EE del manipolatore è attratto nella posizione prevista in fase di pianificazione della traiettoria ma a questo campo di attrazione si somma un potenziale repulsivo che lo allontana da eventuali ostacoli presenti nel workspace. Inoltre, per permettere l'identificazione della presenza e posizione di ostacoli nel campo di lavoro del robot è opportuno integrare alla fase di controllo della traiettoria un dispositivo di visione e localizzazione degli ostacoli. A tale scopo è possibile sviluppare l'algoritmo di controllo del manipolatore sulla base di un sensore di profondità spaziale prodotto da Microsoft chiamato **Kinect**. Sulla base di queste osservazioni è possibile sviluppare un algoritmo di collision avoidance che operi in tempo reale.

3.1.1 Cosa è il Depth Space

Il **depth space** è uno spazio a due dimensioni e mezzo non omogeneo in cui due elementi rappresentano le coordinate della proiezione di un punto cartesiano su un piano e il terzo rappresenta la distanza tra il punto e il piano. La valutazione del depth space avviene attraverso il device Microsoft Kinect che contiene al suo interno un depth sensor oltre ad una normale fotocamera. Il depth sensor funziona basandosi su due set di parametri: i parametri intrinseci contenuti in una matrice $\mathcal{K}$ che racchiude le informazioni della proiezione del punto cartesiano sul piano dell'immagine e i parametri estrinseci contenuti nella matrice $\boldsymbol{\varepsilon}$ che rappresentano la trasformazione di coordinate tra il sistema di riferimento considerato da Kinect e il sistema di riferimento del sensore.

$$\boldsymbol{K} = \begin{bmatrix} f\,s_x & 0 & c_x \\ 0 & f\,s_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.29}$$

$$\boldsymbol{\varepsilon} = [\,\boldsymbol{R} \mid \boldsymbol{t}\,] \tag{3.30}$$

Dove: f è la lunghezza focale della fotocamera;

s_x e s_y sono le dimensioni del pixel in metri;

c_x e c_y sono le coordinate del pixel centrale nel piano dell'immagine;

$\boldsymbol{R}$ e $\boldsymbol{t}$ sono indicative della rotazione e traslazione tra i due sistemi di riferimento.

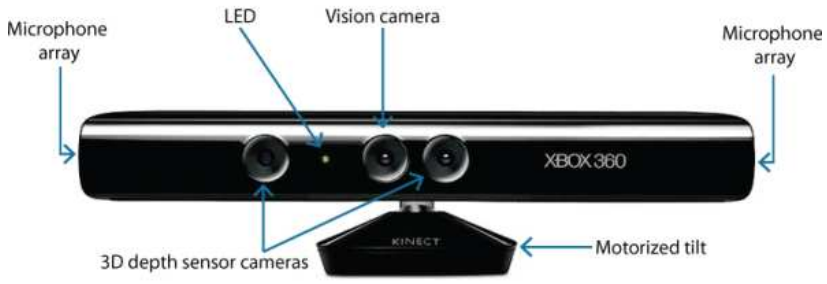


Figura 22 Componenti di Microsoft Kinect

Si noti che solo il punto più vicino lungo il raggio è conservato, tutti i punti la cui visione è occlusa perché si trovano oltre costituiscono una **zona grigia** non percepibile con un solo sensore di profondità. Per tale motivo in alcune applicazioni si è pensato di combinare le informazioni acquisite da due sensori Microsoft Kinect per avere una migliore ricostruzione dell'ambiente tridimensionale nel quale far operare il manipolatore.

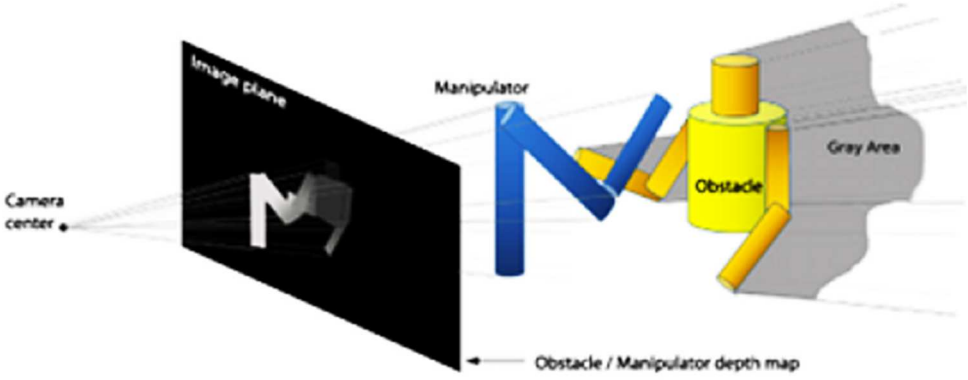


Figura 23 Dimostrazione schematica della zona grigia

Attraverso il depth sensor risulta quindi possibile la valutazione della distanza tra alcuni punti caratteristici del robot e quanto misurato dal Microsoft Kinect come ambiente in cui il manipolatore si muove nell'esecuzione del suo task. Pertanto, è possibile pianificare la traiettoria del manipolatore servendosi di azioni repulsive calcolate in base a queste distanze tra robot ed ostacoli presenti nel suo workspace ed in particolare operatori umani in movimento.

3.1.2 Calcolo delle azioni repulsive per implementare l'anticollisione

Indicando con **P** un punto opportunamente scelto sul robot e con **O** un ostacolo nel workspace è possibile valutare l'**azione repulsiva** mediante un vettore che si presenti nella forma:

$$\boldsymbol{V}_C(\boldsymbol{P},\boldsymbol{O}) = v(\boldsymbol{P},\boldsymbol{O})\frac{\boldsymbol{D}(\boldsymbol{P},\boldsymbol{O})}{\|\boldsymbol{D}(\boldsymbol{P},\boldsymbol{O})\|} \tag{3.31}$$

Ovvero l'azione repulsiva agisce lungo la direzione della congiungente tra il punto **P** sul robot ed il punto **O** dell'ostacolo secondo un modulo indicato da $v(\boldsymbol{P},\boldsymbol{O})$ che si può esplicitare come:

$$v(\boldsymbol{P},\boldsymbol{O}) = \frac{V_{max}}{1 + e^{(\|\boldsymbol{D}(\boldsymbol{P},\boldsymbol{O})\|(2\backslash\rho)-1)\alpha}} \tag{3.32}$$

Dove: V_{max} è il massimo modulo ammissibile dell'effetto repulsivo;

ρ e α sono i due parametri caratteristici che esprimono l'andamento del vettore repulsione in funzione della distanza.

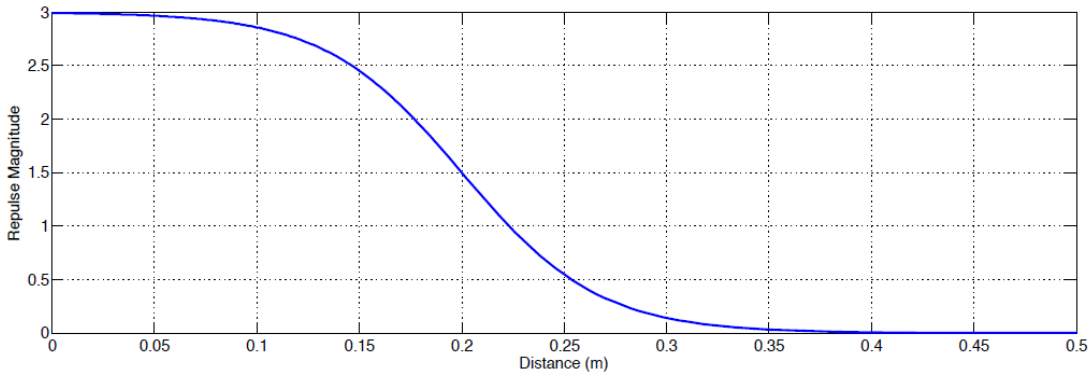


Figura 24 Modulo del vettore repulsivo in funzione della distanza dall'ostacolo ($V_{max} = 3\frac{m}{s}, \rho = 0.4, \alpha = 6$)

Si noti che questo algoritmo di collision avoidance si fonda sulla **distanza** tra manipolatore e ostacolo e non su altri parametri che potrebbero risultare anche molto influenti come la velocità relativa tra i due oggetti, l'inerzia della posa del manipolare ed altri aspetti di cruciale importanza.

Inoltre, sebbene nell'articolo esaminato sia esplicitato in modo molto preciso come sia possibile intervenire a livello di controllo sul manipolatore mediante effetti repulsivi non è altrettanto evidente come scegliere i punti del manipolatore e degli ostacoli tra i quali valutare la distanza ed intervenire. Questo aspetto risulta di cruciale importanza, in quanto, il depth sensor restituisce una nuvola di punti che rappresenta l'intero ambiente ripreso dal sensore e non è possibile pensare di calcolare la distanza tra ogni punto della depth map restituita dal sensore e una serie estesa di punti presi sul manipolatore. Questi ed altri aspetti sono stati oggetto di indagine approfondita e saranno descritti opportunamente nel capitolo seguente.

3.1.3 Miglioramenti dell'algoritmo di anticollisione

Un problema di questo approccio è che, come è stata definita l'azione repulsiva agente sul manipolatore agisce lungo la medesima **direzione** di avvicinamento dell'ostacolo ma in verso

opposto. Questo, inevitabilmente comporta che, se l'ostacolo è in avvicinamento con una velocità superiore rispetto a quella a cui si può muovere il manipolatore, inevitabilmente, a fronte di un'azione repulsiva così realizzata l'urto non può essere evitato. Pertanto, come agiscono anche le persone, una migliore strategia di anticollisione è quella di applicare il potenziale repulsivo in direzione circa normale rispetto alla velocità di avvicinamento dell'ostacolo. Il problema di questo approccio è che valutare la velocità dell'ostacolo non è semplice quando l'ostacolo non è puntiforme o comunque non ha una geometria semplice. Ciononostante, esiste un modo più immediato per il calcolo di questa direzione di applicazione del potenziale repulsivo. Infatti, si può estrapolare l'informazione sulla velocità osservando la variazione temporale del vettore repulsivo. Riportando l'azione repulsiva dal sistema di riferimento della fotocamera al sistema di riferimento assoluto con la relazione:

$$\boldsymbol{V}_R(P) = \boldsymbol{R}^T \boldsymbol{V}_C(P) \tag{3.33}$$

Si può considerare l'ostacolo rispetto al quale si vuole applicare un vettore repulsivo come un fulcro per l'effetto repulsivo stesso e pertanto calcolando i seguenti parametri:

$$\boldsymbol{a} = \frac{\dot{\boldsymbol{V}}_R(\boldsymbol{P})}{\left\|\dot{\boldsymbol{V}}_R(\boldsymbol{P})\right\|}, \qquad \boldsymbol{r} = \frac{\boldsymbol{V}_R(\boldsymbol{P})}{\|\boldsymbol{V}_R(\boldsymbol{P})\|}, \qquad \beta = arcos(\boldsymbol{a}^T \boldsymbol{r}) \tag{3.34}$$

è possibile calcolare il vettore repulsione corretto in direzione con il seguente algoritmo:

$$\begin{aligned}
&\boldsymbol{if}\ \beta < \frac{\pi}{2} \\
&\quad \boldsymbol{n} = \boldsymbol{a} \times \boldsymbol{r}, \qquad \boldsymbol{v} = \frac{\boldsymbol{n} \times \boldsymbol{a}}{\|\boldsymbol{n} \times \boldsymbol{a}\|} \\
&\quad \gamma = \beta + \frac{\beta - \frac{\pi}{2}}{1 - e^{-(\|\dot{\boldsymbol{V}}_R(\boldsymbol{P})\|(2/\dot{\boldsymbol{V}}_{R_max})-1)c}} \\
&\quad \boldsymbol{V}_{R_{pivot}}(\boldsymbol{P}) = \|\boldsymbol{V}_R(\boldsymbol{P})\|(\cos\gamma\ \boldsymbol{a} + \sin\gamma\ \boldsymbol{v}) \\
&\boldsymbol{else} \\
&\quad \boldsymbol{V}_{R_{pivot}}(\boldsymbol{P}) = \boldsymbol{V}_R(\boldsymbol{P}) \\
&\boldsymbol{end}
\end{aligned} \tag{3.35}$$

Sul robot non potendo prendere tutti i punti della reale forma geometrica è possibile usare una **discretizzazione del manipolatore** in una serie di punti ritenuti maggiormente significativi la cui posizione nel tempo si può ricostruire con l'ausilio di un modello cinematico del manipolatore. Per essere certi che non si verifichi un urto contro il manipolatore un semplice espediente consiste nel considerare i punti caratteristici del manipolatore come centro di sfere il cui raggio può essere sottratto in fase di calcolo delle distanze tra uomo e robot, in tal modo, scegliendo sfere sufficientemente grandi da circoscrivere l'intero corpo del manipolatore si può applicare un'azione repulsiva che impedisca il contatto con la superficie esterna dello stesso manipolatore.

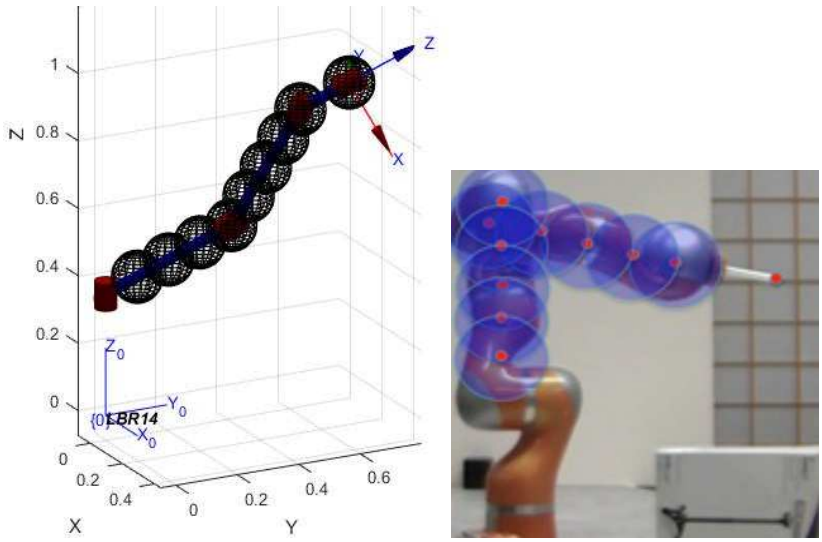


Figura 25 Rappresentazione della discretizzazione del manipolatore

3.2 Cumulative Danger Field Rocco

Un altro efficace algoritmo di collision avoidance è stato, invece, proposto da un gruppo di ricercatori del Politecnico di Milano di cui fanno parte: Rocco P., Lacevic B., Zanchettin A. M. Dal 2010 questi ricercatori hanno proposto differenti approcci per il controllo di manipolatori robotici finalizzato all'anticollisione e quindi ad una collaborazione sicura con l'uomo. Il principio su cui si basa l'algoritmo di controllo è sempre sull'implementazione di potenziali repulsivi dagli ostacoli nel workspace ma varia la filosofia con la quale sono definiti i moduli dei vettori repulsivi (59).

3.2.1 Kinetostatic Danger Field

Un importante contributo proposto da Paolo Rocco ed il suo gruppo di ricercatori in questo campo dell'anticollisione risale al 2010 con l'articolo citato in precedenza (59). L'obiettivo è quello di gettare le fondamenta per un'applicazione real time di un algoritmo di anticollisione di un dispositivo robotico ridondante. Al contrario della trattazione proposta dai ricercatori dell'Università Sapienza di Roma in cui l'algoritmo di anticollisione si basa esclusivamente sulla distanza tra ostacolo e manipolatore l'aspetto innovativo è quello di basare il calcolo di questo vettore repulsivo sulla base di un campo di pericolo definito dagli autori **Kinetostatic Danger Field** (KSDF). La particolarità è l'inserimento, oltre alla distanza oggetto-robot, della velocità relativa tra i due corpi come parametro indicativo del livello di pericolo nell'assunzione di particolari pose assunte dal robot stesso.

Secondo queste osservazioni è possibile esplicitare il KSDF generato dalla porzione elementare di uno dei link del robot mediante la seguente relazione:

$$DF(\boldsymbol{r}, \boldsymbol{r_t}, \boldsymbol{v_t}) = \frac{k_1}{\|\boldsymbol{r} - \boldsymbol{r_t}\|} + \frac{k_2 \|\boldsymbol{v_t}\| [\gamma + cos\angle(\boldsymbol{r} - \boldsymbol{r_t}, \boldsymbol{v_t})]}{\|\boldsymbol{r} - \boldsymbol{r_t}\|^2} \tag{3.36}$$

Dove: k_1, k_2 e $\gamma \geq 1$ sono costanti positive;

$\boldsymbol{r}$ è il vettore che individua il punto nello spazio in corrispondenza del quale si sta valutando il Danger Field;

$\boldsymbol{r_t}$ e $\boldsymbol{v_t}$ sono la posizione e la velocità dell'elemento in movimento.

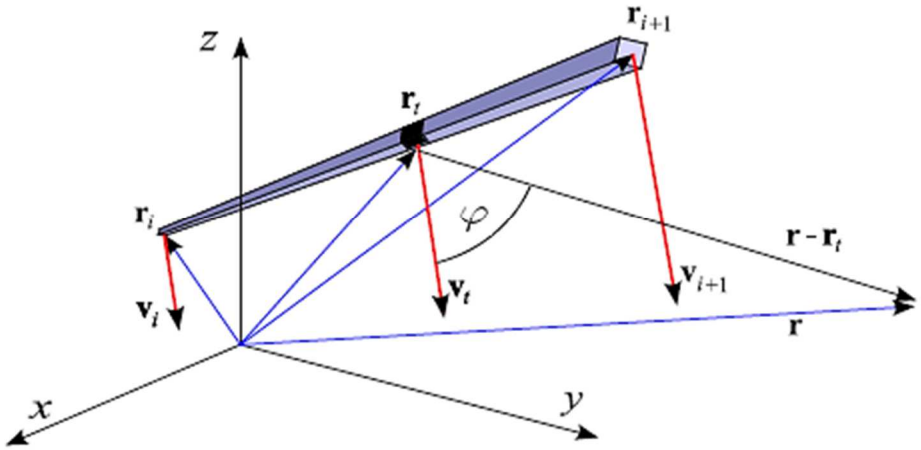


Figura 26 Individuazione elementi per il calcolo del Danger Field

Partendo dal Danger Field così calcolato è possibile valutare il **Cumulative Danger Field** (CDF) che esprime il campo di pericolo valutato considerando l'intero link e non solo un elemento dello stesso integrando per tutta l'estensione del link ed eventualmente sommando il contributo di tutti i link costituenti il manipolatore.

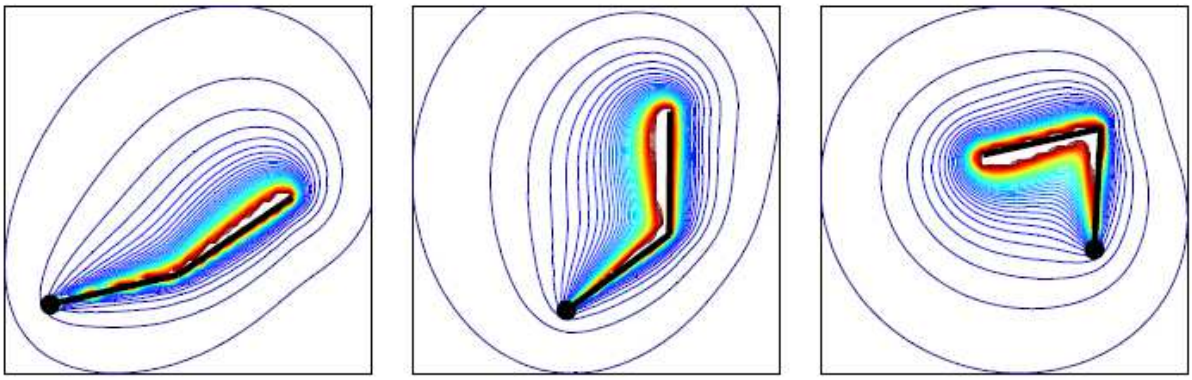


Figura 27 Visualizzazione del CDF intorno ad un manipolatore planare a 2 DOF

3.2.2 Approccio Cinematico basato sul CDF

L'anno successivo alla sua introduzione, gli autori del CDF, hanno proposto un suo utilizzo a livello puramente cinematico per il controllo di un manipolatore ridondante (60). Innanzitutto, è possibile osservare che il CDF è una quantità scalare valutata in uno specifico punto dello spazio ma può essere trasformata in una quantità vettoriale tenendo conto del suo gradiente spaziale mediante la semplice relazione:

$$\boldsymbol{CDF}(\boldsymbol{r}) = CDF(\boldsymbol{r})\frac{\nabla CDF(\boldsymbol{r})}{\|\nabla CDF(\boldsymbol{r})\|} \tag{3.37}$$

Per applicare questo CDF nel controllo di un manipolatore ridondante occorre ricorrere all'algoritmo di inversione cinematica CLIK (Closed-Loop Inverse Kinematic).

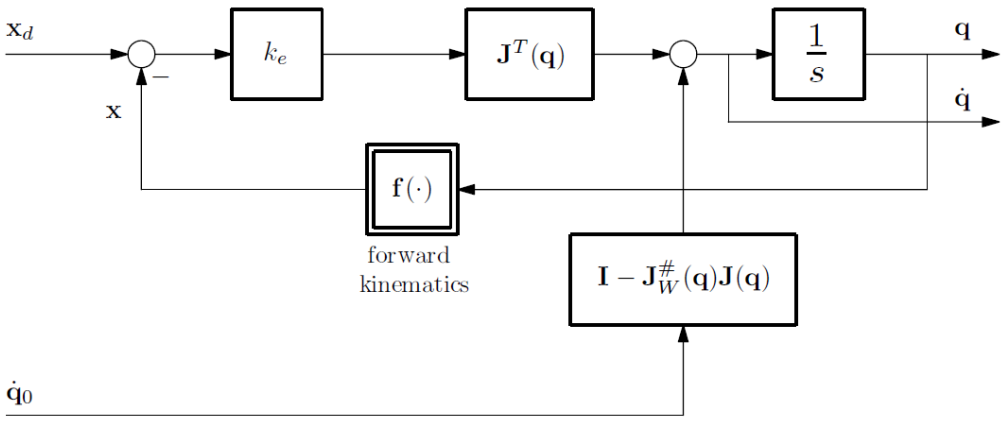


Figura 28 Diagramma di flusso algoritmo CLIK

L'espressione che consente il calcolo delle velocità di giunto è:

$$\dot{\boldsymbol{q}} = k_e \boldsymbol{J}^T(\boldsymbol{q})\,(\boldsymbol{x}_d - \boldsymbol{x}) + \left[\boldsymbol{I} - \boldsymbol{J}_W^{\#}(\boldsymbol{q})\,\boldsymbol{J}(\boldsymbol{q})\right]\dot{\boldsymbol{q}}_0 \tag{3.38}$$

Dove: k_e è un parametro reale e positivo;

$\boldsymbol{J}_W^{\#}$ è la matrice Jacobiana pesata generalizzata.

Indicando il vettore di velocità di giunto appartenente allo spazio nullo con:

$$\dot{\boldsymbol{q}}_0 = k_v\ {\boldsymbol{J}_{S,v}}^T(\boldsymbol{q})\,\boldsymbol{CDF}(\boldsymbol{r}) \tag{3.39}$$

Dove: k_v è una costante reale positiva;

${\boldsymbol{J}_{S,v}}^T(\boldsymbol{q})$ indica le prime tre colonne della matrice Jacobiana ${\boldsymbol{J}_S}^T(\boldsymbol{q})$ associata al punto S;

$\boldsymbol{CDF}(\boldsymbol{r})$ è il vettore caratteristico del CDF calcolato secondo la relazione precedente;

Un'ulteriore perfezionamento dell'algoritmo è possibile prevedendo che il **task** pianificato sia **rilassato** nel caso in cui il CDF superi una certa soglia garantendo una maggiore sicurezza nel funzionamento del manipolatore. Questa operazione è semplicemente prevedibile valutando le velocità di giunto attraverso l'equazione seguente.

$$\dot{\boldsymbol{q}} = mk_e \boldsymbol{J}^T(\boldsymbol{q})\,(\boldsymbol{x}_d - \boldsymbol{x}) + \left[\boldsymbol{I} - m\boldsymbol{J}_W^{\#}(\boldsymbol{q})\,\boldsymbol{J}(\boldsymbol{q})\right]\dot{\boldsymbol{q}}_0 \tag{3.40}$$

Nel caso più semplice è possibile porre il parametro m pari a zero se il CDF è sopra la soglia prefissata, pari ad uno altrimenti. È evidente che quando $m = 0$ le velocità di giunto imposte al manipolatore sono uguali a quelle valutate nello spazio nullo per evitare l'ostacolo in questione.

Ricapitolando, questo algoritmo di anticollisione presenta il vantaggio, rispetto al precedente, di valutare non solo la **distanza** tra robot e ostacoli ma anche la **velocità relativa** tra i due oggetti, il cui quadrato è ovviamente proporzionale all'energia cinetica e quindi collegabile alla quantità di energia che si potrebbe sviluppare in fase di urto. Ciononostante, questo algoritmo come il precedente richiede la definizione dei punti nello spazio nei quali valutare il CDF e non fa riferimento a nessun metodo per valutarli. Anche se si scegliesse di adottare questo algoritmo

di anticollisione con dei sensori di profondità come il Microsoft Kinect rimane il quesito presentato in precedenza di come processare la mole di dati fornita dallo stesso e rendere l'algoritmo non solo efficace ma anche efficiente per un'implementazione real time che possa garantire la sicurezza degli operatori in ambito di robotica collaborativa.

3.2.3 Kinetostatic Safety Field

Anche negli ultimi mesi, lo sviluppo di algoritmi per il controllo sicuro di dispositivi robotici è in continua evoluzione. Un altro approccio, oltre quello del Danger Field risulta essere quello del **Kinetostatic Safety Field** (61). Questo approccio, come il precedente, valuta la posizione e velocità relativa tra il robot in movimento ed i punti dello spazio potenzialmente occupati da un operatore o da un altro robot. Prerogativa di tale approccio, che lo distingue dal Danger Field, è che considera la reale **forma** del dispositivo **sorgente di pericolo**, garantendo comunque una implementazione real time del controllo. Il passaggio da una geometria semplificata, come può essere quella di una successione di link unidimensionali, alla reale geometria del robot incrementa non solo l'accuratezza dell'algoritmo di anticollisione, ma anche, la complessità computazionale dello stesso e la conseguente difficoltà di implementazione.

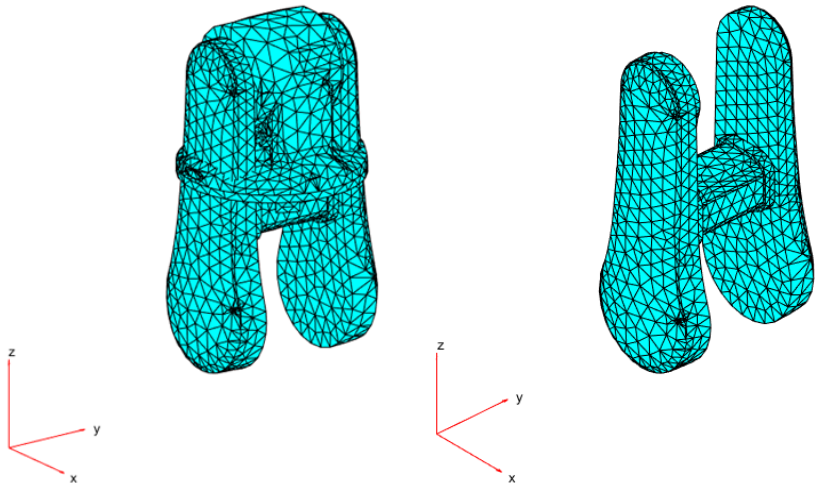


Figura 29 Esempio di implementazione mediante Matlab della reale geometria di un manipolatore

Il generico link di un dispositivo robotico può essere approssimato ad un corpo rigido e studiato con la cinematica dei corpi rigidi in $\mathbb{R}^3$. Fissando un sistema di riferimento locale l e considerando un generico punto T, esso può essere individuato, nel sistema di riferimento locale, mediante un vettore posizione $\boldsymbol{r_t} = (x_t,\ y_t, z_t)^T$. D'altro canto, un generico punto nello spazio può essere individuato, rispetto al sistema di riferimento locale, attraverso il vettore posizione $\boldsymbol{r} = (x,\ y,\ z)^T$ ed il vettore velocità $\boldsymbol{v} = \left(v_x,\ v_y, v_z\right)^T$.

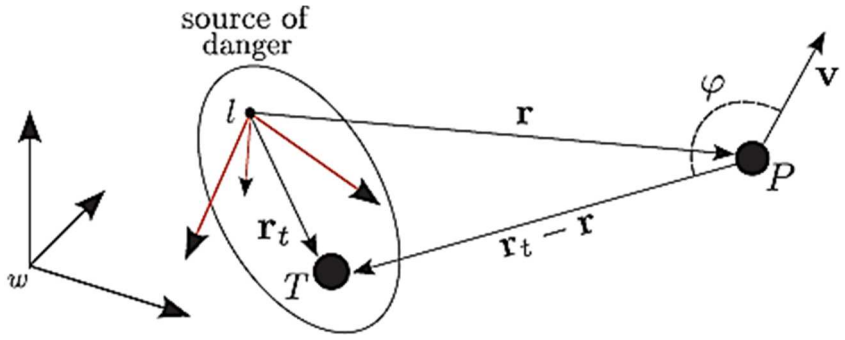


Figura 30 Sistema di riferimento world, sistema di riferimento locale e vettori posizione e velocità caratteristici per il calcolo del KSSF

Con questi elementi è possibile definire il Kinetostatic Safety Field elementare:

$$SF(\boldsymbol{r_t}, \boldsymbol{r}, \boldsymbol{v}) = \|\boldsymbol{r_t} - \boldsymbol{r}\|^2(\gamma - (\boldsymbol{r_t} - \boldsymbol{r})^T\boldsymbol{v}) \tag{3.41}$$

Dove: γ è una costante reale positiva tale che $SF \in \mathbb{R}^+$.

Prima di implementare però il Kinetostatic Safety Field, valutando la reale geometria della sorgente di pericolo però, è necessario discretizzare il corpo rigido, comunque complesso mediante una mesh triangolare. Questo espediente, permette di semplificare il problema e maneggiarlo numericamente con l'ausilio del calcolatore.

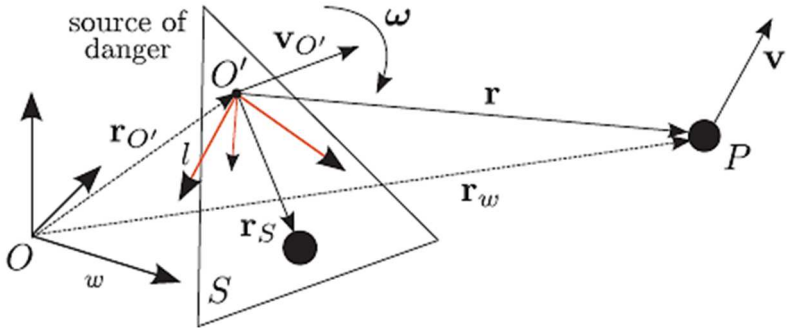


Figura 31Sistema di riferimento world, sistema di riferimento locale e vettori posizione e velocità caratteristici per il calcolo del KSSF di un triangolo

Per estensione della definizione, precedentemente riportata del KSSF è possibile definire il Cumulative Kinetostatic Safety Field (CKSSF) come un integrale di superficie:

$$CSF(\boldsymbol{r}, \boldsymbol{v}) = \frac{1}{A}\int\int_S SF(\boldsymbol{r_t}, \boldsymbol{r}, \boldsymbol{v})\, dS \tag{3.42}$$

Dove: A è l'area della superficie del generico triangolo appartenente alla mesh.

Conseguentemente il CKSSF di un corpo rigido può essere espresso come il valore medio del CKSSF calcolato sugli N triangoli appartenenti alla mesh di discretizzazione:

$$CSF(\boldsymbol{r}, \boldsymbol{v}) = \frac{1}{N}\sum_{i=1}^{N} CSF_i(\boldsymbol{r}, \boldsymbol{v}) \tag{3.43}$$

4 CAPITOLO 4: IMPLEMENTAZIONE ALGORITMI DI COLLISION AVOIDANCE

La comprensione dei fondamenti teorici degli algoritmi di collision avoidance è solo il primo passo per l'implementazione di un algoritmo che sia non solo funzionante ma che risulti anche computazionalmente efficiente per garantire un'applicabilità in real time. A tale scopo è opportuno definire prima di tutto cosa si può intendere come applicazione real time in campo di robotica collaborativa. In questa interazione i protagonisti sono ovviamente due, da un lato, l'uomo, dall'altro il manipolatore robotico. Iniziando dalla controparte più razionale e semplice da descrivere numericamente un robot, come ad esempio l'UR3 della Universal Robot, è capace di **inviare e trasmettere dati** ad una frequenza massima di 125 Hz. Dall'altra parte i **tempi di visione e reazione** di un uomo sono meno semplici da stabilire, è noto che l'uomo non invia al cervello nessuna informazione visiva al di sopra dei 25 Hz di frequenza e che i tempi di reazione difficilmente possono essere al di sotto del decimo di secondo. Quindi, per poter definire una valida applicazione real time in ambito di robotica collaborativa non è semplice definire la frequenza necessaria per il controllo del manipolatore, che comunque per essere superiore ai tempi di reazione di una persona deve essere di alcune decine di Hz.

Si presentano ora gli studi effettuati e l'implementazione realizzata degli algoritmi di anticollisione studiati, aggiungendo le osservazioni ed i miglioramenti apportati per risolvere le problematiche che si sono presentate. In fase di implementazione si è seguito un trand di complessità crescente. In linea generale, l'implementazione degli algoritmi di anticollisione ha seguito il seguente processo evolutivo:

1. Studio della reazione del manipolatore modellato in presenza di uno o più ostacoli puntiformi fissi nello spazio;
2. Studio dell'interazione di un ostacolo puntiforme con traiettoria preimpostata;
3. Studio della reazione ad un ostacolo puntiforme con traiettoria casuale ed arbitraria;
4. Acquisizione di dati relativi al movimento di un braccio ed implementazione in post processing della sua influenza su una traiettoria precedentemente pianificata del robot;
5. Acquisizione dell'intera figura umana ed implementazione in post processing della sua influenza su una traiettoria precedentemente pianificata del robot.

Tutti questi passaggi saranno gradualmente descritti nei paragrafi seguenti. Inoltre, si osservi che tutte le analisi sono condotte in un ambiente di simulazione in cui il robot è modellato come descritto nel capitolo 2 e i sensori sono considerati ideali o con errori e ritardi noti e descritti nei vari casi analizzati.

4.1 Schematizzazione dell'algoritmo implementato

Prima di osservare i vari test e le modifiche eseguite agli algoritmi di anticollisione analizzati se ne analizza una prima versione utilizzata come punto di partenza per evidenziare i **passaggi fondamentali** e le variabili utilizzate.

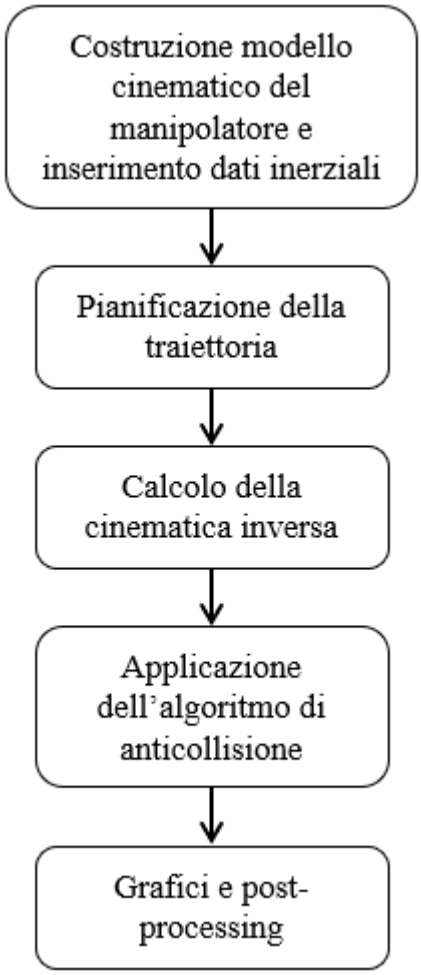


Figura 32 Schema di base controllo manipolatore con algoritmo di anticollisione

La Figura 32 mostra uno schema di concetto utilizzato per l'implementazione di una logica di controllo applicata al manipolatore ed evidenzia le varie fasi che occorre seguire in ordine logico. Ora si analizzino più in dettaglio queste fasi evidenziando quali siano i passaggi chiave e le variabili di maggiore interesse che bisogna utilizzare. Si fa presente che, per proseguire con l'implementazione di tutti gli algoritmi realizzati, si utilizza l'ambiente di programmazione **Matlab** e ci si è serviti di apposite **librerie** e funzioni precedentemente implementate.

4.1.1 Costruzione modello cinematico del manipolatore e inserimento dati inerziali

Il primo passaggio riportato nello schema in Figura 32 è quello di realizzare un modello cinematico del manipolatore che possa permettere di testare in modo verosimile le logiche di controllo proposte in ambiente virtuale. A tale scopo si utilizza la **toolbox** di Matlab realizzata dal noto ricercatore australiano dell'Università di Melbourne **Peter Corke**. Questo è uno degli strumenti più noti ed utilizzati in ambito didattico-sperimentale per la robotica ed ha subito molteplici miglioramenti nel corso degli anni, giungendo, ad oggi, alla versione 9.10. Proprio tale versione è utilizzata per il presente lavoro di tesi.

Si riporta ora nuovamente uno schema di concetto che permetta di comprendere quali siano le variabili in ingresso ed in uscita dal primo blocco riportato in Figura 32 per avere una migliore comprensione del suo effettivo funzionamento. Nel corso del presente lavoro di tesi si è adottato

in precedenza un modello del robot puramente cinematico, poi è stato eseguito l'inserimento dei dati inerziali. In questa sede, si riporta solo la versione più completa ovvero quella di modello di manipolatore con dati cinematici ed inerziali, opportunamente presi dai documenti tecnici.

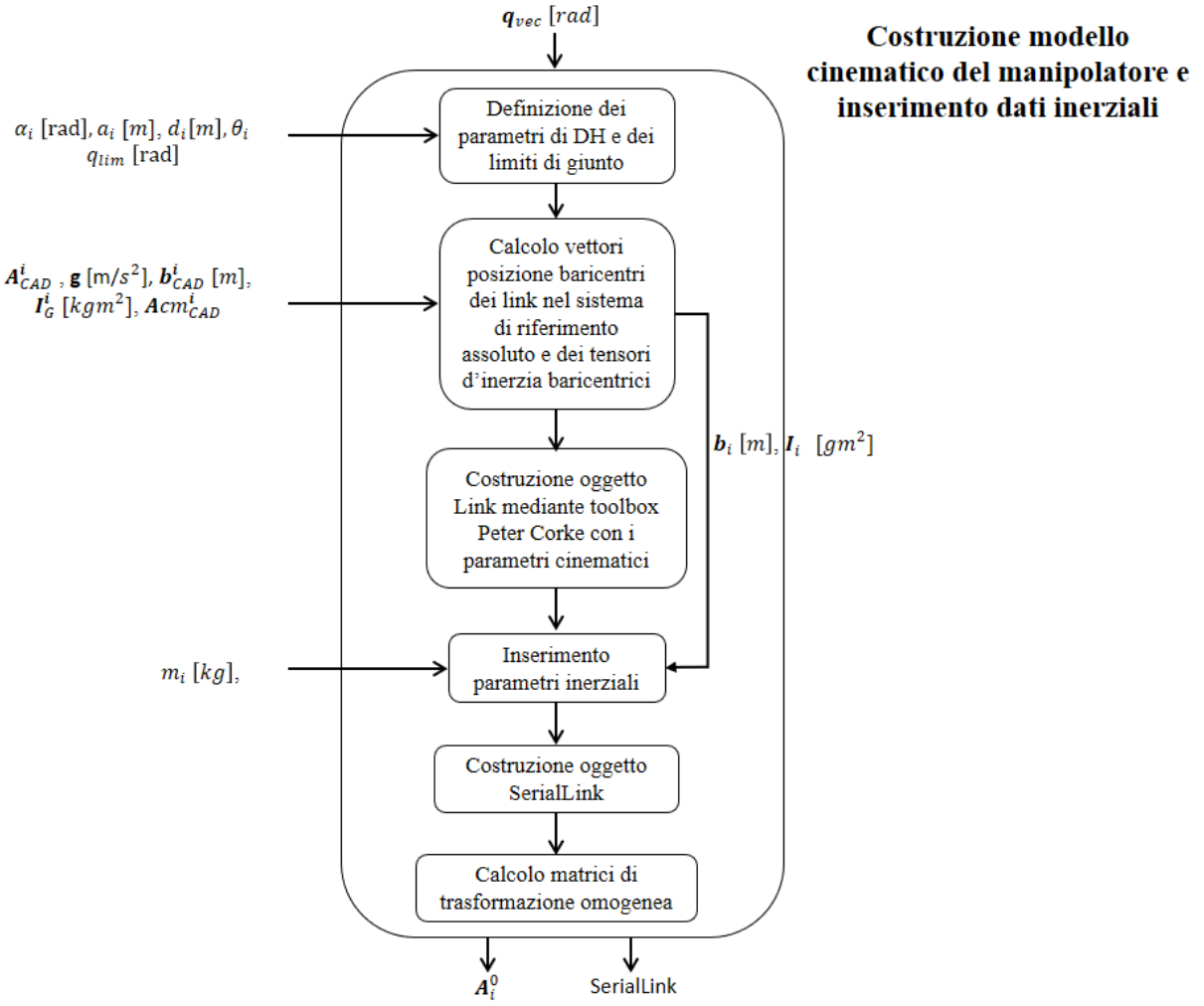


Figura 33 Schema logico costruzione modello cinematico con dati inerziali

Come riportato in Figura 33 nel primo step per la realizzazione di un efficace algoritmo di controllo si parte dal vettore $\boldsymbol{q_{vec}}$ esprimente gli angoli di giunto del manipolatore in una certa posa e si arrivano a definire le matrici di trasformazione omogenea di ogni link e un oggetto SerialLink che raccoglie tutte le informazioni geometriche ed inerziali prelevate dai cataloghi tecnici. Si noti inoltre che, per il calcolo dei vettori posizione dei baricentri dei link nel sistema di riferimento assoluto e dei tensori di inerzia, occorre definire dapprima tali parametri nel sistema di riferimento del CAD dal quale sono stati prelevati, ed in seguito, trasferirli nel sistema di riferimento assoluto utilizzando delle funzioni di trasformazione omogenea dal sistema di riferimento CAD al sistema di riferimento realizzato mediante la convenzione di Denavit-Hartenberg.

4.1.2 Pianificazione della traiettoria

Si analizzi ora il macro blocco di pianificazione della traiettoria. In questo caso occorre distinguere il tipo di traiettoria che si vuole realizzare, sebbene si sono implementate sia il

mantenimento di una posa fissa, sia una traiettoria rettilinea, in questa sede si analizza quest'ultima in quanto più completa.

Come prima, si cerca di esprimere in modo schematico e semplificato la successione delle operazioni da eseguire per pianificare la traiettoria e le variabili utilizzate.

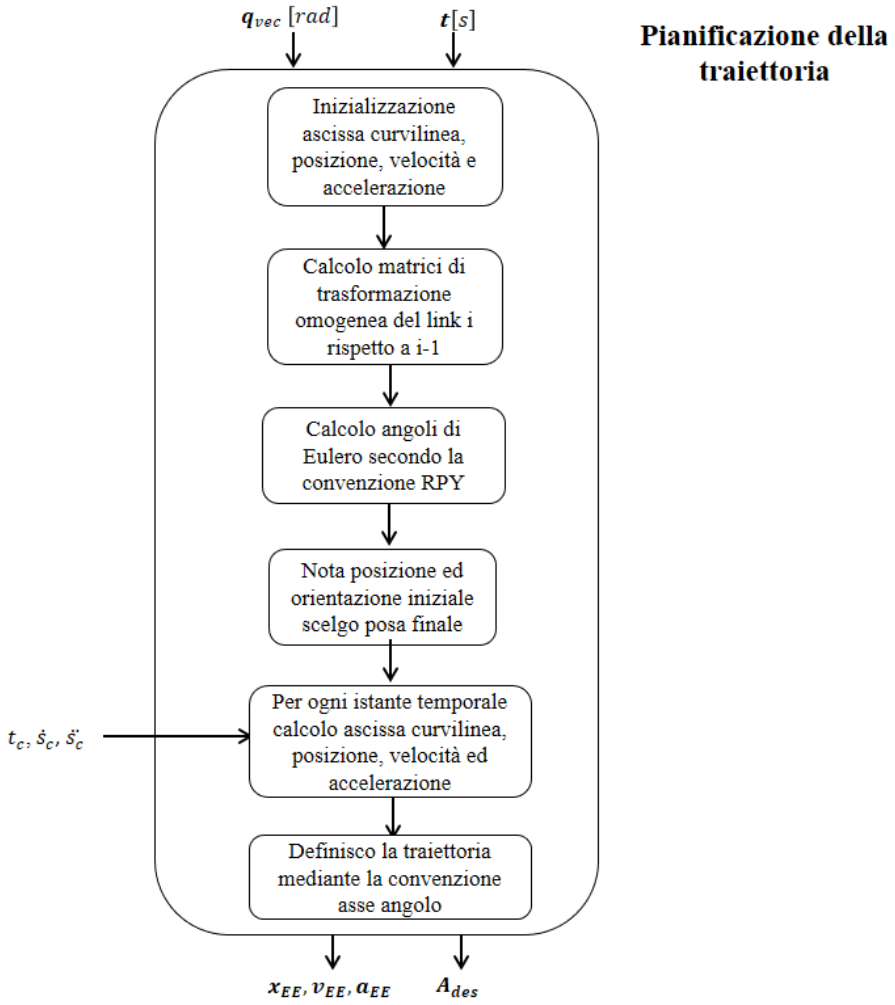


Figura 34 Schema logico pianificazione della traiettoria

In questo secondo step logico è possibile constatare che, per la realizzazione della traiettoria rettilinea, è sufficiente partire da un vettore che esprima la posa iniziale del manipolatore e da un vettore indicante la discretizzazione temporale per uscire con tutte le informazioni necessarie per descrivere la traiettoria del manipolatore mediante la convenzione asse angolo. Si noti che, per procedere ad una corretta pianificazione della traiettoria su un segmento nello spazio operativo, la scelta più comune è definire un trapezio di velocità. A tal proposito, è necessario definire quale sia il valore di velocità da mantenere costante mediante l'ausilio dell'ascissa curvilinea ($\dot{s}_c$), il tempo della fase di accelerazione (t_c) ed il conseguente valore di accelerazione costante da mantenere in fase di accelerazione e decelerazione ($\ddot{s}_c$). Infine, altro dato importante ma arbitrariamente imponibile è la posa finale. Nel caso in esame, per semplicità, si è assunta come posa finale una posa con posizione dell'EE traslata, rispetto alla posizione iniziale, di un certo valore indicante la lunghezza del segmento che si desidera tracciare e orientazione uguale a quella iniziale.

4.1.3 Calcolo della cinematica inversa

Sulla base di quanto riportato in precedenza, in seguito all'operazione di pianificazione della traiettoria, è necessario fare il calcolo della cinematica inversa. Anche in questo caso come per il passaggio di pianificazione della traiettoria possono essere intraprese differenti strade. In particolare, negli script realizzati si adotta l'algoritmo CLIK (Closed Inverse Kinematic) basato sulla pseudo-inversa come descritto nel paragrafo 1.5.2.

Si riporta ora uno schema di concetto per esprimere quanto implementato.

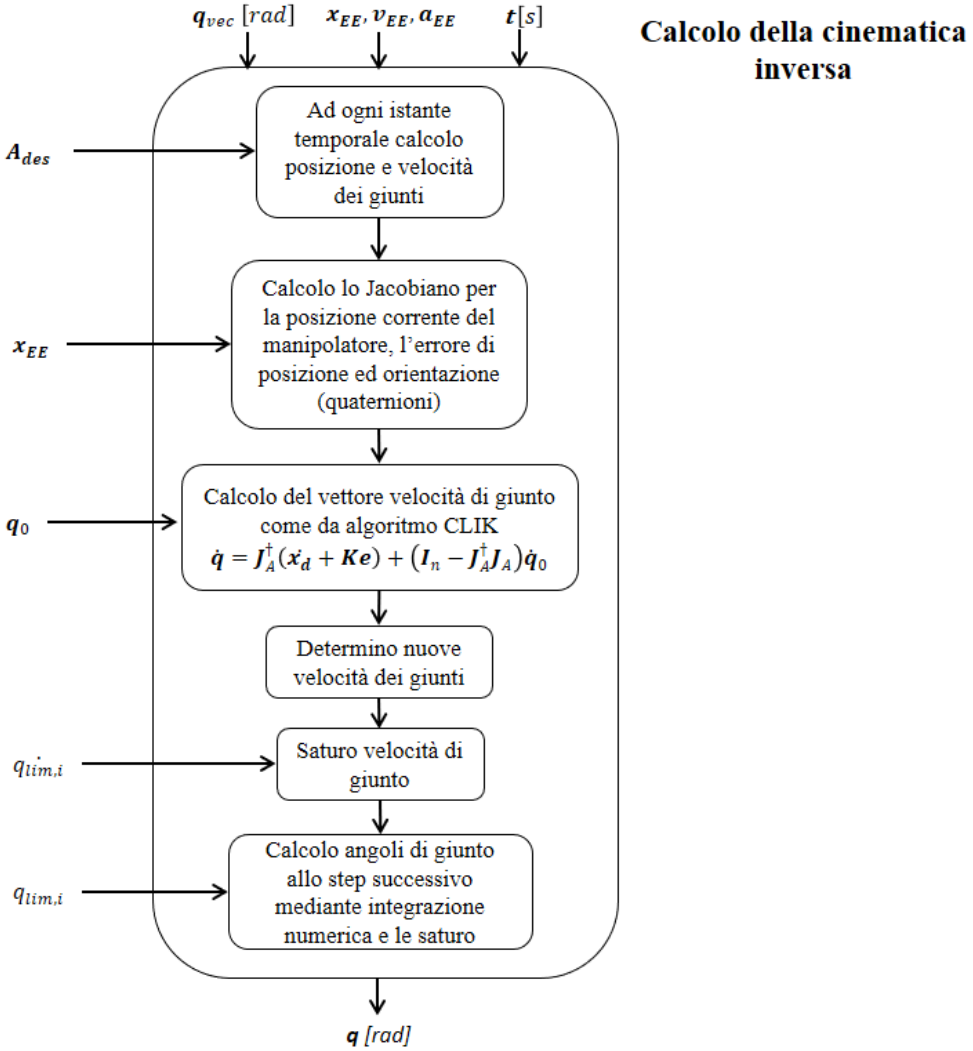


Figura 35 Schema logico cinematica inversa

Come accennato la cinematica inversa nell'algoritmo implementato utilizza la tecnica CLIK con matrice Jacobiana pseudo-inversa. In questa fase, note le informazioni direttamente ricavate dalla pianificazione della traiettoria è possibile ricavare il vettore delle variabili di giunto da imporre come controllo allo step temporale successivo. Ovviamente, per un efficace implementazione occorre inserire varie informazioni secondarie ovvero i limiti di escursione angolare e velocità angolare dei giunti in modo da riprodurre quanto più fedelmente possibile il reale comportamento del robot. In questa fase rientra anche l'output del cuore dell'algoritmo di anticollisione ovvero il vettore delle velocità di giunto appartenenti allo spazio nullo $\dot{q}_0$

calcolato appositamente per sfruttare i gradi di ridondanza del manipolatore inficiando il meno possibile sulla traiettoria pianificata mediante il self-motion.

4.1.4 Applicazione dell'algoritmo di anticollisione

Il modo di implementare l'algoritmo di collision avoidance varia ovviamente in base al tipo di algoritmo che si vuole applicare, che si tratti del metodo dei potenziali di De Luca, del metodo del campo di pericolo oppure di modifiche e personalizzazioni a tali metodi fatti nel corso dello studio della presente tesi. Pertanto, in questa fase si descrive, nel modo più generale possibile, l'implementazione dell'algoritmo di anticollisione al fine di evidenziare il metodo di determinazione del vettore di velocità di giunto appartenenti allo spazio nullo che deve essere retro-azionato al blocco di risoluzione della cinematica inversa.

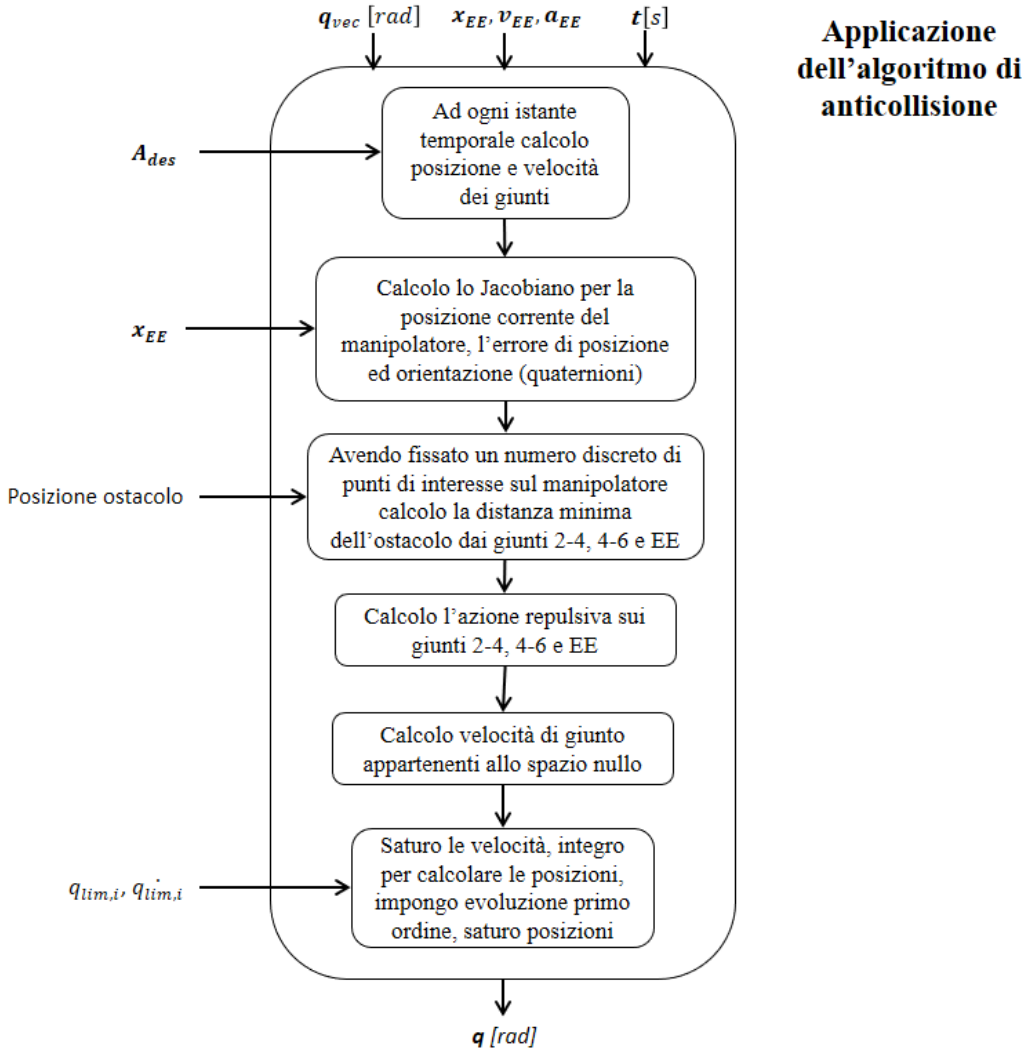


Figura 36 Schema logico applicazione algoritmo anticollisione

Lo schema precedentemente riportato è la versione più semplice, successivamente modificata e resa più completa. Infatti, in questo schema logico è previsto il calcolo delle azioni repulsive basandosi solo sulle distanze. Successivamente, come vedremo in seguito, l'algoritmo ha visto l'implementazione della valutazione delle distanze di più punti sul manipolatore con più punti esterni considerati come ostacoli puntiformi o sferici, poi si sono considerati ostacoli in moto

ed è stata introdotto come parametro per la determinazione delle azioni repulsive anche la velocità relativa rendendo via via più complesso l'algoritmo.

4.1.5 Grafici e post processing

Infine, nell'ultima fase si cercano di rappresentare graficamente i risultati ottenuti per analizzare in modo completo l'efficacia e l'efficienza dell'algoritmo implementato. In tale fase è anche possibile visualizzare graficamente il robot, gli ostacoli esterni ed i loro movimenti relativi nel caso non lo si faccia già in fase di implementazione per ogni istante temporale per non rallentare l'esecuzione dello stesso algoritmo.

4.2 Depth Space Approach

Compatibilmente alle informazioni acquisite attraverso lo studio e la valutazione degli articoli scientifici inerenti al Depth Space Approach si implementa questo algoritmo di anticollisione basato sui potenziali in ambiente di programmazione Matlab. Si parte da questo algoritmo, in quanto, concettualmente più semplice. Infatti, come parametri che influiscono sul calcolo del vettore velocità di giunto nello spazio nullo non ci sono le velocità, bensì, solo le posizioni degli ostacoli esterni.

4.2.1 Ostacolo/ostacoli puntiformi fissi

Come descritto nel paragrafo 3.1.1 avendo a disposizione un manipolatore ridondante, è possibile intervenire sulle velocità di giunto appartenenti allo spazio nullo $\dot{\boldsymbol{q}}_0$. In questo modo, compatibilmente con lo svolgimento della traiettoria precedente pianificata per l'EE, è possibile evitare la collisione con gli ostacoli eventualmente presenti del workspace del manipolatore.

Per un primo semplice utilizzo del metodo dei potenziali è possibile considerare un **oggetto puntiforme** in una **posizione** spaziale **fissa** e nota. A questo punto, si valuta la capacità del manipolatore di eseguire il task prestabilito adattando la traiettoria seguita in funzione di quanto definito dall'algoritmo di anticollisione. Si riporta di seguito una rappresentazione grafica realizzata mediante il toolbox di Matlab di Peter Corke per questa prima prova eseguita:

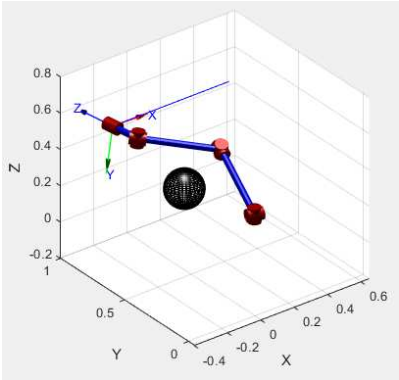


Figura 37 Collision Avoidance con un oggetto puntiforme fermo che intralcia il gomito del manipolatore

È possibile constare, come il manipolatore esegua correttamente la traiettoria rettilinea implementata evitando l'ostacolo mediante rotazione del gomito.

Per testare ulteriormente l'algoritmo di anticollisione si esegue la medesima prova con **due oggetti puntiformi fissi** nello spazio in posizione prestabilita.

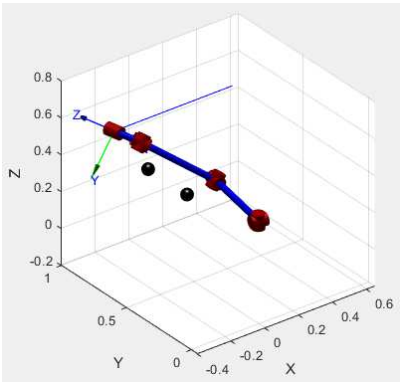


Figura 38 Collision Avoidance con due oggetti puntiformi fissi che intralciano il gomito del manipolatore

Anche in questo secondo caso studio è possibile constare, come il manipolatore esegua correttamente la traiettoria rettilinea implementata evitando l'ostacolo mediante rotazione del gomito.

A questo punto, eseguite queste prove preliminari per verificare che non ci siano problemi o errori di implementazione, si può testare l'algoritmo di anticollisione in condizioni più critiche disponendo uno degli **ostacoli lungo la traiettoria** secondo cui il manipolatore dovrebbe muoversi. In tal modo, è possibile constatare come il manipolatore modifica la propria traiettoria, allontanandosi dalla traiettoria per evitare l'imminente urto.

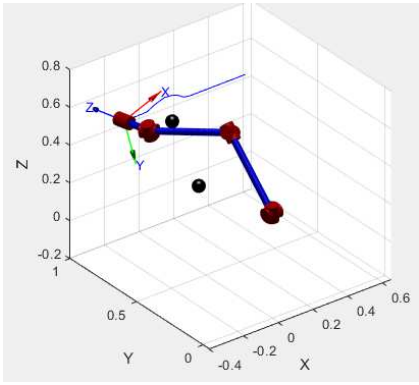


Figura 39 Collision Avoidance con due oggetti puntiformi fissi, uno intralcia il gomito del manipolatore l'altro è disposto lungo la traiettoria rettilinea pianificata per il movimento del manipolatore stesso

Come è possibile constatare dalla rappresentazione in Figura 39, la posizione occupata dall' ostacolo costringe il manipolatore ad allontanarsi dalla traiettoria rettilinea implementata per non colpire l'ostacolo. Compatibilmente con la presenza dell'ostacolo però, quando il pericolo di collisione viene meno, il manipolatore torna sulla traiettoria implementata. Questo fenomeno è dovuto alla prevalenza del potenziale repulsivo di allontanamento dell'EE dall'ostacolo rispetto al potenziale attrattivo del medesimo EE rispetto al target da raggiungere.

4.2.2 Ostacolo puntiforme comandato mediante mouse

Per testare in condizioni più complesse l'algoritmo di collision avoidance implementato e perfezionato si eliminano gli ostacoli puntiformi fissi e si scrive uno script Matlab che permetta di **movimentare** con input esterno proveniente **da** un **mouse l'ostacolo** considerato ancora puntiforme. Attraverso un mouse è possibile dare un controllo lungo due coordinate spaziali pertanto la terza coordinata (quota in altezza su z) è posta fissa per semplicità. Con il comando

in real time dell'ostacolo puntiforme, pensato per mettere alla prova l'efficacia dell'algoritmo di anticollisione, sorgono alcune osservazioni importanti. Innanzitutto, la necessità di movimentare l'ostacolo richiede la necessità di visualizzare la sua posizione corrente e quella del manipolatore e se si visualizza il robot mediante la rappresentazione grafica fornita dal toolbox di Corke la rappresentazione risulta rallentare notevolmente l'intera esecuzione dell'algoritmo. Per ovviare tale problema si può utilizzare una rappresentazione più semplificata dell'ostacolo e del robot.

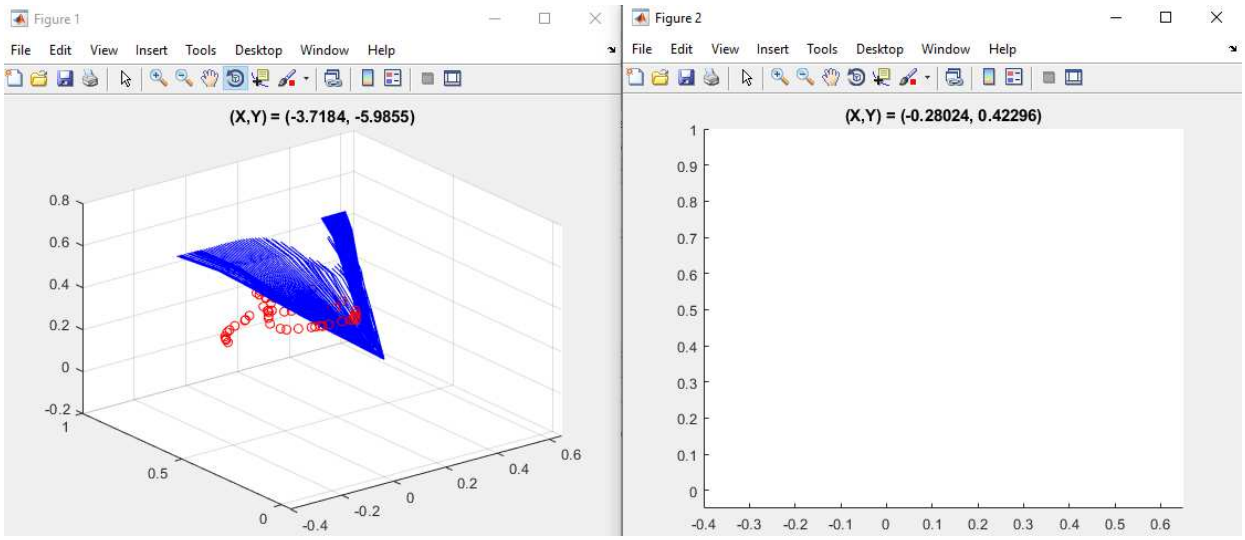


Figura 40 Pannello di controllo per il test dell'algoritmo di anticollisione con ostacolo comandato mediante mouse

Nella Figura 40 si riportano le due finestre di dialogo con Matlab per l'esecuzione di un test di efficacia dell'algoritmo di anticollisione con ostacolo puntiforme comandato mediante mouse. Sulla sinistra si vede una rappresentazione molto semplificata del robot mediante una serie di segmenti blu, la figura riporta tutte le pose del manipolatore nei vari istanti temporali sovrapposte sul medesimo plot. Inoltre, sempre a sinistra, si ritrova, indicato con un cerchietto rosso la posizione corrente dell'ostacolo sovrapposta alle precedenti posizioni temporali. Sulla destra, invece, viene fatto scorrere il mouse per acquisire la posizione corrente del mouse che viene utilizzata per spostare l'ostacolo in modo interattivo ed in real-time.

Si osservi che, sebbene i test illustrati facciano riferimento ad uno o più ostacoli puntiformi, in modo semplice è possibile estendere l'applicazione a **ostacoli sferici**. Infatti, con tale semplice forma geometrica le uniche informazioni necessarie sono la posizione del centro della sfera ed il suo raggio. Definito il centro della sfera coincidente con il punto a cui si fa riferimento nei paragrafi precedenti e scelto un raggio arbitrario è possibile calcolare l'azione repulsiva considerando come distanza dall'ostacolo la differenza della posizione del punto sul manipolatore e del centro della sfera, infine, sottraendo il raggio della sfera a tale distanza è possibile estendere l'algoritmo di anticollisione ai corpi sferici oltre che agli ostacoli puntiformi.

Con tale ambiente di prova si può testare l'efficacia dell'algoritmo di anticollisione in condizioni di diversa traiettoria e diversa velocità data all'ostacolo puntiforme. Dai vari test condotti è possibile constatare che il manipolatore segue la traiettoria pianificata

compatibilmente con l'avvicinamento ed il disturbo arrecato dall'ostacolo comandato facendo comunque prevalere la sicurezza di non collidere con lo stesso alla necessità di seguire la traiettoria rettilinea e riprendendola a seguire fedelmente ogni qualvolta l'ostacolo si allontana ad una certa distanza di sicurezza.

Si riportano ora i **risultati** di un test eseguito con ostacolo puntiforme liberamente controllato mediante mouse.

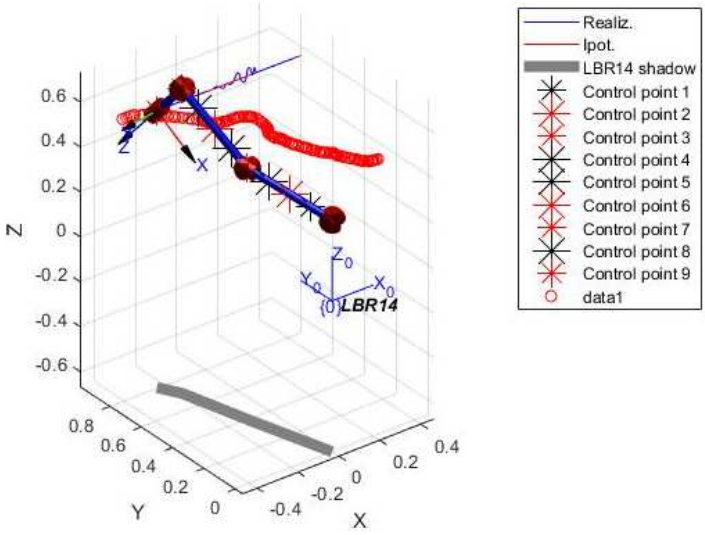


Figura 41 Confronto percorso pianificato e percorso eseguito

Il primo elemento da esaminare è il percorso effettivamente eseguito dall'EE del manipolatore in confronto alla traiettoria rettilinea pianificata. In Figura 41 è riportato il manipolatore schematizzato con la rappresentazione del toolbox di Corke nella posa finale, la traiettoria dell'ostacolo comandato mediante mouse e le due traiettorie dell'EE: quella eseguita e quella pianificata. Si noti come il manipolatore si discosti dalla traiettoria pianificata solo quando l'ostacolo è vicino ad uno dei punti di controllo presi sul manipolatore per calcolare la distanza ostacolo – robot.

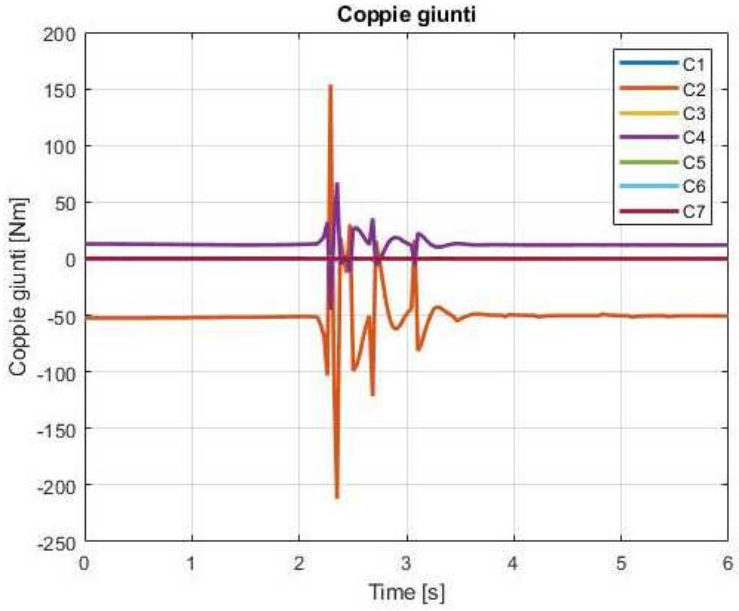


Figura 42 Coppie erogate

In Figura 42 sono riportate le coppie erogate dai motori elettrici di ciascun giunto stimate mediante il metodo ricorsivo di Newton - Eulero. Si noti una notevole variazione delle coppie erogate a causa dell'intervento dell'algoritmo di anticollisione nell'intervallo di tempo tra 2 e 3 secondi.

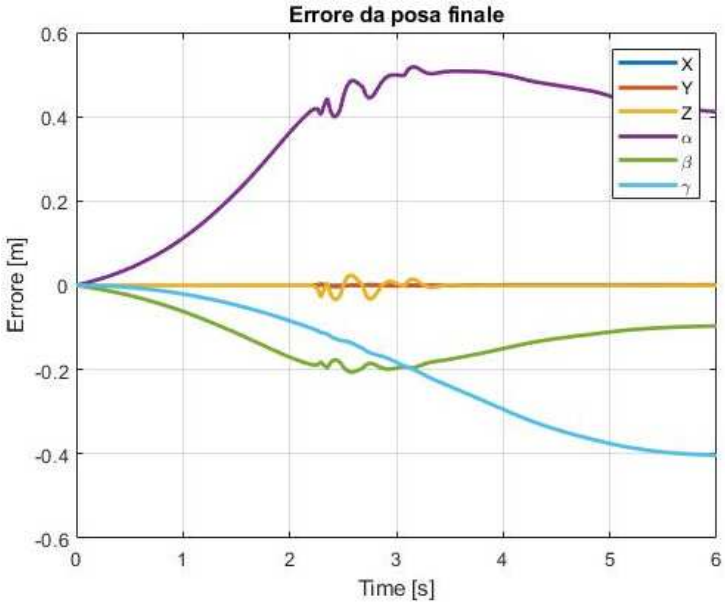


Figura 43 Errore da posa finale

Il grafico in Figura 43 riporta l'errore di posa finale del manipolatore. Si noti che i parametri più grandi in valore assoluto sono i tre angoli che individuano l'orientazione dell'EE in quanto, la traiettoria pianificata, considera il manipolatore 4 volte ridondante perchè è pianificata solo definendo la posizione dell'EE e non l'orientazione.

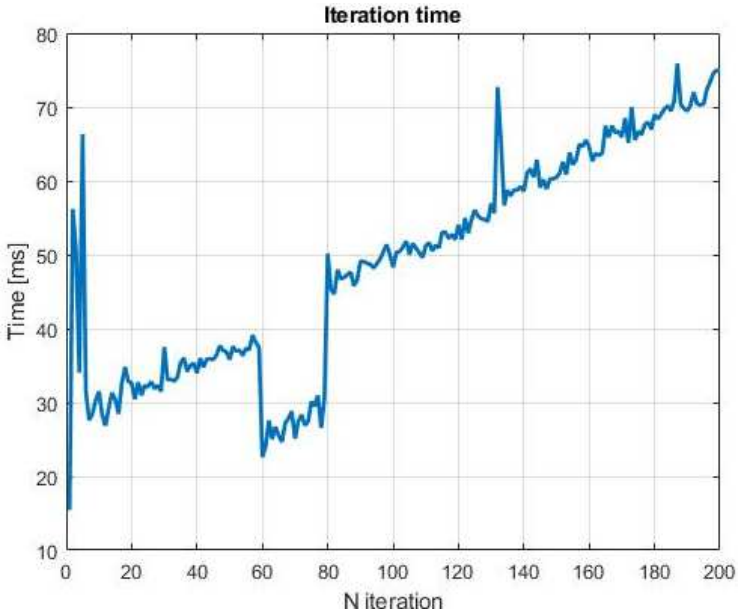


Figura 44 Tempo di iterazione

Il grafico sopra riportato indica il tempo necessario per il calcolo degli angoli di giunto da imporre al manipolatore in fase di controllo mediante l'algoritmo di anticollisione a ciascuna iterazione temporale. Si noti che il trend crescente di tale tempo è causato del fatto che le

variabili utilizzate (angoli, velocità di giunto ecc.) aumentano la loro dimensione da un istante temporale al successivo per una successiva operazione di post-processing e rappresentazione grafica delle variabili. Pertanto, se si deve stimare il tempo necessario per l'esecuzione dell'algoritmo di anticollisione occorre fare riferimento alle prime iterazioni in cui si riscontra un tempo di circa 30 millisecondi e pertanto una frequenza di circa 30 Hz. Si noti, inoltre, che sull'asse x del grafico è riportato il numero di iterazione che è pari a 200. Infatti, la traiettoria pianificata è una traiettoria rettilinea che deve essere eseguita in 6 secondi con una discretizzazione di 200 iterazioni pertanto viene definito un set di angoli di giunto ogni 0.03 secondi.

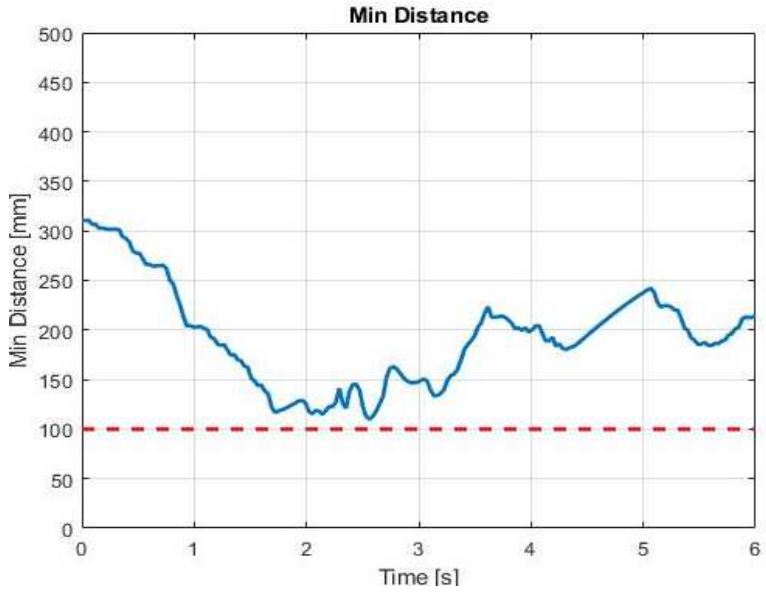


Figura 45 Minima distanza ostacolo da punti caratteristici del manipolatore

In Figura 45 è riportata la minima distanza dell'ostacolo dai punti caratteristici considerati presi sul manipolatore. Si noti che sono stati definiti i parametri caratteristici dell'algoritmo di anticollisione per garantire una minima distanza dell'ostacolo dal manipolatore di 10 cm.

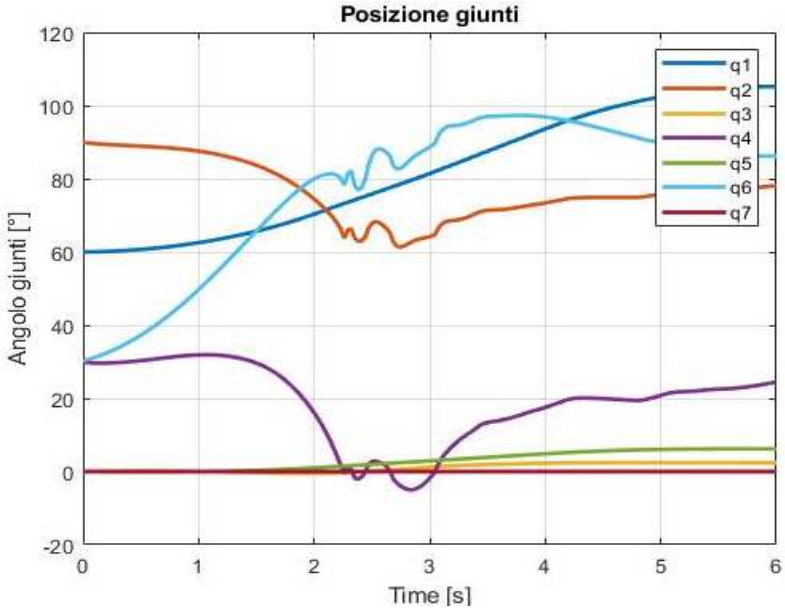


Figura 46 Angoli di giunto

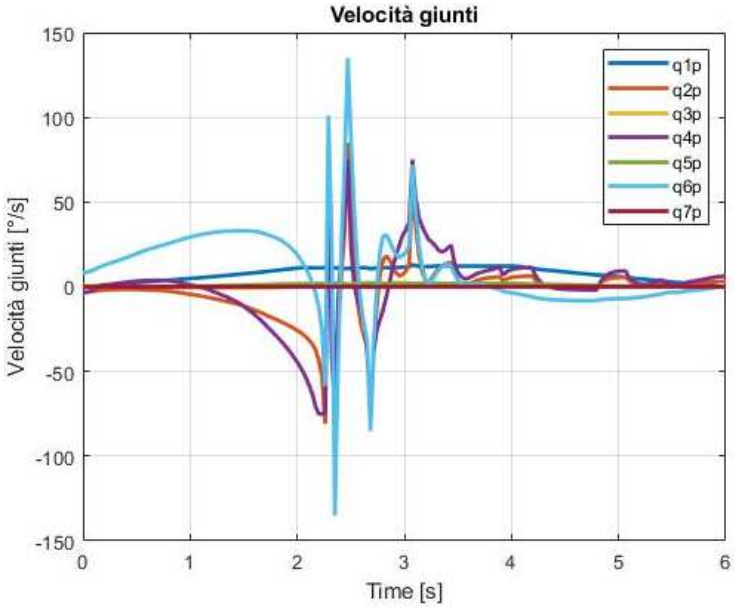


Figura 47 Velocità di giunto

Gli ultimi due grafici riportati indicato rispettivamente gli angoli di giunto e le velocità di giunto. Si noti un intervento dell'algoritmo di anticollisione abbastanza repentino e delle velocità di giunto che, in alcuni istanti, raggiungono il massimo valore ammissibile.

4.3 Cumulative Danger Field

4.3.1 Calcolo CDF in una griglia di punti equispaziati

Per l'implementazione dell'algoritmo di collision avoidance basato sul CDF il primo obiettivo da proporsi è quello di calcolare il CDF in alcuni punti dello spazio. A tale proposito è possibile considerare una **griglia equispaziata** di punti nello spazio e valutare il livello di pericolo in ognuno di essi. Tale procedura si rende necessaria per tre motivi:

1. Per verificare la capacità di calcolare tale parametro avendo un'indicazione sulla correttezza dell'implementazione;
2. Per osservare la variabilità nello spazio del campo di pericolo in funzione dei movimenti del manipolatore;
3. Per constatare il tempo impiegato dal calcolatore per eseguire tale operazione.

Nell'applicazione di tale procedura si schematizza il manipolatore robotico con una serie di punti ritenuti caratteristici del manipolatore stesso.

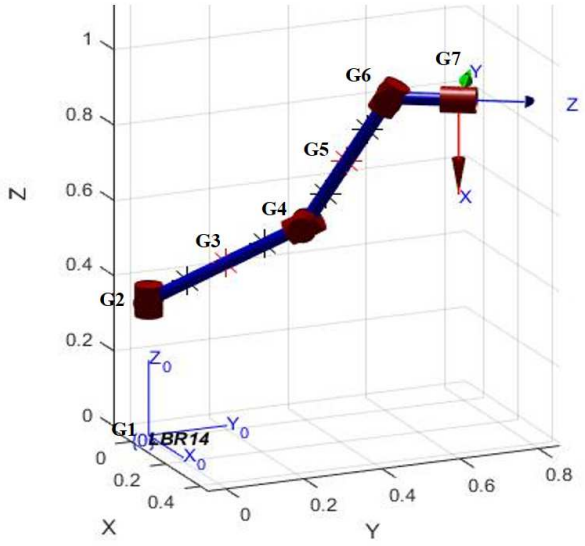


Figura 48 Schema con denominazione dei giunti utilizzati per descrivere il manipolatore

In Figura 48 è riportato uno schema del manipolatore con la denominazione dei vari giunti dall'1 al 7. Inoltre, sono evidenziati in nero altri quattro punti aggiuntivi presi in considerazione per la **discretizzazione del manipolatore** stesso. Si osservi che i punti aggiuntivi sono in posizione intermedia tra due giunti consecutivi ma non sono nel punto medio di tutti i link, infatti, non è considerato nessun punto intermedio sul link tra i giunti 1 e 2 in quanto non è possibile applicare nessuna azione repulsiva in questa zona se si ipotizza il manipolatore su base fissa. Inoltre, tali punti intermedi non sono presi sull'ultimo link in quanto di dimensione molto ridotte rispetto ai restanti bracci del manipolatore.

Fissata questa convenzione di denominazione dei punti caratteristici del manipolatore, per verificare la correttezza di calcolo del CDF con la procedura implementata si parte dal calcolare il Danger Field prodotto in questa griglia di punti per effetto di un solo punto del manipolatore (giunto numero 3 o G3 nella figura precedente). Successivamente, verificata la correttezza del calcolo, si definisce il Cumulative Danger Field come la somma dei contributi di una serie di punti presi, prima su un singolo link (link compreso tra G3 e G4) del manipolatore e poi di tutto il manipolatore.

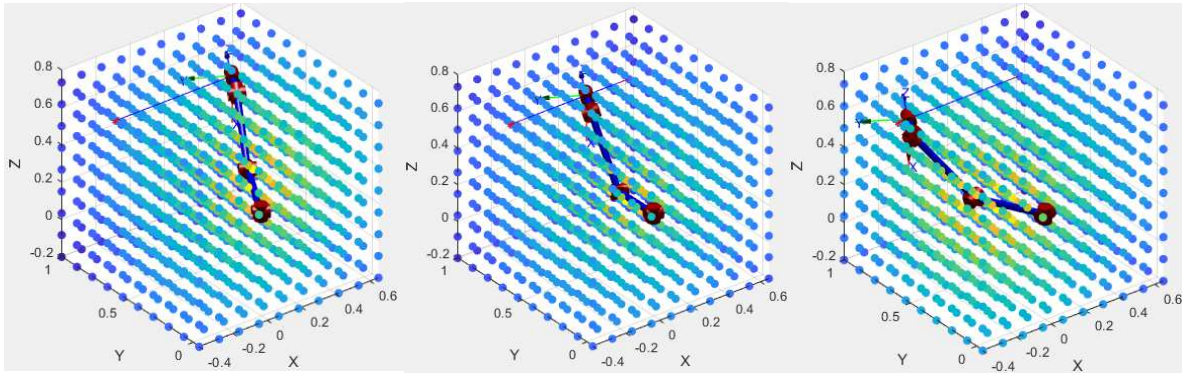


Figura 49 Evoluzione del CDF calcolato sul solo link 3-4 durante l'esecuzione di una traiettoria rettilinea (vari istanti temporali)

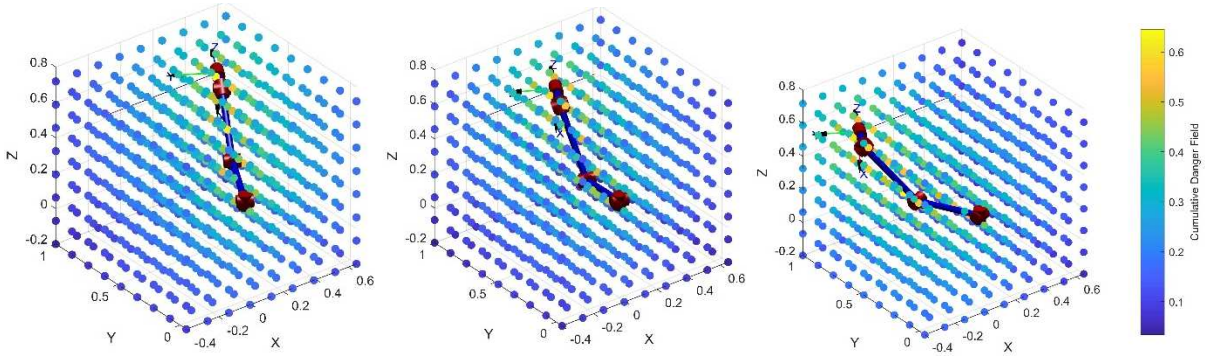


Figura 50 Evoluzione del CDF calcolato su tutto il manipolatore durante l'esecuzione di una traiettoria rettilinea (vari istanti temporali)

A valle di tale processo di calcolo ed implementazione del solo Cumulative Danger Field è possibile constatare:

1. L'effettiva valutazione di un livello di pericolo maggiore in corrispondenza dei punti più vicini al manipolatore e dei punti, dello stesso, che risultano avere una velocità assoluta maggiore (in quanto, i punti della griglia spaziale considerata sono fermi quindi non è stato necessario, in questa fase valutare l'effetto della velocità relativa ostacolo-robot);
2. I tempi di calcolo, del Cumulative Danger Field, su un numero così esteso di punti (griglia spaziale 8 x 8 x 8) non permette una implementazione di successivi algoritmi di anticollisione in real time. Infatti, sono richiesti diversi secondi, ad ogni step temporale per il solo calcolo del CDF in tutti questi punti considerati. Pertanto, in seguito, sarà opportuno andare a discretizzare in modo oculato gli ostali, o gli oggetti, rispetto al quale calcolare il CDF perché non è computazionalmente possibile il calcolo di tale parametro in punti arbitrari dello spazio.

Si noti che nelle figure riportate sono contraddistinti, con colore maggiormente tendente al rosso, i punti dello spazio in cui si ha maggiore pericolo, mentre le zone blu sono quelle in cui il CDF assume valore inferiore. Si noti inoltre che, in questa fase preliminare, non si è posti attenzione sulla scala di variabilità del CDF, in fase successiva, invece, si è intervenuti su tale fattore usano i parametri di scala caratteristici del CDF come visto nei paragrafi precedenti (coefficienti k_1 e k_2).

In conclusione, al termine di questa prima fase si è valutato il CDF in una griglia di punti spaziali equamente distribuiti, tenendo conto di un numero di punti discreti presi sul manipolatore e in ogni istate temporale dell'esecuzione di una traiettoria, a questo punto occorre implementare l'algoritmo per sfruttare tale indice di pericolo e modificare la traiettoria precedentemente pianificata, compatibilmente con l'algoritmo di anticollisione.

4.3.2 Ostacolo/ostacoli puntiformi fissi

Come descritto nel paragrafo 3.2.2, avendo a disposizione un manipolatore ridondante, è possibile usare il CDF come parametro per intervenire sulle velocità di giunto appartenenti allo spazio nullo $\dot{\boldsymbol{q}}_0$. In questo modo, compatibilmente con lo svolgimento della traiettoria precedente pianificata per l'EE del manipolatore, è possibile evitare la collisione con gli ostacoli eventualmente presenti del workspace del manipolatore stesso.

Per un primo semplice utilizzo del campo di pericolo è possibile considerare un **oggetto puntiforme** in una **posizione** spaziale **fissa** e nota come descritto nel paragrafo 4.2.1.

Anche in questo caso si può constatare come il manipolatore esegua correttamente la traiettoria rettilinea implementata evitando l'ostacolo mediante rotazione del gomito. Per testare ulteriormente l'algoritmo di anticollisione si esegue la medesima prova con **due oggetti puntiformi fissi** nello spazio riscontrando anche qui risultati analoghi a quelli del paragrafo 4.2.1. Si prosegue nello stesso modo con prove in cui **ostacoli** puntiformi sono inseriti in posizioni tali da **intralciare la traiettoria** pianificata. In questo caso, il manipolatore si allontana dalla traiettoria rettilinea implementata per non urtare contro l'ostacolo per poi ritornare sulla stessa traiettoria una volta superato l'ostacolo. Questo fenomeno è dovuto all'implementazione della sospensione del task in esecuzione secondo le indicazioni ricavate dall'articolo (60) e descritte nel paragrafo 3.2.2.

4.3.3 Ostacolo puntiforme con traiettoria precedentemente impostata

Proseguendo con le prove per testare il funzionamento corretto dell'algoritmo di anticollisione implementato si è deciso di mettere in **movimento** l'**oggetto puntiforme** con una traiettoria rettilinea preimpostata.

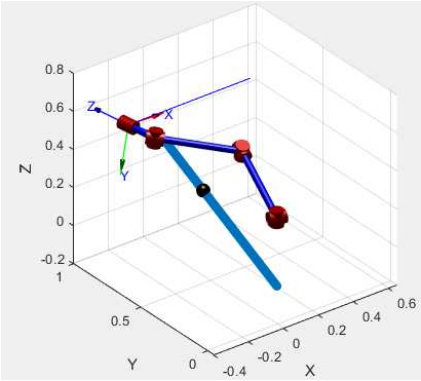


Figura 51 Collision avoidance con oggetto puntiforme in movimento su traiettoria rettilinea (caso non critico)

Si conduce un primo test senza far intersecare la traiettoria dell'ostacolo da evitare con la traiettoria del manipolatore. Sebbene però l'ostacolo puntiforme segue una traiettoria tale da non intralciare l'EE, è evidente l'intervento dell'algoritmo di anticollisione nello spostamento del gomito del manipolatore in posizione tale da essere quanto più possibile distante dalla traiettoria che sta seguendo l'ostacolo in movimento.

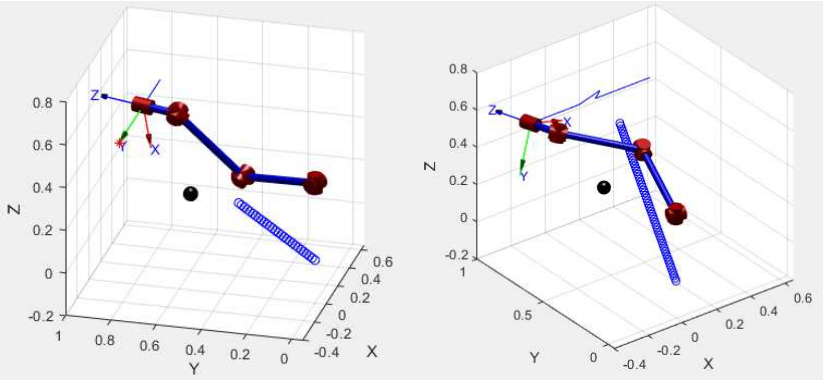


Figura 52 Collision avoidance con oggetto puntiforme in movimento su traiettoria rettilinea (caso critico)

Nel secondo test condotto si cerca di analizzare come risponde l'algoritmo a fronte di un ostacolo che intralci il gomito del manipolatore. Anche in questo caso interviene la task suspension. Il manipolatore, per evitare l'urto contro l'oggetto mobile, risponde con un allontanamento del gomito dall'ostacolo in direzione opposta rispetto alla traiettoria che sta seguendo l'oggetto per poi ritornare sulla traiettoria preimpostata una volta terminato il pericolo di collisione. Da questo comportamento emerge un'importante osservazione, ovvero, l'algoritmo di anticollisione interviene con un allontanamento del robot nella medesima direzione ma in verso opposto rispetto alla traiettoria che sta seguendo l'ostacolo da evitare. Invece, in una condizione come la precedente, sarebbe più opportuno che il manipolatore ruotasse maggiormente il gomito per evitare la collisione essendo la traiettoria dell'ostacolo rettilinea. Pertanto, si potrebbe pensare ad un miglioramento dell'implementazione di tale algoritmo cercando di analizzare la storia temporale della posizione dell'ostacolo per valutarne preventivamente la traiettoria ed intervenire in modo più smart sulla posizione del manipolatore. Tale osservazione, sebbene complessa da implementare, apre comunque le porte ad un possibile miglioramento dell'algoritmo utilizzato.

4.3.4 Miglioramento task suspension e implementazione ritardo risposta

Dalle prove eseguite è possibile constatare che, nel momento in cui l'ostacolo si avvicina all'EE o ad un altro punto del corpo del manipolatore, interviene la sospensione del task descritta nel paragrafo 3.2.2. Però, avendo realizzato un modello di manipolatore esclusivamente cinematico, la risposta dello stesso manipolatore risulta immediata nell'allontanarsi dall'ostacolo dimostrando la realizzazione di una traiettoria non regolare e poco verosimile.

Per quanto concerne la **task suspension**, si può passare dall'implementazione del coefficiente m presente nella relazione:

$$m = \begin{cases} 1 & se \ \| \boldsymbol{CDF}(\boldsymbol{r}) \| \leq \Delta \\ 0 & altrimenti \end{cases} \tag{4.1}$$

Alla relazione:

$$m = \begin{cases} 1 & se \ \| \boldsymbol{CDF}(\boldsymbol{r}) \| \leq \Delta(1-\varepsilon) \\ 0 & se \ \| \boldsymbol{CDF}(\boldsymbol{r}) \| > \Delta(1+\varepsilon) \\ \frac{1}{2} - \frac{1}{2}\sin\frac{\pi(\| \boldsymbol{CDF}(\boldsymbol{r}) \| - \Delta)}{2\varepsilon\Delta} & altrimenti \end{cases} \tag{4.2}$$

Dove: ε è una costante reale positiva piccola (solitamente posta $= 0.9$ per avere una variabilità più graduale possibile del coefficiente m);

Δ indica la soglia massima di CDF oltre la quale applicare la task suspension.

Con tale relazione è evidente una più graduale variabilità del contributo dovuto al potenziale repulsivo nei confronti degli ostacoli da evitare. Ad esempio, assumendo un valore di $\varepsilon = 0.9$ come accennato risulta $m = 0 \ se \| \boldsymbol{CDF}(\boldsymbol{r}) \| > 1.9\,\Delta$, $m = 1 \ se \| \boldsymbol{CDF}(\boldsymbol{r}) \| \leq 0.1\,\Delta$ e nel range pari a $1.9\,\Delta - 0.1\,\Delta = 1.8\,\Delta$ si ha una variazione graduale di m tra 0 ed 1.

Oltre tale accorgimento si inserisce un **modello dinamico del primo ordine** per esplicitare la variazione degli angoli di giunto tra un'iterazione e la successiva in modo da rendere ancora più graduale e verosimile la risposta del manipolatore alle variabili di giunto calcolate mediante pianificazione della traiettoria e algoritmo di anticollisione e usate come parametri di controllo della posizione del manipolatore. Avendo il manipolatore reale delle variabili di giunto, velocità di giunto e coppie erogabili limitate è verosimile implementare una evoluzione temporale delle velocità di giunto di tipo dinamico del primo ordine caratterizzata da una costante di tempo che individua il tempo di risposta del manipolatore stesso. A tal proposito le velocità di giunto si possono calcolare mediante la relazione:

$$\dot{\boldsymbol{q}}(t) = \dot{\boldsymbol{q}}(t - \Delta t) + \left(1 - e^{-\frac{\Delta t}{\tau}}\right)(\dot{\boldsymbol{q}}(t) - \dot{\boldsymbol{q}}(t - \Delta t)) \tag{4.3}$$

Dove: $\dot{\boldsymbol{q}}(t)$ è il vettore delle velocità di giunto all'istante corrente;

$\dot{\boldsymbol{q}}(t - \Delta t)$ è il vettore delle velocità di giunto all'istante precedente;

Δt è l'intervallo di tempo tra una iterazione e la successiva;

τ è la costante di tempo caratteristica del sistema dinamico del primo ordine.

a) b)

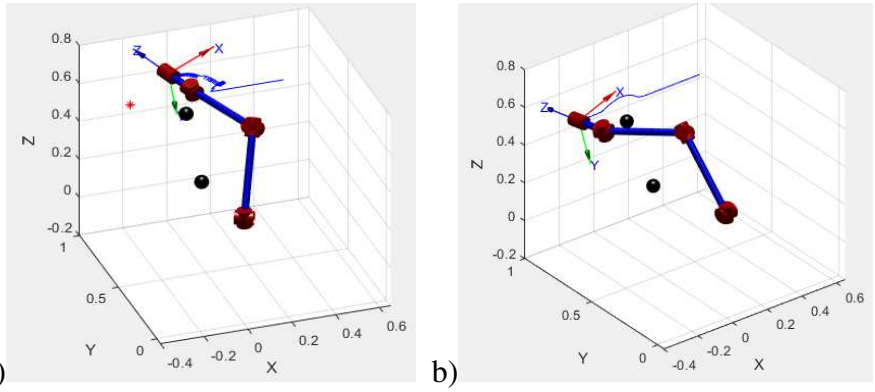


Figura 53 Traiettoria del manipolatore a) prima b) dopo l'implementazione del ritardo nella risposta

La maggiore gradualità nel calcolo delle velocità di giunto appartenenti allo spazio nullo, mediante il nuovo metodo di calcolo del coefficiente m e l'implementazione di una risposta

con una dinamica del primo ordine, in termini di velocità di giunto, ha permesso la risoluzione di alcune instabilità presenti precedentemente e il perfezionamento dell'algoritmo di controllo del manipolatore.

4.3.5 Presenza di velocità relativa ostacolo - manipolatore

Sebbene nel capitolo precedente è stato descritto l'algoritmo di controllo prevedendo il calcolo del CDF mediante l'uso di **velocità relativa**. Nella prima fase di implementazione (descritta nel presente capitolo) usando ostacoli fissi nello spazio o al più con moto rettilineo uniforme, è stato trascurato l'effetto della velocità relativa. Per un miglioramento dell'algoritmo è però necessario procedere inserendo nel calcolo del DF la velocità relativa tra oggetto e corrispondente punto sul manipolatore. Infatti, sebbene considerare la velocità assoluta dell'oggetto è un miglioramento rispetto alla sola valutazione della distanza non è una soluzione ottimale in quanto comunque il manipolatore robotico è soggetto a movimenti quindi tanto il modulo quanto la direzione della velocità relativa sono profondamente influenzati non solo dall'ostacolo ma anche dallo stesso robot.

Per perseguire tale scopo è possibile esplicitare tale velocità relativa mediante la seguente relazione:

$$\boldsymbol{v} = \boldsymbol{v_w} - \boldsymbol{v_{o'}} - \boldsymbol{S}(\omega)(\boldsymbol{r_w} - \boldsymbol{r_{o'}}) \tag{4.4}$$

Dove: $\boldsymbol{v}$ è il vettore velocità relativa ad un punto sul robot di un generico punto nello spazio;

$\boldsymbol{v_w}$ e $\boldsymbol{r_w}$ sono i vettore posizione e velocità dell'ostacolo espressi nel sistema di riferimento assoluto (world);

$\boldsymbol{S}(\omega)$ è la matrice anti-simmetrica delle velocità angolari;

$\boldsymbol{v_{o'}}$ e $\boldsymbol{r_{o'}}$ sono il vettore posizione e velocità del punto sul manipolatore espressi nel sistema di riferimento assoluto.

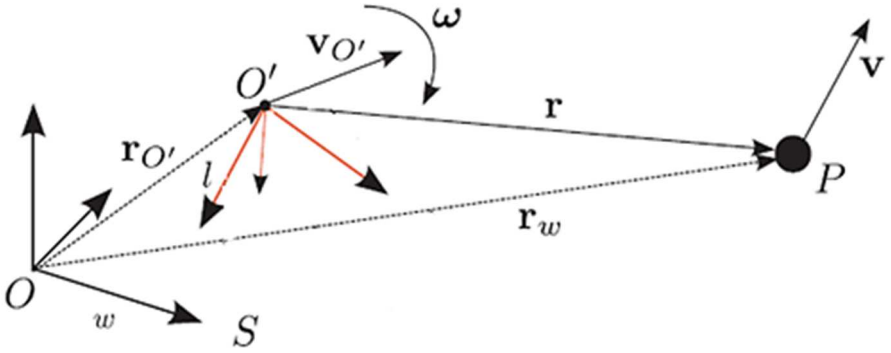


Figura 54 Rappresentazione geometrica delle quantità indicate

4.3.6 Ostacolo puntiforme comandato mediante mouse

Per testare in condizioni più complesse anche l'algoritmo di collision avoidance che fa uso del CDF si eliminano gli ostacoli puntiformi fissi e con traiettoria prestabilita e si utilizza il medesimo script Matlab, descritto in precedenza, che permetta di movimentare con input esterno proveniente da un mouse l'ostacolo.

Anche in questo caso, come visto in precedenza, si dimostra l'efficacia dell'algoritmo di intervenire quando necessario e di riportare, compatibilmente con il livello di pericolo rilevato, il manipolatore sulla traiettoria pianificata. In questo caso, però, in aggiunta rispetto a quanto visto precedentemente, l'algoritmo di anticollisione interviene, non solo quando l'ostacolo si trova ad una distanza inferiore al valore di sicurezza ma anche quando l'ostacolo, seppur lontano, si sta avvicinando con velocità considerevole. Infatti, ricordando la definizione di CDF, con opportuni coefficienti di peso, esso valuta tanto la distanza ostacolo-manipolatore quanto la velocità relativa.

Si esaminano ora una serie di **risultati** ottenuti testando l'algoritmo di anticollisione basato sul CDF utilizzando la medesima traiettoria dell'ostacolo adottata nel test condotto con l'approccio dei potenziali in modo da poter fare dei confronti tra i due algoritmi di anticollisione.

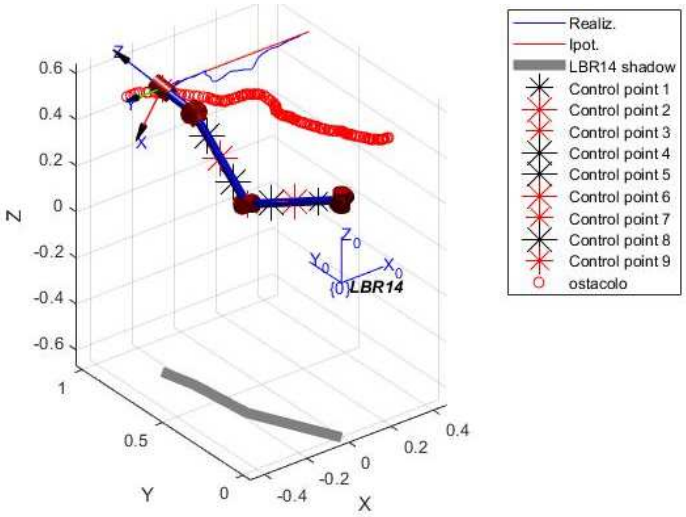


Figura 55 Confronto percorsi

Si riporta anche in questo caso, come primo risultato, il confronto tra la traiettoria pianificata e la traiettoria effettivamente realizzata dal manipolatore per evitare l'ostacolo in movimento comandato mediante muse. Si noti che, rispetto all'algoritmo implementato mediante potenziali esaminato nei precedenti paragrafi, il manipolatore si allontana maggiormente dalla traiettoria implementata a causa della task suspension implementata.

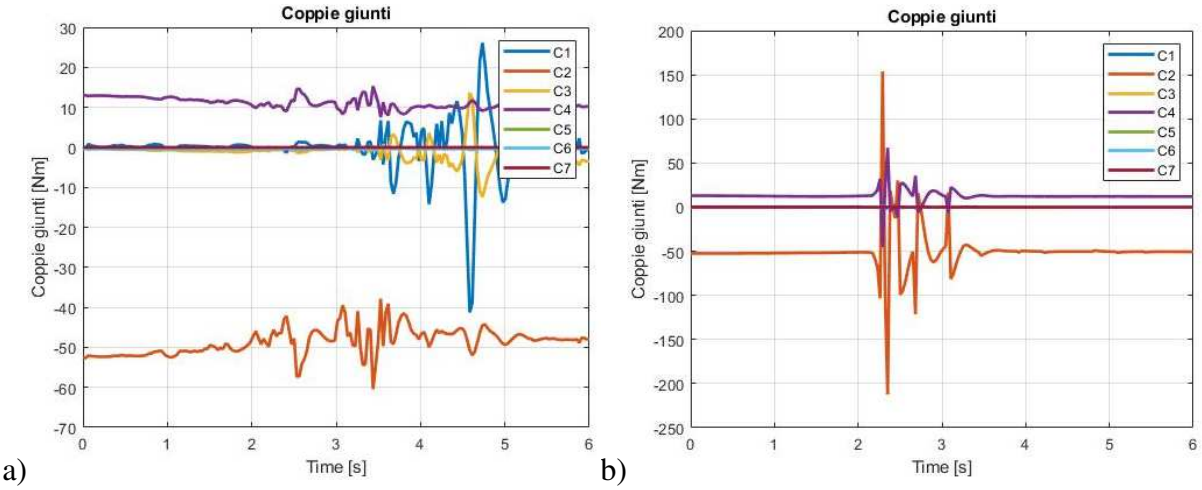


Figura 56 a) Coppie erogate con CDF b) Coppie erogate con algoritmo potenziali

Per quanto concerne le coppie erogate dai motori elettrici di ciascun giunto, si riporta un confronto con l'algoritmo di anticollisione basato sui potenziali. Si noti che, a causa della task suspension implementata, il manipolatore ha dei picchi di coppia erogata inferiori nel caso di algoritmo basato sul CDF rispetto a quello basato sui potenziali.

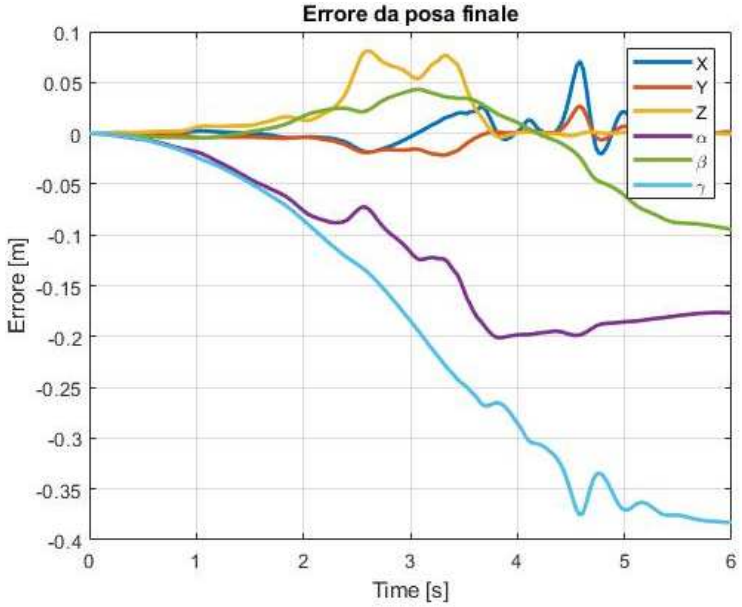


Figura 57 Errore da posa finale

In Figura 57 si riportano gli errori da posa finale del manipolatore. Si noti che anche in questo caso, come fatto con l'algoritmo di anticollisione basato sui potenziali, si è pianificata la traiettoria rettilinea solo definendo la posizione dell'EE e non l'orientazione.

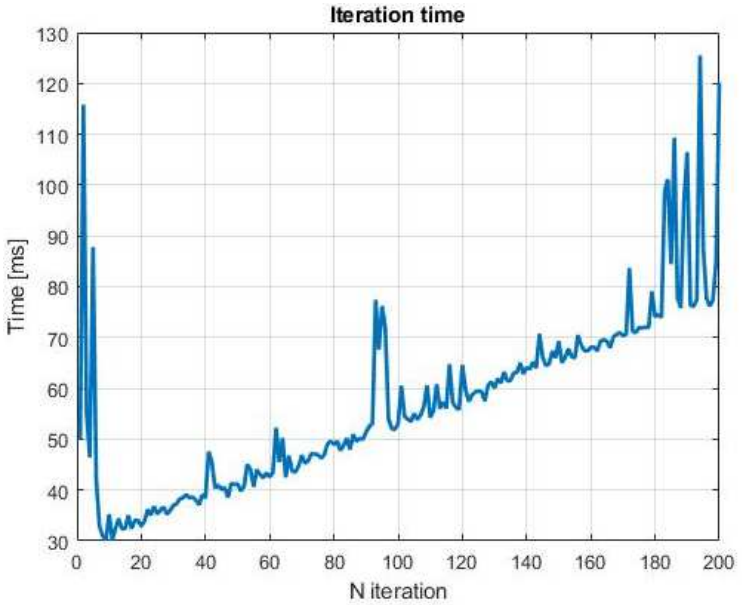


Figura 58 Tempo di iterazione

Per quanto concerne i tempi di iterazione, si riscontra, anche in questo caso, un trand crescente a causa dell'incremento delle dimensioni delle variabili utilizzate. In ogni caso, la frequenza con cui è eseguito questo controllo è solo leggermente superiore a quella che si ha nel caso di

algoritmo basato sui potenziali. Infatti, qui si ha una frequenza di aggiornamento degli angoli di giunto di set di poco inferiore ai 30 Hz.

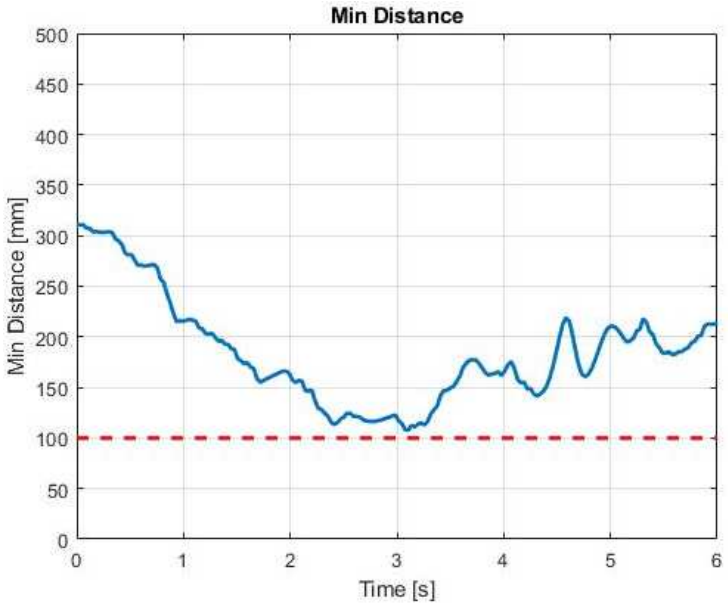


Figura 59 Minima distanza tra ostacolo e punti caratteristici del manipolatore

Per un confronto con il precedente algoritmo, anche in questo caso, sono state fissate le variabili caratteristiche dell'algoritmo di anticollisione per garantire una distanza minima dall'ostacolo di almeno 10 cm.

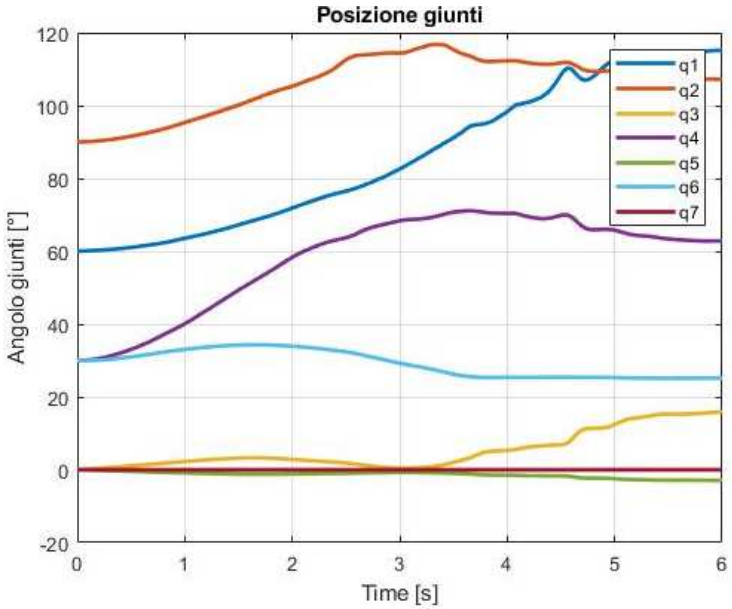


Figura 60 Angoli di giunto

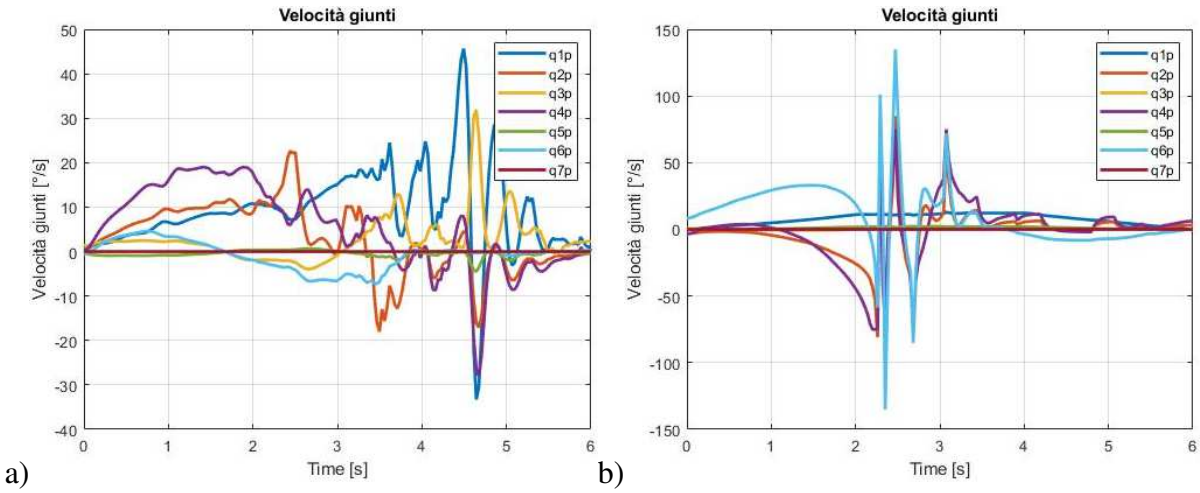


Figura 61 a) Velocità di giunto con CDF b) Velocità di giunto con algoritmo potenziali

In Figura 60 e Figura 61 si riportano gli angoli di giunto e le velocità di giunto raggiunte. Si noti, anche in questo caso, come già accennato per le coppie, che l'implementazione di una task suspension permetta di avere delle velocità di giunto che non abbiano delle variazioni troppo accentuate e non raggiungono mai i limiti di giunto.

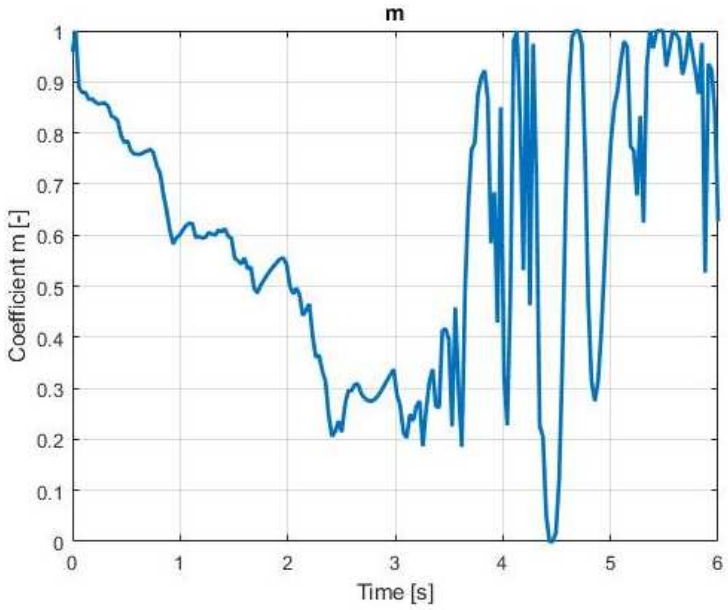


Figura 62 Coefficiente m task suspension

Si riportano, infine, i due parametri caratteristici dell'algoritmo di anticollisione basato sul campo di pericolo ovvero: il coefficiente m caratterizzante la task suspension ed il CDF che è utilizzato per calcolare le azioni repulsive da imporre ai giunti del manipolatore.

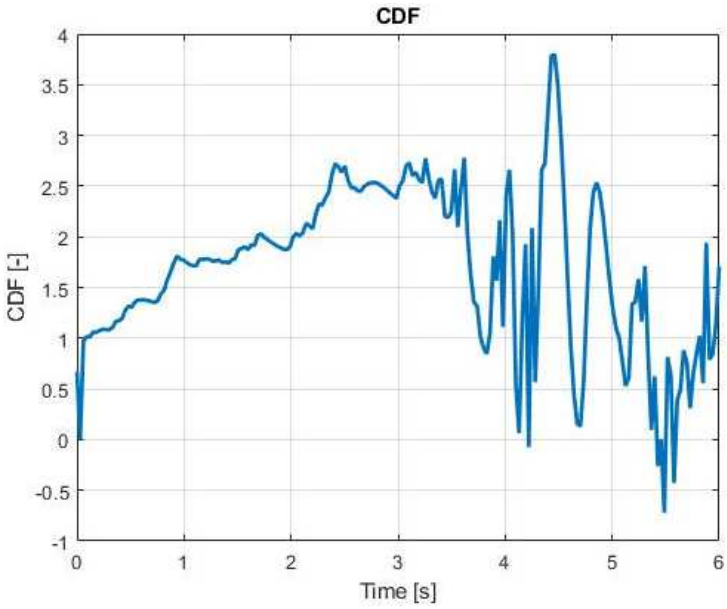


Figura 63 Cumulative Danger Field

4.4 Simulazioni in ambiente V-REP

Per una più **efficace rappresentazione** degli algoritmi sviluppati è possibile utilizzare l'ambiente di simulazione di Coppelia Robotics: V-REP. È possibile scaricare, in modo gratuito, la versione per studenti EDU V3.5.0 rev 4 ed utilizzarla per condurre simulazioni che abbiano un ambiente di visualizzazione molto più realistico rispetto a quello offerto dai plot di Matlab e dal toolbox di Corke. Infatti, con poche modifiche agli script sviluppati in ambiente Matlab, è possibile aprire una comunicazione tra Matlab e V-REP secondo un collegamento client e server e visualizzare la posa del manipolatore istante per istante e gli ostacoli, più o meno complessi che possono essere comandati tanto in ambiente Matlab (come visto in precedenza) quanto direttamente in ambiente V-REP. A tal proposito si utilizzano le funzioni:

- `VREPCommand_KUKA`: per imporre al modello virtuale del manipolatore KUKA iiwa LBR R820 la posa da fargli acquisire istante per istante in base all'algoritmo sviluppato in ambiente Matlab;
- `vrep.simxSetObjectPosition`: per imporre ad un oggetto in ambiente V-REP una posizione acquisita o definita a priori in ambiente Matlab;
- `vrep.simxGetObjectPosition`: per rilevare dall'ambiente V-REP la posizione di un oggetto e trasmetterla in ambiente Matlab e utilizzarla nell'algoritmo di anticollisione.

Con questi semplici accorgimenti è possibile implementare tutte le prove precedentemente eseguite in ambiente di programmazione Matlab. In particolare, si conducono le seguenti prove e si può constatare l'efficacia degli algoritmi di programmazione implementati e testati più volte in precedenza.

Implementazione **traiettoria rettilinea senza ostacoli**.

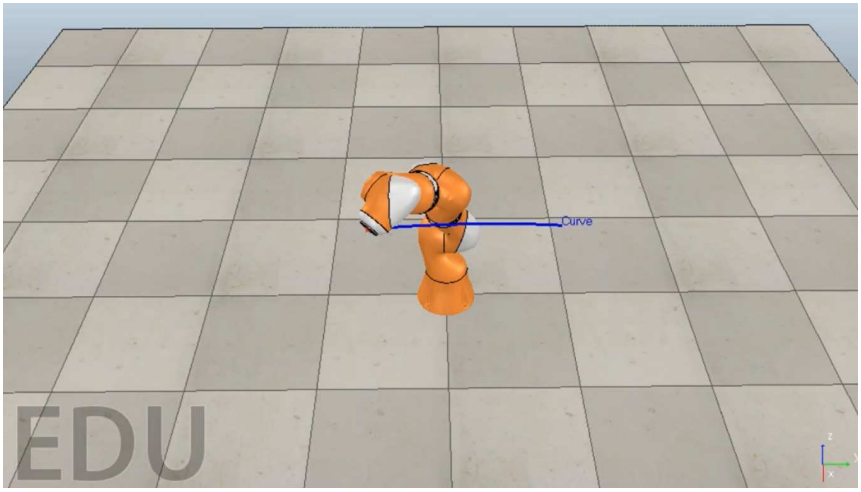


Figura 64 Un frame del robot durante l'esecuzione di traiettoria rettilinea senza ostacoli

Implementazione **traiettoria rettilinea con ostacolo puntiforme fisso**.

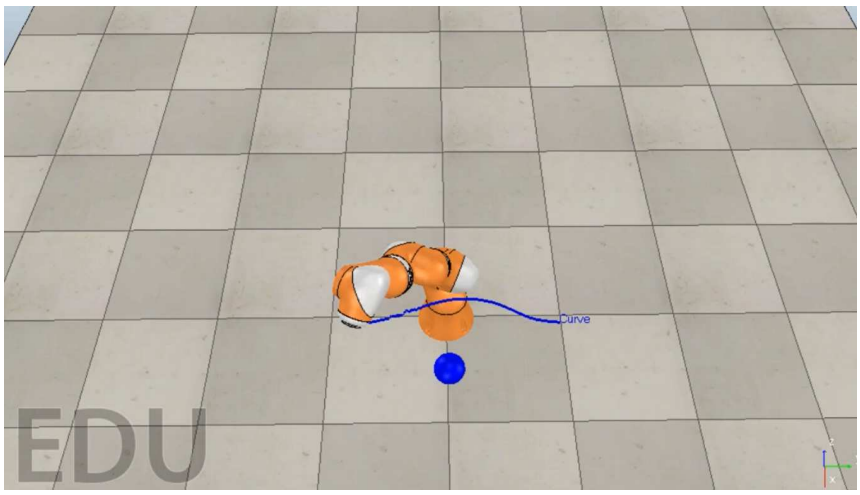


Figura 65 Un frame del robot durante l'esecuzione di traiettoria rettilinea in presenza di un ostacolo puntiforme fisso

Implementazione **traiettoria rettilinea con ostacolo puntiforme mobile**.

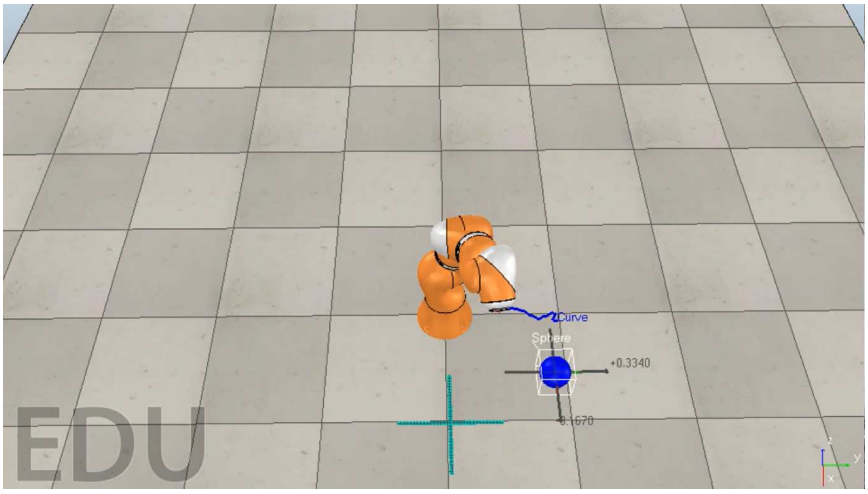


Figura 66 Un frame del robot durante l'esecuzione di traiettoria rettilinea in presenza di un ostacolo puntiforme movimentato in ambiente V-REP

4.5 Confronto tra i due algoritmo di collision avoidance

A questo punto della trattazione, i due algoritmi di collision avoidance implementati dimostrano essere qualitativamente efficaci per evitare la collisione contro un ostacolo puntiforme ma per fare un confronto tra i due occorre determinare quali **parametri** possono essere considerati **indicativi** di una buona **performance** nell'esecuzione dell'algoritmo di anticollisione. A questo proposito, è necessario considerare:

- Coppia erogata per la complessiva esecuzione del task compatibilmente al disturbo applicato;
- Valori di velocità ed accelerazione di giunto raggiunti;
- Distanza minima del manipolatore dall'ostacolo;
- Distanza media del manipolatore dall'ostacolo;
- Quanto tempo, a causa dell'intervento dell'algoritmo di anticollisione, l'EE si allontana dal task oltre una soglia prestabilita;
- Tempo di calcolo dell'algoritmo di collision avoidance.

Questi ed altri possono essere parametri indicativi di un algoritmo di anticollisione più o meno efficiente.

4.5.1 Stima coppie erogate ai giunti

Un aspetto di cui fino a questo punto non si è tenuto conto, in fase in implementazione, è quello relativo alle coppie erogate dai motori elettrici per l'applicazione di certe velocità ed accelerazioni ai giunti del manipolatore. Infatti, il modello di manipolare utilizzato per i test condotti in precedenza è un modello cinematico. Il primo step per la stima delle coppie erogate è quello di **raccogliere i dati inerziali** del manipolatore con la quale si stanno conducendo questi test al calcolatore. Per tale motivo è necessario far riferimento al sito ufficiale del produttore di robot industriali KUKA Spa. Attraverso la sezione download del sito è possibile reperire il **modello CAD** del manipolatore KUKA LBR iiwa R820. Scaricato il file è possibile aprirlo con l'ausilio di un software di modellazione CAD come Solidworks.

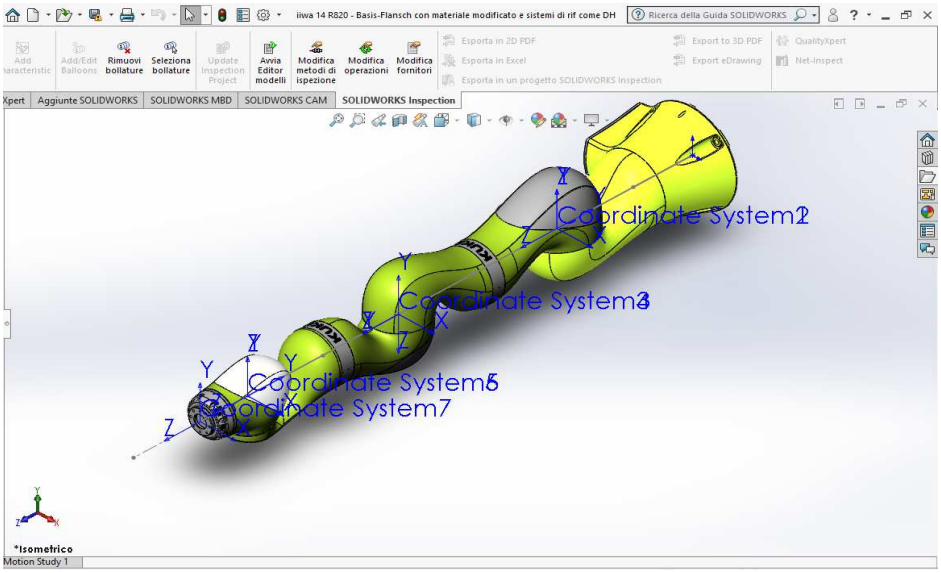


Figura 67 Modello CAD 3D KUKA LBR iiwa R820

Da questo modello 3D è possibile reperire i dati inerziali di ciascun link del robot ed in particolare:

- m_i massa dell'i-esimo link;
- $\boldsymbol{b}_{CADi}$ vettore posizione del baricentro del link i-esimo rispetto al sistema di riferimento CAD;
- $\boldsymbol{I}_{Gi}$ tensore centrale d'inerzia del link i-esimo;
- $\boldsymbol{A}_{cmCADi}$ matrice di orientazione del sistema centrale d'inerzia del link i-esimo rispetto al sistema di riferimento CAD;

Attraverso questi dati è possibile ricavare il vettore di posizione del baricentro del link i-esimo:

$$\boldsymbol{b}_{io} = \boldsymbol{A}_{CADio} \cdot [\boldsymbol{b}_{CADi}; 1] \tag{4.5}$$

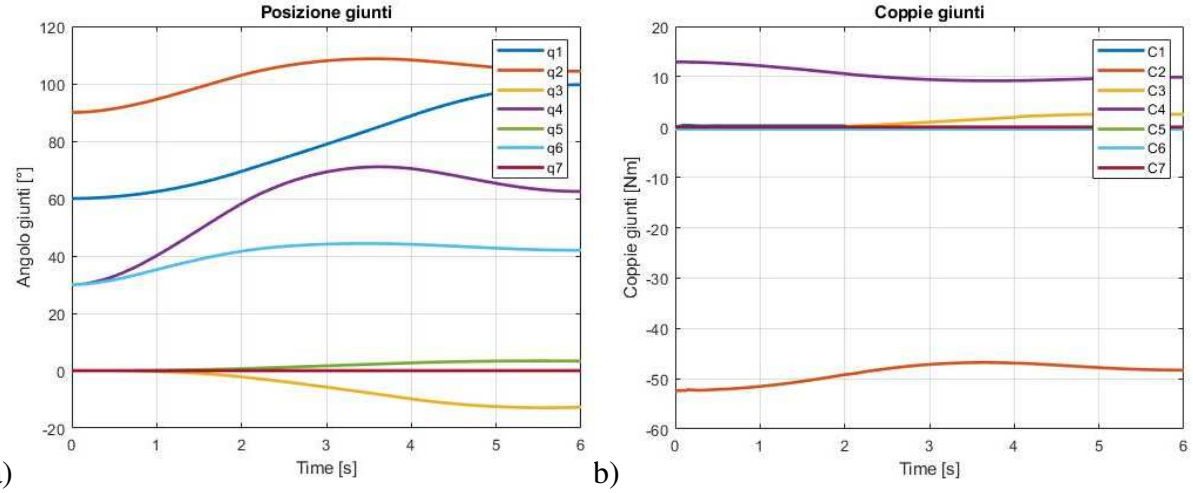
Dove: $\boldsymbol{A}_{CADio}$ è la matrice di trasformazione omogenea dal sistema di riferimento CAD a quello dell'i-esimo link.

Inoltre, è possibile calcolare i tensori d'inerzia baricentrici espressi nel sistema di riferimento del singolo link attraverso a relazione:

$$\boldsymbol{I}_i = \boldsymbol{A}_{CADio}(1{:}3; 1{:}3) \cdot \boldsymbol{A}_{cmCADi} \cdot \boldsymbol{I}_{Gi} \cdot (\boldsymbol{A}_{CADio}(1{:}3; 1{:}3) \cdot \boldsymbol{A}_{cmCADi})' \tag{4.6}$$

A questo punto, definendo il vettore di accelerazione di gravità $\boldsymbol{g} = [0\ 0\ 9{,}81]$ ed i vettori di forze ($\boldsymbol{F}_{ext}$) e momenti ($\boldsymbol{M}_{ext}$) esterni applicati al TCP (Tool Center Point) è possibile ricavare le coppie ai vari giunti mediante la function `rne_mdh_POLITO` prelevata dalla libreria di Peter Corke ed opportunamente modificata per l'implementazione del metodo iterativo, basato sulle equazioni di Newton-Eulero, che conoscendo le azioni dinamiche al TCP e le caratteristiche dinamiche del manipolatore permette di risalire alle azioni scaricate alla base e la coppia erogata dai motori dei giunti come descritto nel paragrafo 2.5.

Per testare la corretta implementazione dei parametri dinamici del manipolatore e della stima delle coppie è possibile condurre una semplice **prova** utilizzando la **traiettoria rettilinea** implementata per il robot KUKA LBR iiwa R820.



a) b)

Figura 68 a) Angolo giunti b) Coppie giunti durante l'esecuzione di un tratto rettilineo nello spazio operativo

Si può constatare che, a causa dell'assente movimento sul giunto 7 che si trova in corrispondenza dell'EE, anche la coppia erogata dal rispettivo motore è nulla. Analogamente, le coppie erogate dagli ultimi giunti (5 e 6) sono circa nulla a causa delle ridotte azioni della forza peso da vincere da parte di tali link del manipolatore. Le due coppie maggiori, in valore assoluto, risultano quelle sul giunto 4 e sul giunto 2 in quanto sono tali giunti a dover vincere il maggior contributo di forza peso e a dover spostare il manipolatore in fase di esecuzione della traiettoria rettilinea lungo l'asse x del manipolatore KUKA LBR iiwa R820.

4.5.2 Definizione dei parametri numerici caratterizzanti le performance degli algoritmi di Collision Avoidance

Oltre ai valori di coppia erogata dai motori elettrici per la movimentazione dei giunti del manipolatore tra gli altri parametri caratteristici si è citato il **tempo** per cui, a causa dell'intervento dell'algoritmo di anticollisione, l'**EE si allontana dal task** oltre una soglia prestabilita. Tale soglia, al di sotto della quale si può considerare che il manipolatore stia eseguendo esattamente la traiettoria impostata, deve tener conto dell'accuratezza con cui il manipolatore esegue la traiettoria in assenza di qualsiasi disturbo esterno. Pertanto, è possibile eseguire una prova in cui l'ostacolo comandato non generi nessun disturbo e lasci la possibilità al manipolatore di seguire la traiettoria pianificata.

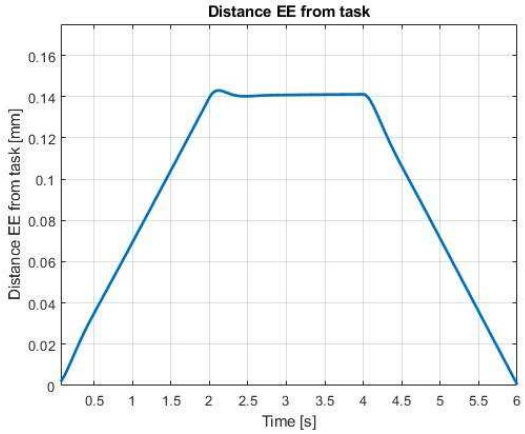


Figura 69 Distanza EE dalla traiettoria rettilinea da seguire in assenza di disturbi esterni

Quello che si può constatare da questa prova preliminare è un errore intrinseco di alcuni decimi di mm nel posizionamento dell'EE rispetto alla traiettoria pianificata. Interessante è anche il profilo di questo **errore** che è **trapezoidale**, tale andamento deriva dal profilo di velocità trapezoidale implementato per l'esecuzione della traiettoria rettilinea. In ogni caso da questa valutazione si desume che la soglia per la quale si può considerare che l'algoritmo di anticollisione stia allontanando l'EE dalla traiettoria pianificata deve essere maggiore di questo errore comunque presente.

4.6 Limiti degli algoritmi implementati e miglioramenti applicabili

Successivamente, valutata l'efficacia degli algoritmi di anticollisione presentati si possono riscontrare comunque limiti e miglioramenti effettuabili. In particolare:

- L'algoritmo di De Luca si basa esclusivamente sulla distanza robot-ostacolo;

- L'algoritmo CDF sebbene inserisca la considerazione della velocità relativa in aggiunta alla semplice distanza dall'ostacolo, per come è stato implementato tiene conto (come anche l'algoritmo di De Luca) solo di un numero di punti limitato sul manipolatore;
- Tutte le valutazioni condotte fino a questo punto hanno considerato uno o più ostacoli puntiformi (o al più sferici) ma, in realtà, un ostacolo ha una geometria complessa, soprattutto se si desidera applicare tali algoritmi considerando come ostacoli persone o parti del corpo di una persona.

Per quanto concerne questi limiti, senza dubbio, se considerare un numero di punti limitato sul manipolatore può essere una soluzione adottabile, sicuramente l'implementazione dell'algoritmo considerando ostacoli puntiformi deve essere migliorata. A tal proposito è possibile testare diversi **miglioramenti**:

- Valutazione dell'ostacolo come costituito da segmenti sui quali prendere un numero di punti discreti;
- Valutazione degli ostacoli mediante costruzione tridimensionale realizzata da voxel;
- Utilizzo di una nuvola di punti proveniente da sensori di profondità come Microsoft Kinect.

Dalla prima all'ultima di queste possibili soluzioni ovviamente il livello di accuratezza nel descrivere gli ostacoli è progressivamente maggiore e pertanto anche l'affidabilità del relativo algoritmo di anticollisione sarà maggiore.

4.6.1 Discretizzazione braccio umano mediante segmenti lineari

Un'estensione dell'algoritmo implementato, sia mediante potenziali repulsivi sia mediante CDF, è realizzabile non facendo un calcolo con ***N* punti sul manipolatore** ed uno esterno rappresentante l'ostacolo, bensì considerando ***M* punti sull'oggetto** o il corpo **da evitare**. Infatti, così come il braccio robotico è stato schematizzato come una catena cinematica di link sui quali si sono presi un totale di N punti caratteristici, allo stesso modo gli ostacoli sono gli **arti superiori** di una persona è possibile discretizzarli nello stesso modo.

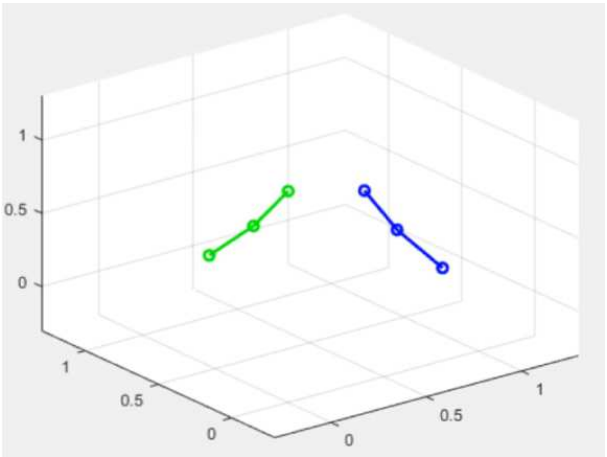


Figura 70 Esempio di schematizzazione degli arti superiori mediante segmenti

Un problema notevole però è come risalire alla posizione assoluta, nel sistema di riferimento world, dei giunti di un braccio (spalla, gomito, braccio). Per farlo ci sono molteplici espedienti,

si possono usare sistemi ad ultrasuoni, complessi sistemi di videocamere o anche un semplice sensore di profondità come **Microsoft Kinect**. Quest'ultima è una strada percorsa anche in letteratura. Infatti, attraverso un Kinect è possibile ricavare la posizione dei giunti caratteristici del corpo umano.

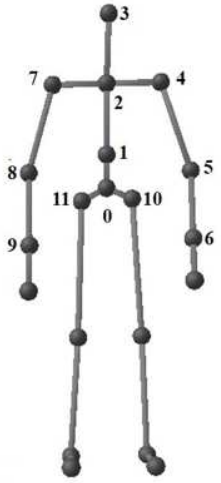


Figura 71 Schematizzazione corpo umano mediante Microsoft Kinect

Sebbene però la soluzione sembri così semplice, in realtà non lo è. Infatti, come qualsiasi misura anche quella eseguita dal sensore di profondità del Kinect è soggetta ad incertezza e rumore. Per compensare tale problema, sui dati acquisiti mediante Kinect in laboratorio sul corpo umano, si può applicare un **filtro di Kalman**. Il filtro di Kalman è un processo matematico iterativo che usa un set di equazioni ed input di dati consecutivi per stimare rapidamente il reale valore di posizione, velocità, accelerazione ecc. degli oggetti misurati quando i valori misurati contengono imprevedibili e casuali errori, incertezze e variazioni.

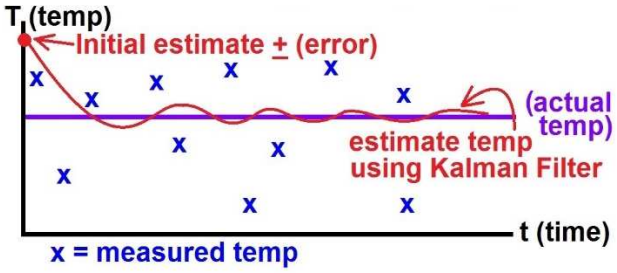


Figura 72 Esempio filtro di Kalman applicato alla stima di una temperatura

Mediante questo filtro si ottengono, in post-processing, le coordinate dei punti di interesse e si utilizzano per l'esecuzione di un algoritmo di collision avoidance più evoluto rispetto a quelli visti in precedenza. Indipendentemente dal fatto che si decida di applicare l'algoritmo basato sui potenziali (che tiene conto delle sole distanze tra i punti) o che si applichi il CDF (che considera anche le velocità relative tra i corpi) per estensione si realizza un algoritmo nel quale le velocità di giunto dello spazio nullo, utilizzate per evitare gli ostacoli, tengono conto di M punti sul manipolatore e N sull'ostacolo.

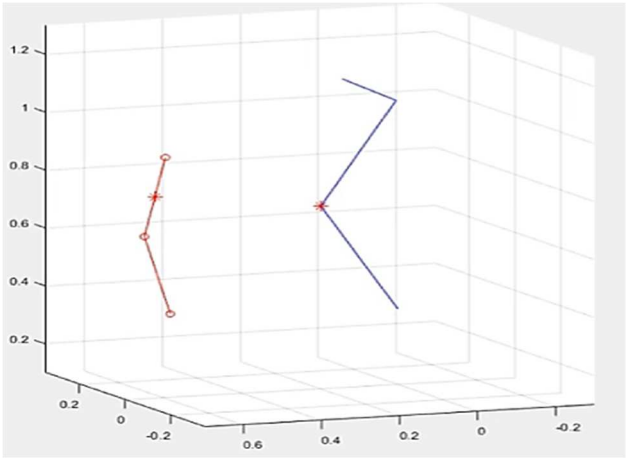


Figura 73 Rappresentazione, in fase di esecuzione dell'algoritmo di Collision Avoidance, del manipolatore e del braccio umano schematizzati mediante segmenti

Anche in questa fase, come fatto in precedenza, per garantire una rapidità di visualizzazione, senza però perdere le informazioni principali si può usare una rappresentazione schematica. In Figura 73 è riportato un istante temporale durante l'esecuzione di un algoritmo di anticollisione sul manipolatore per evitare un braccio umano in movimento (movimento acquisito in laboratorio mediante Kinect e successivamente filtrato). In blu, troviamo sempre il manipolatore, costituito dai suoi link, in rosso, invece, troviamo l'ostacolo che nel caso particolare è un braccio umano. Con la prova condotta si è dimostrata la validità dell'algoritmo di anticollisione anche con ostacoli più complessi dei semplici ostacoli puntiformi.

Come visto nei paragrafi precedenti anche per questa applicazione si può utilizzare l'ambiente di visualizzazione V-REP per implementare la traiettoria con **mantenimento posizione** in concomitanza con i seguenti **ostacoli: polso, gomito e spalla** come punti rappresentativi del braccio di una persona.

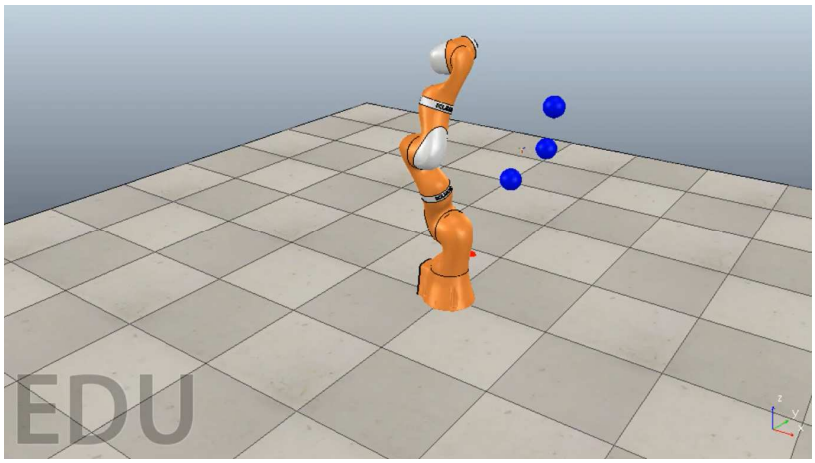


Figura 74 Un frame del robot durante il mantenimento della posizione nello spazio in presenza di un ostacolo costituiti da polso, gomito e spalla in movimentato in ambiente V-REP

In ogni caso, in questa implementazione si perde un aspetto cruciale dell'algoritmo di anticollisione ovvero la sua implementazione in real time, infatti, a causa del processamento a cui sono sottoposti i dati acquisiti il test è stato eseguito in un secondo tempo in seguito all'acquisizione della posizione dell'ostacolo. Inoltre, tale algoritmo presenta un notevole problema di fondo, ovvero, prendendo N punti sul manipolatore e M sul braccio umano la

complessità computazionale cresce rapidamente e conseguentemente i tempi di calcolo si allungano, ciononostante si è constatato che, anche prendendo semplicemente 5 punti sul braccio (spalla, gomito, polso e due punti posizione intermedia su braccio ed avambraccio) e i soliti punti caratteristici sul robot (6 giunti + 4 punti intermedi), l'algoritmo mantiene un'ottima efficacia.

4.6.2 I voxel e l'algoritmo GJK

Un ulteriore passo in avanti, per la modellazione del problema, può essere fatto mediante l'uso dei voxel. Un **voxel** è una regione spaziale cubica, pertanto è possibile immaginare un volume solido costituito da un certo numero di voxel così come un'immagine digitale è costituita da un certo numero di pixel.

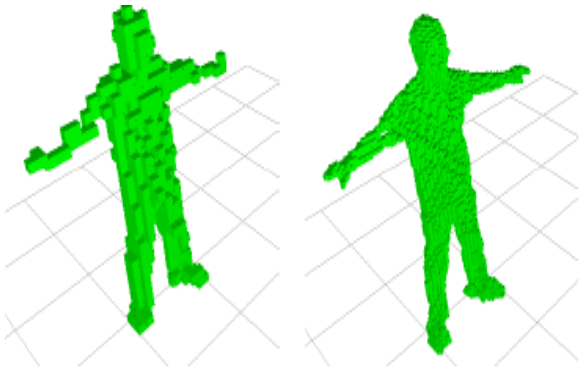


Figura 75 Rappresentazione di un corpo umano mediante voxel di differente dimensione

Tra l'altro questo approccio è quello che viene comunemente usato in computer graphic e nei più elaborati videogiochi. Una possibile estensione degli algoritmi di anticollisione sviluppati potrebbe essere quella di ricorrere alla **programmazione** in parallelo mediante **CUDA** (Computer Unified Device Architecture). CUDA è un'estensione della programmazione in linguaggio C che piuttosto che utilizzare la CPU del computer come Matlab fa ricorso alla GPU che esegue le medesime operazione ma in modo estremamente più rapido perché progettata per lavorare con processi in parallelo piuttosto che in serie (62). Non avendo però tali competenze informatiche si è deciso di non percorrere tale strada sebbene rimane una possibile soluzione per migliorare l'intero algoritmo implementato.

Ciononostante, esiste un altro modo per estendere l'algoritmo sviluppato usando figure geometriche tridimensionali che certamente approssimano meglio la reale geometria degli elementi in gioco. Infatti, conoscendo la posizione spaziale dei punti caratteristici del braccio umano, che si vuole assumere come ostacolo da evitare, e conoscendo la posizione del manipolatore è possibile schematizzare i due elementi con insiemi di parallelepipedi e successivamente utilizzare l'**algoritmo GJK** per il calcolo delle distanze tra due corpi convessi (63).

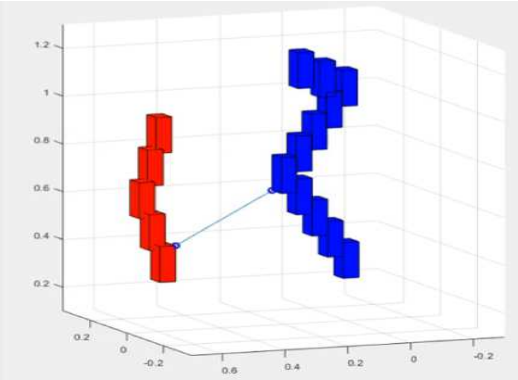


Figura 76 Rappresentazione, in fase di esecuzione dell'algoritmo di collision avoidance, del manipolatore e del braccio umano schematizzati mediante voxel

Proseguendo con lo stesso codice colore adottato in precedenza, nella Figura 76 è rappresentato l'ostacolo (braccio umano) in rosso e il manipolatore in blu. Mediante l'algoritmo GJK poi si può valutare la distanza minima tra questi volumi.

L'algoritmo di calcolo delle distanze GJK prende il nome dai suoi inventori: Gilbert – Johnson – Keerthi. Questo algoritmo utilizza la differenza di Minkowski per determinare la minima distanza tra due set volumi convessi in modo efficiente.

Questo metodo sicuramente approssima meglio i corpi rigidi in gioco ma presenta diversi aspetti negativi. In particolare, costruendo i voxel sui punti ricavati da Kinect, anche questo metodo come il precedente richiede l'applicazione del filtro di Kalman. Inoltre, anche in questo caso avendo N voxel per il robot e M voxel per il braccio è necessario applicare il GJK N x M volte con il conseguente appesantimento temporale e rallentamento in esecuzione.

4.6.3 Kinect Point Cloud

Se si vuole incrementare ancora di più l'accuratezza di rappresentazione degli ostacoli da evitare è necessario ricorrere ad una rappresentazione tridimensionale che segua adeguatamente le forme complesse degli oggetti in questione. Per quanto concerne il corpo umano, un utile espediente può essere quello di utilizzare la **nuvola di punti** restituita dal sensore di profondità Microsoft Kinect.

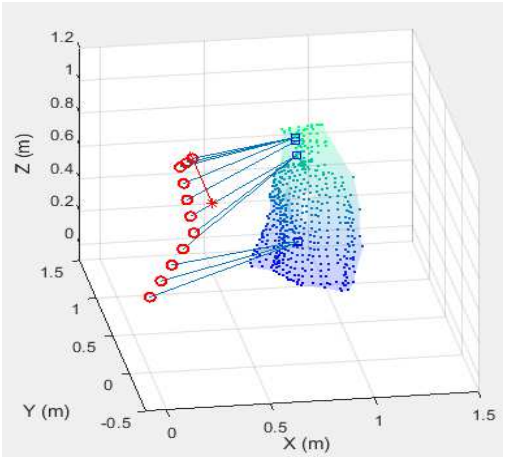


Figura 77 Rappresentazione, in fase di esecuzione dell'algoritmo di Collision Avoidance, del manipolatore e della point cloud estratta da Kinect

Una implementazione degli algoritmi di collision avoidance sviluppati può essere condotta come mostrato in Figura 77 in cui è possibile vedere con una serie di cerchi rossi i punti caratteristici del manipolatore robotico durante l'esecuzione di una traiettoria rettilinea. Dall'altro lato, invece, è possibile vedere una nuvola di punti indicativa di una persona. Infine, i segmenti blu rappresentano le minime distanze tra l'uomo e gli N punti presi per caratterizzare il robot.

Sebbene tale rappresentazione sia apparentemente semplice, al fine di applicare un algoritmo di anticollisione servendosi della point cloud estratta mediante Kinect, è necessario eseguire una serie di manipolazioni dei dati acquisiti. In particolare, la nuvola di punti è costituita da un gran numero di punti, pertanto, non è pensabile valutare la distanza da ciascuno di essi a ciascuno dei punti caratterizzanti il manipolatore. Per questo motivo è necessario realizzare una figura geometrica semplice che sia un inviluppo della point cloud e poi da tale figura calcolare la distanza con i punti caratteristici del robot.

5 CAPITOLO 5: PROVE ESEGUITE IN LABORATORIO CON UR3

I test condotti ed analizzati nei precedenti capitoli hanno utilizzato come manipolatore di riferimento il più volte citato KUKA LBR iiwa R820 in quanto è un manipolatore a 7 gradi di libertà e pertanto risulta essere ridondante nello spazio almeno una volta anche imponendo posizione e orientazione dell'EE. Questo manipolatore risulta essere versatile e pertanto è uno dei più utilizzati per applicazioni collaborative perché può essere usato tanto con un grado di ridondanza specificando posizione e orientazione dell'EE quanto con quattro gradi di libertà ridondanti se si specifica solo la posizione dell'EE.

Presso i laboratori del DIMES del Politecnico di Torino, il gruppo di ricerca, in collaborazione con il quale si svolge questo lavoro di tesi ha acquistato un robot collaborativo della Universal Robot: l'UR3. Pertanto, al fine di condurre delle sperimentazioni con un reale manipolatore in laboratorio e testare il corretto funzionamento degli algoritmi di anticollisione sviluppati, si modificano gli script Matlab usati e si realizza un modello cinematico del manipolatore UR3. A questo punto, si possono condurre i medesimi test in ambiente virtuale Matlab e V-REP ed infine, si possono svolgere delle prove con il manipolatore reale in laboratorio. Si ripercorre in questo capitolo questo iter di adattamento degli algoritmi di anticollisione al manipolatore UR3 e si descrivono le prove eseguite in laboratorio.

5.1 Realizzazione modello cinematico UR3

Così come visto per il manipolatore KUKA anche per il robot **UR3** è possibile realizzare un modello cinematico utilizzando i parametri di Denavit-Hartenberg e facendo riferimento alle informazioni geometriche e le specifiche tecniche direttamente riportate sul sito del produttore danese Universal Robot.

Per le prove condotte in laboratorio, avendo a disposizione un reale robot non è necessario sviluppare un modello dinamico del manipolatore, in quanto, le informazioni di coppia erogata ai giunti e le altre informazioni inerziali sono direttamente restituite dal manipolatore, qualora si desiderasse fare un'analisi approfondita della dinamica del robot. Pertanto, almeno in via preliminare, per applicare gli algoritmi di anticollisione sviluppati è sufficiente realizzare un modello puramente cinematico.

5.1.1 Informazioni caratteristiche UR3

Per costruire un modello di un manipolatore robotico in ambiente virtuale è necessario raccogliere una serie di dati geometrici.

Il manipolatore UR3 è un braccio robotico dotato di 6 giunti rotoidali e quindi (solitamente) non ridondante nello spazio se si fissano come indicazioni da soddisfare sia la posizione che l'orientazione dell'EE (sebbene, si ricordi, che la definizione di ridondanza dipende dallo specifico task assegnato al robot). Per ricavare i **dati geometrici** di tale manipolatore si è fatto riferimento al sito ufficiale del produttore.

Specification	
Payload	3 kg / 6.6 lbs
Reach	500 mm / 19.7 in
Degrees of freedom	6 rotating joints
Programming	Polyscope graphical user interface on 12 inch touchscreen with mounting

Figura 78 Caratteristiche principali manipolatore UR3

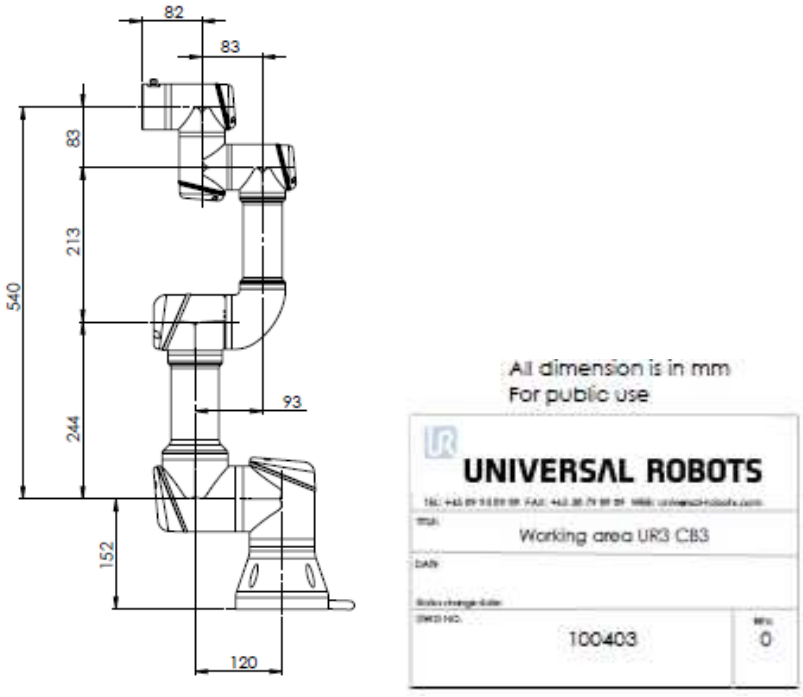


Figura 79 Dimensioni caratteristiche UR3

Attraverso questi dati geometrici è possibile ricavare tutte le matrici di trasformazione omogenea citate precedentemente applicando la convenzione di Denavit-Hartenberg.

Figura 80 Manipolatore UR3 rendering in V-REP

Dalla Figura 80 è possibile vedere i giunti caratteristici del manipolatore. Subito dopo la base che costituisce il giunto 1, vale la pena notare che il giunto 2 del manipolatore costituisce quella che comunemente è chiamata spalla, il giunto 3, invece, è detto gomito mentre i giunti 4, 5 e 6 costituiscono il polso del robot.

Oltre a tali informazioni geometriche per sviluppare un modello corretto del manipolatore analizzato occorre anche valutare quali siano i **limiti** di escursione angolare dei giunti del manipolatore e conseguentemente anche i limiti di velocità angolare massima raggiungibile.

Movement

Axis movement robot arm	Working range	Maximum speed
Base	± 360°	± 180°/Sec.
Shoulder	± 360°	± 180°/Sec.
Elbow	± 360°	± 180°/Sec.
Wrist 1	± 360°	± 360°/Sec.
Wrist 2	± 360°	± 360°/Sec.
Wrist 3	Infinite	± 360°/Sec.
Typical tool		1 m/Sec. / 39.4 in/Sec.

Tabella 5 Limiti angolari e di velocità manipolatore UR3

5.1.2 Trascrizione dei parametri di Denavit-Hartenberg

Come accennato nel paragrafo precedente, conoscendo quali siano le dimensioni caratteristiche del manipolatore di riferimento utilizzato, è possibile costruirne un modello matematico adottando la convezione di Denavit-Hartenberg modificata. Di seguito si riportano i quattro **parametri di Denavit-Hartenberg** per ciascun link del manipolatore.

Link	**i-1 rispetto i**	$\boldsymbol{\alpha_i}$ [°]	$\boldsymbol{a_i}$ $[\boldsymbol{m}]$	$\boldsymbol{d_i}[\boldsymbol{m}]$	$\boldsymbol{\theta_i = \theta_{i_0} + q_i}$ [°]
1	B-1	0	0	0.152	q_1
2	1-2	90	0	0	$-180 + q_2$
3	2-3	0	0.244	0	q_3
4	3-4	0	0.213	0.112	$-180 + q_4$
5	4-5	90	0	0.083	q_5
6	5-6	-90	0	0.082	q_6

Tabella 6 Parametri di Denavit-Hartenberg (convenzione modificata) UR3

Utilizzando la toolbox Matlab di Peter Corke per i robot è possibile riportare una rappresentazione schematica ma efficace del manipolatore descritto corredato dei sistemi di riferimento adottati di qui in avanti per la trattazione eseguita.

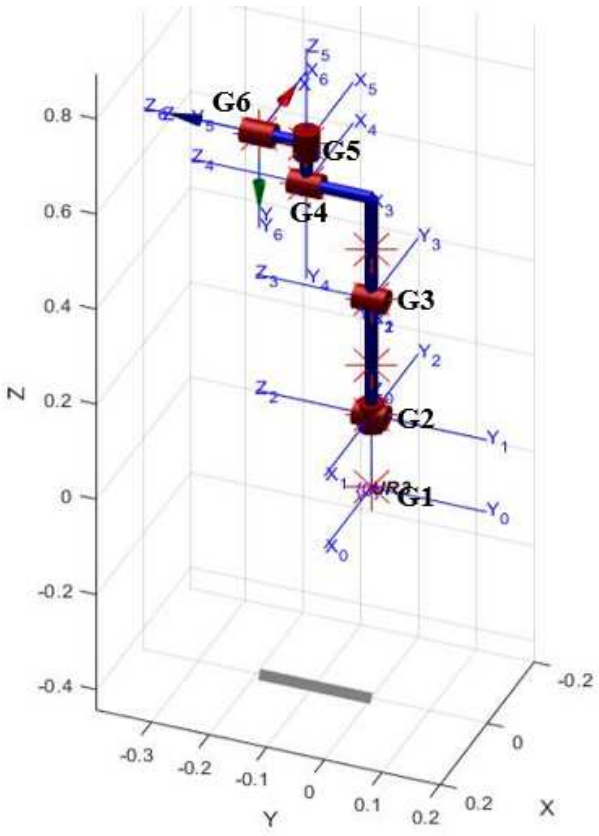


Figura 81 Sistemi di riferimento dei link utilizzati per robot UR3

Infine, dai parametri di Denavit-Hartenberg individuati è anche possibile ricavare le matrici di trasformazione omogenea necessarie per proseguire con la presente trattazione e con la successiva pianificazione della traiettoria del manipolatore.

5.2 Pianificazione della traiettoria

Per i test condotti nel corso del presente lavoro di tesi si prevedono due semplici traiettorie del manipolare. La prima consiste nel **mantenere** l'EE del manipolatore in una certa **posizione** spaziale ed intervenire mediante un task aggiuntivo usando lo spazio nullo per compensare i restanti 3 gradi di libertà ridondanti del manipolatore nello spazio. La seconda traiettoria pianificata è un **movimento lineare** dell'EE nello spazio operativo lungo uno specifico asse.

Si evita di descrivere quanto eseguito rimandando al paragrafo 2.4.

5.3 Prove eseguite in ambiente virtuale

Come fatto per il manipolatore KUKA anche per il robot della Universal Robot si possono preventivamente testare gli algoritmi di anticollisione basati sul campo di pericolo e sui potenziali repulsivi di De Luca in ambiente virtuale con più o meno semplici ostacoli esterni. Si segue il medesimo iter fatto per il KUKA, infatti, per entrambi gli algoritmi di anticollisione si valuta la relativa efficacia nei seguenti casi:

- Ostacolo puntiforme fisso;
- Ostacolo puntiforme mobile;
- Serie di punti presi sul braccio umano.

5.3.1 Esecuzione traiettoria rettilinea

Il primo step è valutare la correttezza della **traiettoria** implementata.

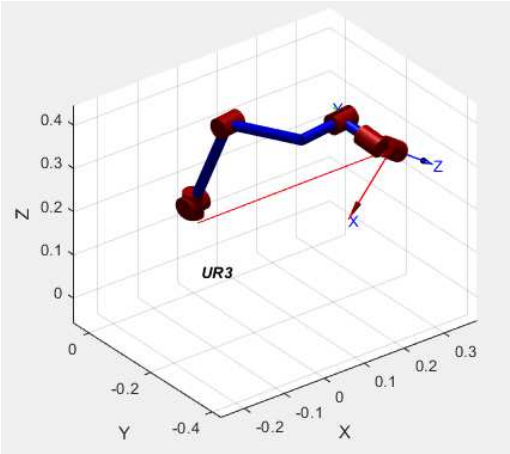


Figura 82 Esecuzione traiettoria rettilinea con manipolatore UR3

Avvalendosi dell'utilizzo del toolbox di Peter Corke è possibile verificare la corretta esecuzione della traiettoria implementata. Si ricordi che, per eseguire questa traiettoria e implementare in contemporanea un algoritmo di anticollisione avendo a disposizione diversi gradi di ridondanza, la traiettoria implementata è specificata indicando solo la posizione e non l'orientazione dell'EE. Inoltre, così come per il manipolatore KUKA, anche in questo caso, non si risolve la cinematica inversa del manipolatore in modo analitico ma ci si avvale del medesimo algoritmo CLIK (Closed Loop Inverse Kinematic) descritto nel paragrafo 1.5.1.

5.3.2 Ostacolo/ostacoli puntiformi fissi

In analogia con quanto fatto con il manipolatore KUKA, si testa il manipolatore con un **ostacolo puntiforme fisso** per analizzare la task suspension dovuta alla presenza di un ostacolo posizionato lungo la traiettoria rettilinea pianificata.

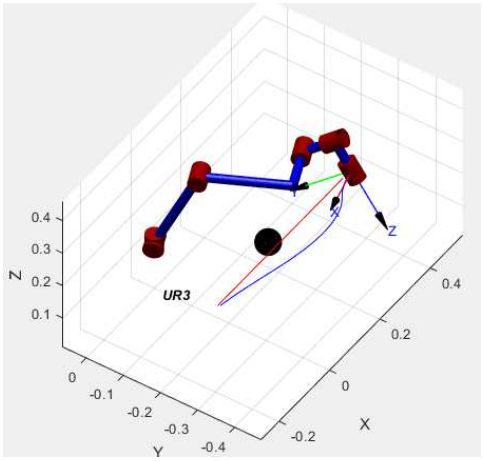


Figura 83 Esecuzione traiettoria rettilinea manipolatore UR3 e ostacolo puntiforme fisso

Si può constatare dal test condotto che il manipolatore abbandona la traiettoria pianificata per evitare urti con l'ostacolo. La Figura 83, come la precedente figura, riporta una prova eseguita con l'utilizzo dell'algoritmo di anticollisione basato sui potenziali di De Luca. Stessa prova si esegue con l'algoritmo del campo di pericolo riscontrando un risultato molto simile.

5.3.3 Ostacolo puntiforme comandato mediante mouse

Il medesimo metodo visto in precedenza con acquisizione della **posizione** dell'ostacolo **mediante mouse** in ambiente Matlab è utilizzabile anche per il manipolatore UR3.

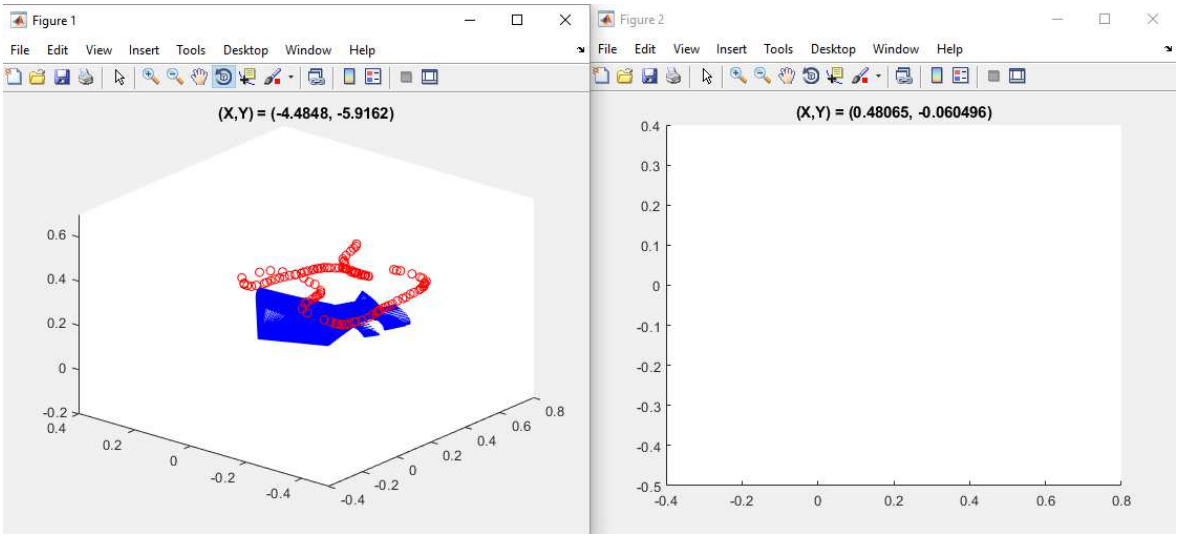


Figura 84 Esecuzione traiettoria rettilinea con manipolatore UR3 con ostacolo puntiforme mobile

In questo modo si può riscontrare l'efficacia dei due algoritmi sviluppati con una generica traiettoria non precedentemente stabilita degli ostacoli.

5.3.4 Prove in ambiente virtuale V-REP

Gli stessi test sono stati condotti in ambiente V-REP per una più efficace rappresentazione del modello robotico e dell'algoritmo realizzato, si riportano di seguito brevemente le prove eseguite con i relativi risultati.

Implementazione **traiettoria rettilinea senza ostacoli**.

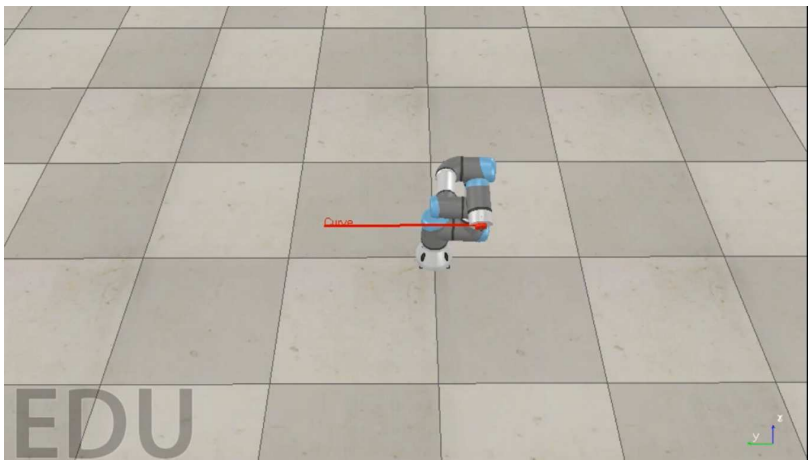


Figura 85 Un frame del robot durante l'esecuzione di traiettoria rettilinea senza ostacoli

Implementazione **traiettoria rettilinea con ostacolo puntiforme fisso**.

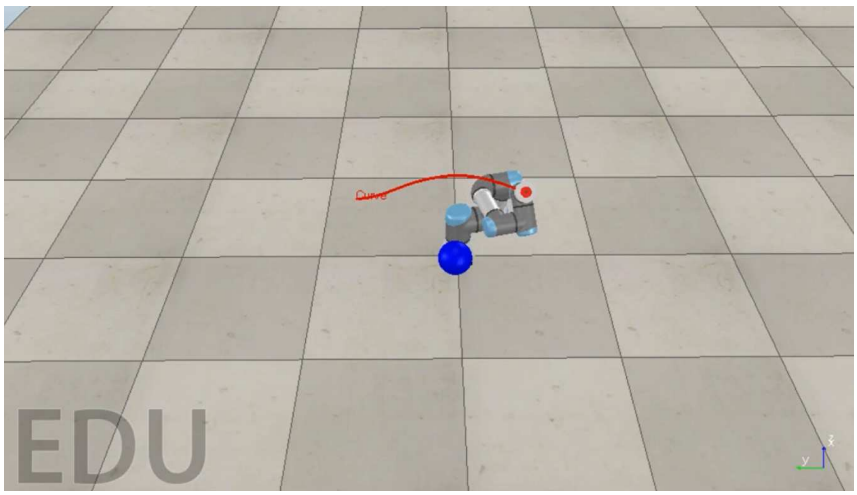


Figura 86 Un frame del robot durante l'esecuzione di traiettoria rettilinea in presenza di un ostacolo puntiforme fisso

Implementazione **traiettoria rettilinea con ostacolo puntiforme mobile**.

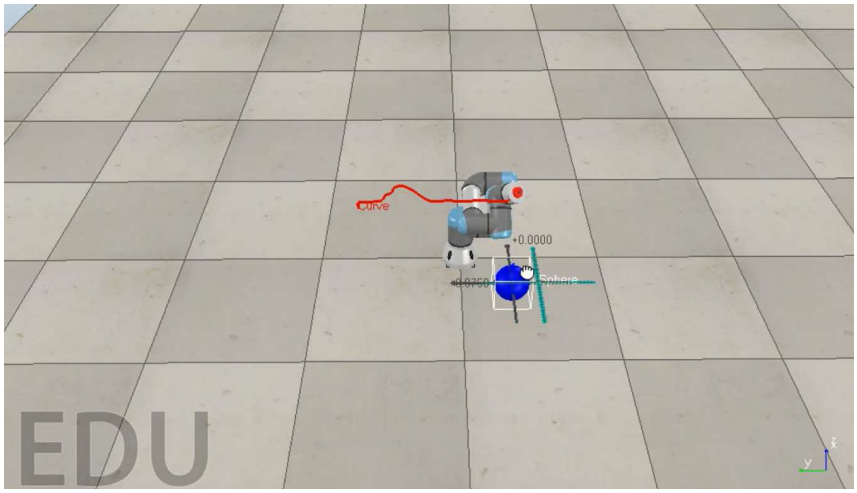


Figura 87 Un frame del robot durante l'esecuzione di traiettoria rettilinea in presenza di un ostacolo puntiforme movimentato in ambiente V-REP

Si può constatare anche in questo ambiente, utilizzato per la visualizzazione del manipolatore, l'efficacia degli algoritmi di anticollisione sviluppati.

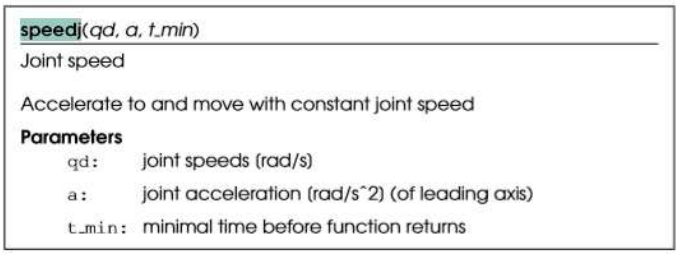
5.4 Prove eseguite in laboratorio con UR3

Dopo essersi assicurati di un corretto funzionamento di entrambi gli algoritmi di anticollisione in ambiente virtuale, con Matlab prima e con V-REP dopo, si è potuto passare al controllo del robot UR3 disponibile in laboratorio.

5.4.1 Set-up sperimentale

Il robot UR3 è pensato per un utilizzo user friendly mediante teach pendant da parte di operatori anche non molto specializzati ma presenta anche la possibilità di un controllo più avanzato. Tali aspetti di utilizzo del manipolatore nelle sue piene capacità sia in ambito previsto dal produttore mediante touch screen sia in modo avanzato sono stati oggetto di due tesi condotte in parallelo con il presente lavoro. Grazie ai progressi avanzati e le ricerche condotte si è compreso come

controllare il robot mediante comandi sotto forma di velocità di giunto attraverso la funzione **speedj**.



speedj(qd, a, t_min)

Joint speed

Accelerate to and move with constant joint speed

Parameters

qd:	joint speeds (rad/s)
a:	joint acceleration (rad/s^2) (of leading axis)
t_min:	minimal time before function returns

Figura 88 Descrizione funzione speedj dal manuale "The URScript Programming Language" di Universal Robot

Con la funzione speedj è possibile comandare il robot con le velocità di giunto passandogli anche l'accelerazione di giunto massima ed il tempo minimo prima che la funzioni ritorni. Inoltre, grazie ad una libreria appositamente realizzata per **comunicare** con il manipolatore **UR3**, inviando e ricevendo dati mediante **Matlab**, è possibile interfacciare il robot con un calcolatore. Questi due aspetti di invio di comandi ai giunti mediante velocità di giunto e ricezione della posizione e velocità effettiva dei giunti permette di controllare il manipolatore con gli algoritmi di anticollisione sviluppati. Infatti, come analizzato nel Capitolo 714, entrambi gli algoritmi di anticollisione sviluppati si basano sul controllo robot attraverso velocità di giunto ricavate mediante algoritmo di inversione cinematica CLIK, calcolo di distanze e/o velocità da uno o più ostacoli, aggiunta di un contributo di velocità di giunto appartenenti allo spazio nullo e conseguente calcolo delle posizioni angolari di ciascun giunto mediante integrazione numerica. In questa fase sperimentale, interfacciandosi con un robot reale e non con un modello virtuale, si calcolano allo stesso modo le velocità di giunto, ciò che cambia è che le posizioni dei giunti non sono ricavate numericamente come integrazione ma sono prelevate come feedback dal robot. Inoltre, altra differenza e che per utilizzare la funzione speedj è anche necessario calcolare la massima delle accelerazioni di giunto. Per tale scopo si può utilizzare una semplice derivazione numerica calcolando le velocità di giunto come differenza tra la velocità di giunto di set da dare come comando, la velocità di giunto attuale (anch'essa restituita in real time dal manipolatore) il tutto diviso l'intervallo temporale con cui si sta comandando il manipolatore.

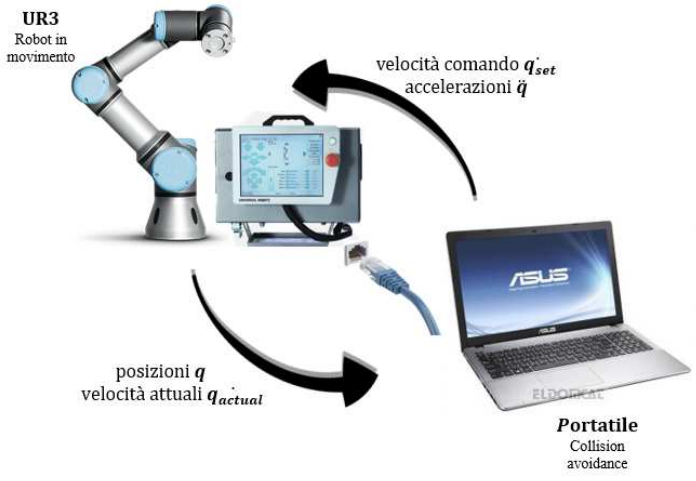


Figura 89 Schema di controllo del robot in laboratorio per applicazione algoritmi di anticollisione

In Figura 89 si riporta uno **schema** del **set up sperimentale** utilizzato e delle variabili in gioco nella comunicazione tra calcolatore e robot. Il computer è collegato all'indirizzo IP del robot mediante cavo ethernet che permette di ricevere ed inviare dati in real time. Le variabili in uscita dal calcolatore per il controllo del manipolatore sono le velocità di set dei giunti e le accelerazioni che costituiscono gli input della funzione speedj. Le variabili restituite dal manipolatore sono molteplici e vanno dalle coppie ai giunti alle variabili cinematiche attuali (ovvero reali) e stimate ma quelle che interessano in questa applicazione sono le posizioni e velocità reali del manipolatore.

5.4.2 Test 1: esecuzione traiettoria pianificata

Il primo test eseguito per constatare il corretto collegamento tra personal computer e robot è la semplice esecuzione di una traiettoria rettilinea, senza l'intervento di un algoritmo di anticollisione. Si riporta di seguito la traiettoria eseguita e quella pianificata per il movimento del manipolatore.

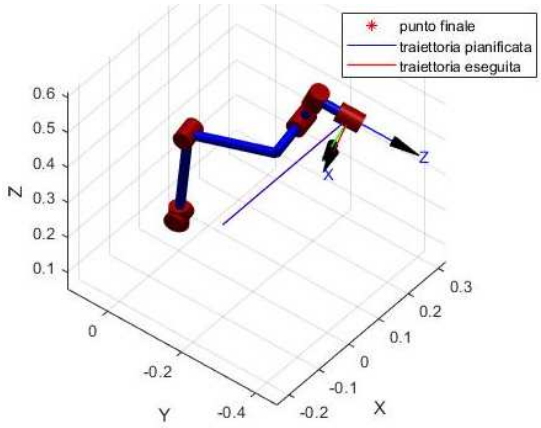


Figura 90 Confronto traiettoria pianificata e traiettoria eseguita dal manipolatore reale

Con questa rappresentazione si evince che il manipolatore reale segue in modo abbastanza accurato la traiettoria pianificata, per avere un migliore riscontro però è necessario applicare uno zoom.

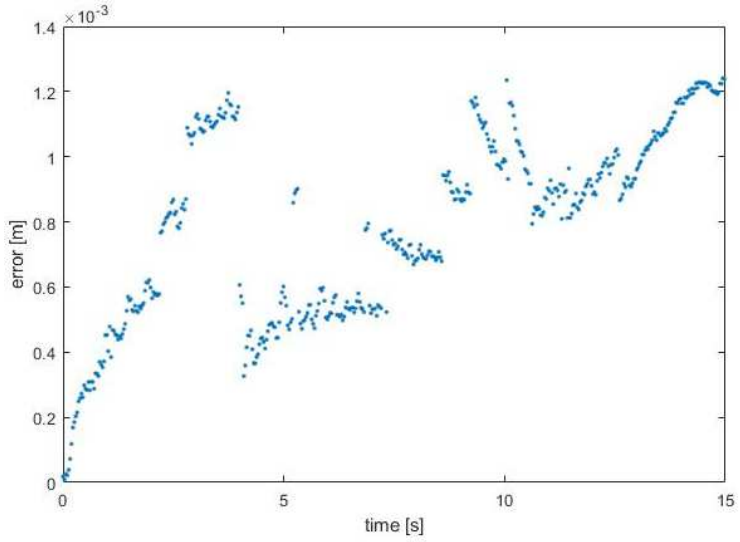


Figura 91 Errore tra traiettoria programmata e traiettoria eseguita dal robot

Si è constatato che il manipolatore ha una certa vibrazione durante l'esecuzione della traiettoria, ciononostante, si nota che l'**errore** sull'esecuzione della traiettoria è molto ridotto. Infatti, sull'asse verticale z la traiettoria è seguita fedelmente a meno di un errore di 0.4 millimetri, mentre sull'asse y si ha un errore di 1.5 millimetri. Complessivamente, la norma dell'errore tra traiettoria pianificata e traiettoria eseguita dal manipolatore è al più di 1.2 mm. Pertanto, questo test preliminare fa constatare un corretto controllo del manipolatore a meno di errori che possono essere associati a: calcolo, vibrazioni, attriti e altri fattori non facilmente rilevabili.

5.4.3 Test 2: esecuzione traiettoria con presenza ostacolo fisso

Constato un ottimale controllo del manipolatore su una semplice traiettoria implementata su Matlab nello spazio operativo ed un'efficace comunicazione tra l'unità di controllo del robot e il computer utilizzato si è verificato il funzionamento dell'algoritmo di anticollisione con il robot reale con la semplice prova riportante un **ostacolo fisso**.

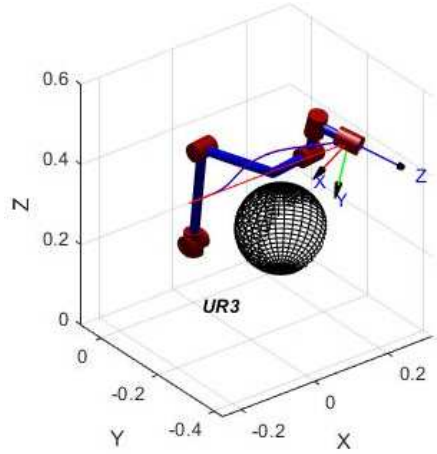


Figura 92 Confronto traiettoria anticollisione oggetto fermo in ambiente virtuale e reale

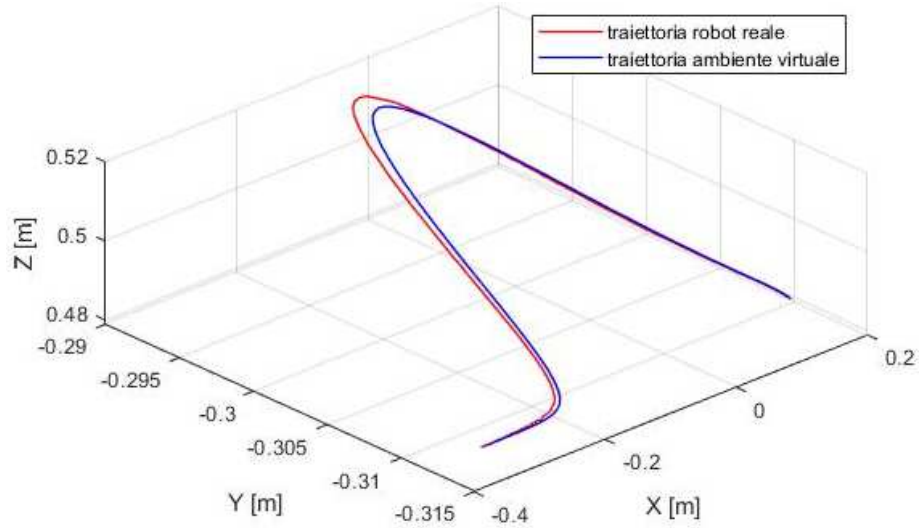


Figura 93 Confronto traiettoria anticollisione oggetto fermo in ambiente virtuale e reale zoom

Anche con questa prova è possibile verificare che l'ambiente virtuale realizzato riproduce in modo molto simile al reale la traiettoria del manipolatore. Infatti, anche qui, come nel caso precedente il massimo errore tra le due traiettorie è di appena 1,5 millimetri.

Questa prova ha anche avuto rilevante importanza per la definizione della **frequenza** con la quale è possibile **controllare il manipolatore**. Infatti, fissate le specifiche del computer utilizzato per il controllo, fissata la versione adottata dell'algoritmo di anticollisione si è reso possibile controllare il robot reale ad una frequenza di 31.25 Hz ovvero sono stati inviati dati di controllo al robot ogni 32 millisecondi.

5.4.4 Test 3: esecuzione traiettoria con ostacolo in movimento

Infine, si è condotta un'ultima prova con l'applicazione dell'algoritmo di anticollisione in presenza di un **oggetto sferico** liberamente **movimentato in V-REP**. Ovviamente in questa fase è anche possibile utilizzare un oggetto reale come ostacolo, acquisirne la posizione mediante un sistema di visione come Microsoft Kinect o OptiTrack e utilizzare la posizione acquisita dell'oggetto reale.

Anche in questo caso, come visto in precedenza, è possibile constatare un'ottima corrispondenza tra il comportamento del manipolatore modellato in ambiente virtuale Matlab e V-REP e la reale risposta data dal manipolatore nel test di laboratorio.

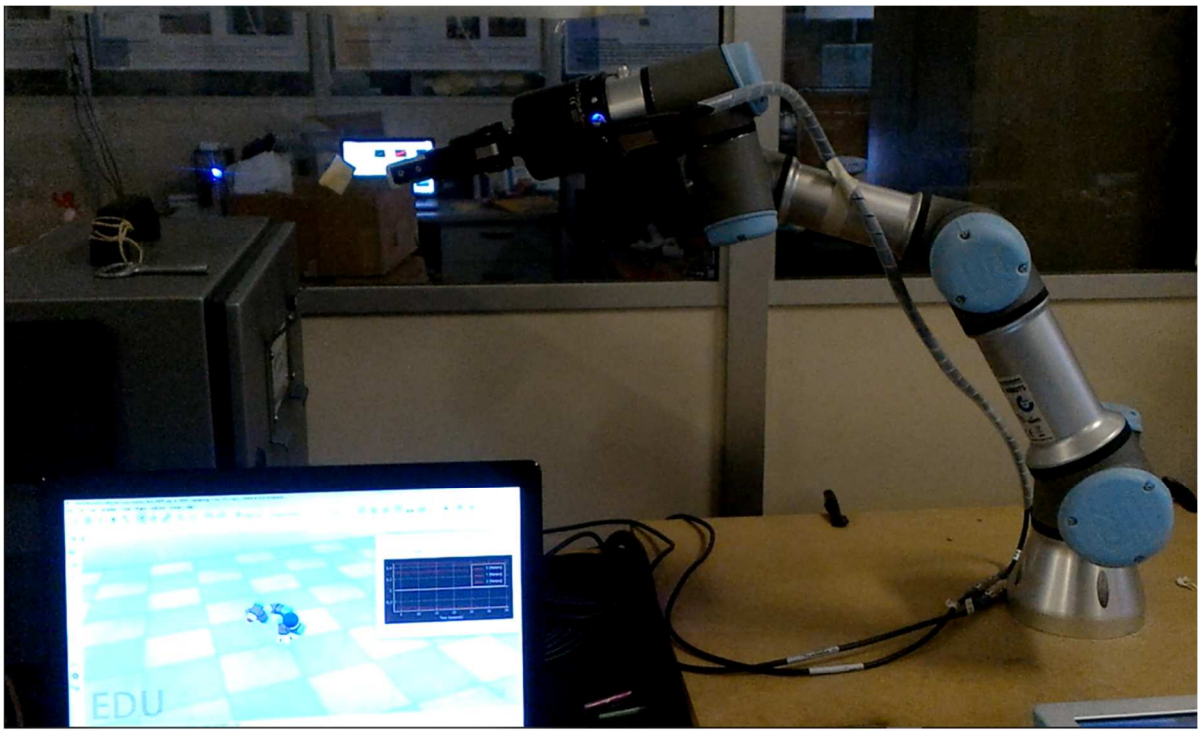


Figura 94 Prova eseguita in laboratorio con ostacolo liberamente movimentabile in V-REP

6 CAPITOLO 6: MIGLIORAMENTI E SVILUPPI ALGORITMI IMPLEMENTATI

6.1 Point cloud

Tra i vari approcci analizzati per l'estensione del problema di anticollisione da ostacoli puntiformi ad ostacoli rappresentativi del corpo umano è opportuno concentrare l'attenzione sulla nuvola di punti restituita dal sensore di profondità Microsoft Kinect. Infatti, relativamente a tale dispositivo ci sono librerie implementate per i più comuni ambienti di programmazione tra i quali anche Matlab. In particolare, per utilizzare la nuvola di punti che ben approssima la forma del corpo umano e la sua posizione spaziale, si possono valutare diversi approcci:

- Utilizzo di forme geometriche semplici (come sfere e cilindri) per realizzare un inviluppo delle parti del corpo più significative (braccio, avambraccio, testa);
- Utilizzo di una mesh, più o meno complessa che inviluppi tutti i punti del corpo umano.

Prima ancora di poter proseguire però con una di queste trattazioni è necessario fare delle osservazioni e un processamento sulla nuvola di punti direttamente acquisita dal Microsoft Kinect.

6.1.1 Operazioni di estrazione, downsamplig e denoise

La point cloud direttamente restituita da Kinect appartiene ad un oggetto indicato come mappa di profondità (variabile `depthMap`). Questa nuvola di punti indica le posizioni spaziali di tutti i punti nell'angolo di ripresa del sensore di profondità con una dimensione di 424×512 caratterizzante la presenza di ogni oggetto, che sia una persona o un qualsiasi oggetto inanimato. Pertanto, se si vuole estrarre i punti di questa nuvola indicativi della persona ed utilizzare tale figura per un algoritmo di anticollisione occorre:

1. **Estrarre** la sola porzione della **nuvola** di punti indicativa del corpo umano dall'intera point cloud, escludendo oggetti inanimati eventualmente presenti nella ripresa;
2. Effettuare una operazione di **sotto-campionamento** della porzione di nuvola estratta per ridurre le dimensioni di questa nuvola e velocizzare le successive operazioni di calcolo;
3. **Eliminare** eventuali **punti spuri** presenti nella porzione di nuvola estratta, frutto di un errore legato al rumore del sensore utilizzato.

La prima operazione è, senza dubbio, di grande importanza perché permette di concentrare l'attenzione solo sull'elemento rispetto al quale bisogna applicare, successivamente, l'algoritmo di anticollisione: la figura umana.

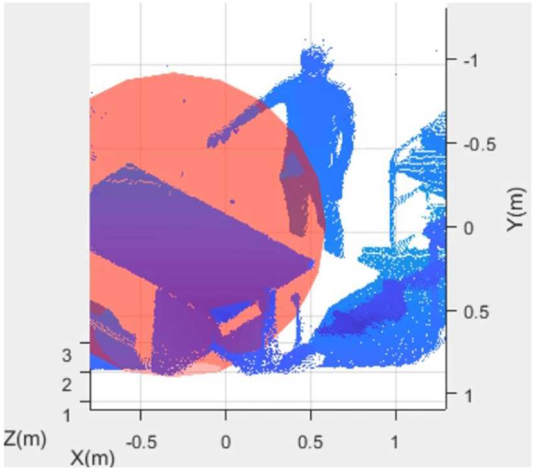


Figura 95 Nuvola di punti acquisita da Microsoft Kinect prima dell'estrazione della sola figura umana

Risulta evidente che questa operazione riduce notevolmente la mole di dati appartenenti alla nuvola di punti racchiudendo solo la quota parte di dati realmente necessari.

Altra operazione fondamentale da eseguire è anche quella di **sotto-campionamento** della nuvola di punti. Fissando la dimensione di un volume spaziale è possibile ridurre notevolmente la dimensione della nuvola di punti.

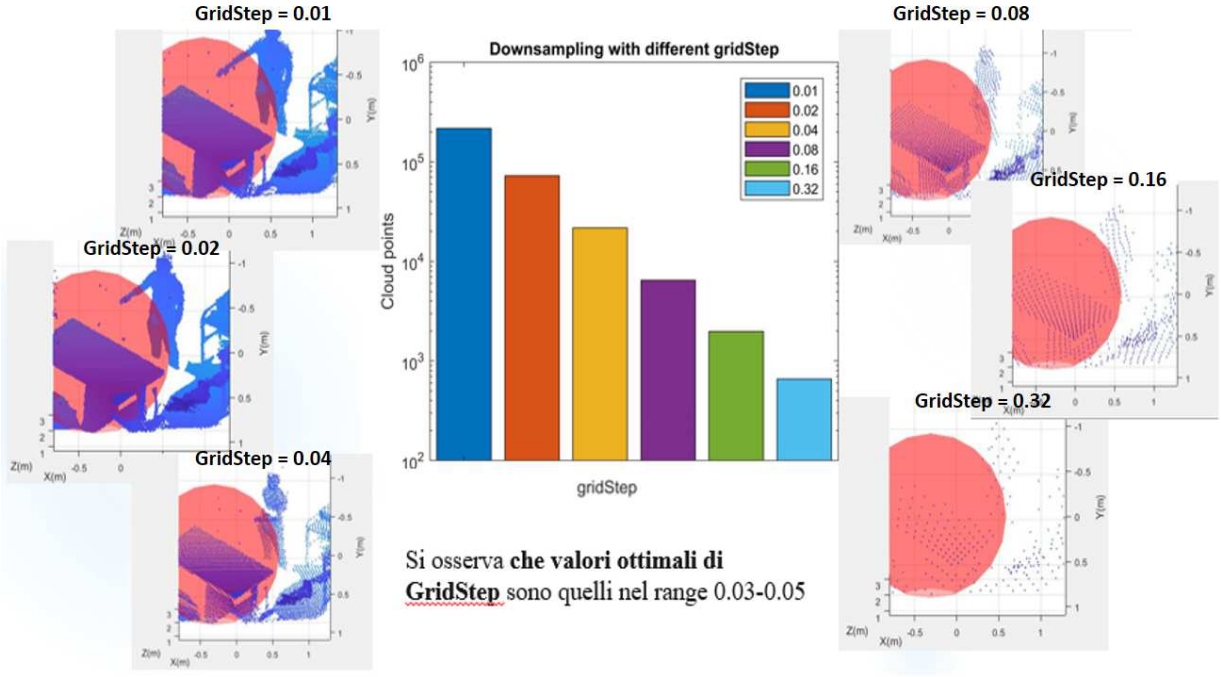


Figura 96 Effetto dell'operazione di downsamplig sul numero di punti dell'intera nuvola

Come si osserva dalla Figura 96 variando il parametro caratteristico dell'operazione di **downsamplig** (GridStep) è possibile ridurre notevolmente la mole di punti appartenenti alla nuvola. In particolare, si noti che l'istogramma riportato è in scala logaritmica quindi la semplice operazione di passare da una GridStep pari a 0.01 (barra blu e frame in alto a sinistra) a 0.04 (barra gialla e frame in basso a sinistra) causa una riduzione del numero di punti di un fattore 10. Si noti anche che con un GridStep pari a 0.01 sono acquisiti tutti i punti della nuvola che sono in numero pari a: $424 \text{ x } 512 = 2{,}17 \cdot 10^5$. Ovviamente, riducendo il numero di punti

si ha non solo un incremento della rapidità di calcolo delle successive fasi, bensì, si ha una riduzione dell'accuratezza con cui il corpo è ricostruito. Ciononostante, si può constatare che con valori caratteristici di questo parametro denominato GridStep tra 0.03 e 0.05 si ha un buon compromesso tra accuratezza di ricostruzione della figura e numero di punti complessivo.

Infine, la terza operazione necessaria prima di procedere con l'applicazione del reale algoritmo di anticollisione è effettuare un'operazione di **denoise**, questa è necessaria per eliminare tutti quei punti spuri presenti a causa del rumore e che potrebbero compromettere il funzionamento corretto dell'algoritmo di anticollisione. L'operazione di denoise, come l'operazione di estrazione della figura umana, è realizzabile mediante opportune funzioni implementate in Matlab per lavorare con la point cloud di Kinect.

6.1.2 Inviluppo point cloud con superfici elementari (cilindri sfere)

Per applicare un algoritmo di anticollisione, non è possibile utilizzare direttamente la nuvola di punti, in quanto, sarebbe computazionalmente troppo oneroso calcolare le distanze e velocità dei punti caratterizzanti il manipolare e di tutti i punti appartenenti alla nuvola. Per questo motivo, occorre usare delle figure che permettono di ridurre la successiva operazione di calcolo delle distanze e conseguente rilevamento dei potenziali repulsivi da applicare nello spazio nullo del manipolatore.

Una prima strada che può essere seguita in questo senso è la schematizzazione delle parti più significative del corpo umano (braccio, avambraccio e testa) con semplici figure solide convesse quali **cilindri** e **sfere**.

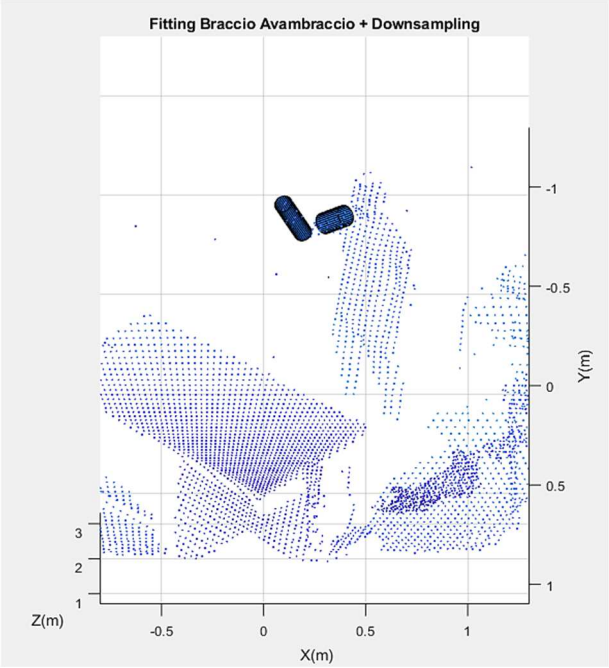


Figura 97 Operazione di fitting di due cilindri sul braccio ed avambraccio

Questa operazione è possibile in ambiente Matlab anche direttamente attraverso l'utilizzo delle funzioni `pcfitcylinder`, `pcfitplane`, `pcfitsphere` che, dotate di opportuni parametri di input, permettono di inviluppare i volumi più comuni con cilindri, sfere e piani.

6.1.3 Inviluppo mediante mesh 3D

Sebbene l'operazione di fitting con geometrie solide note abbia delle buone potenzialità in termini applicativi per un algoritmo di controllo dei manipolatori con anticollisione esiste un metodo più accurato e computazionalmente anche più efficiente per descrivere il volume di un corpo umano. Infatti, è possibile ricostruire la figura umana mediante una **mesh 3D** e successivamente, pensando il manipolatore come una successione di punti caratteristici, applicare il calcolo delle distanze punti-mesh. A questo punto, note le distanze, è possibile applicare gli algoritmi di anticollisione visti in precedenza.

In particolare, per proseguire con questa operazione si possono utilizzare due diverse funzioni `bonundary` e `convhull`. Tali funzioni permettono di circoscrivere la nuvola di punti con un volume delimitato da una mesh non convessa che segue meglio la reale forma del corpo umano (**boundary**) e con una più semplice mesh convessa che ripercorre la geometria umana meno fedelmente (**convhull**). Un'importante osservazione è che, in aggiunta agli input richiesti da convhull, boundary richiede anche un fattore che contraddistingue l'accuratezza con cui la mesh inviluppa la nuvola di punti detto shrink factor (variabile tra 0 e 1, più grande → approssimazione più precisa). Si riporta ora un confronto tra queste due funzioni in fase di esecuzione dell'algoritmo di anticollisione.

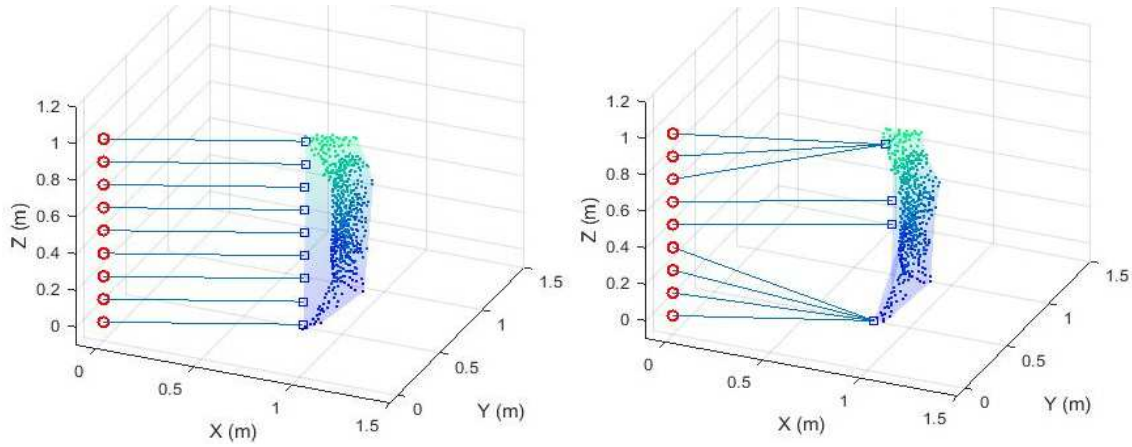


Figura 98 A sinistra è riportato l'algoritmo di Collision Avoidance in fase di esecuzione con convhull, a destra con boundary con shrink factor = 0.3

A questo punto, settando lo stesso GridStep a 0.04 (compromesso tra ridotto numero di punti e precisione di rappresentazione della figura umana) è possibile confrontare i tempi caratteristici di esecuzione usando le due funzioni boundary e convhull. Per tale operazione si osservi che tutte le valutazioni in termini di prestazioni e tempistiche ed i confronti eseguiti sono riferiti al personal computer usato per tutto il presente lavoro di tesi. In particolare, si riportino le specifiche tecniche del **computer** adottato per le simulazioni eseguite:

Specifiche dispositivo

Nome dispositivo	DESKTOP-OF3K967
Processore	Intel(R) Core(TM) i7-4500U CPU @ 1.80GHz 2.40 GHz
RAM installata	8,00 GB (7,89 GB utilizzabile)
ID dispositivo	B6997113-26B9-4932-A27E-B52696D4AE5C
ID prodotto	00326-10000-00000-AA534
Tipo sistema	Sistema operativo a 64 bit, processore basato su x64
Penna e tocco	Nessun input penna o tocco disponibile per questo schermo

Figura 99 Specifiche tecniche personal computer adottato

A questo punto si definiscono:

- t_extract: tempo di estrazione del dato ptCloudk dalla variabile depthMap fornita da Kinect;
- t_pctransform: tempo di trasformazione delle coordinate della point cloud nel sistema di riferimento world;
- t_pctuning: tempo per effettuare downsampling e denoise;
- t_mesh: tempo per costruire la mesh che inviluppa la point cloud;
- t_dist: tempo per il calcolo della distanza da una serie di punti significativi indicanti il robot e l'inviluppo creato.

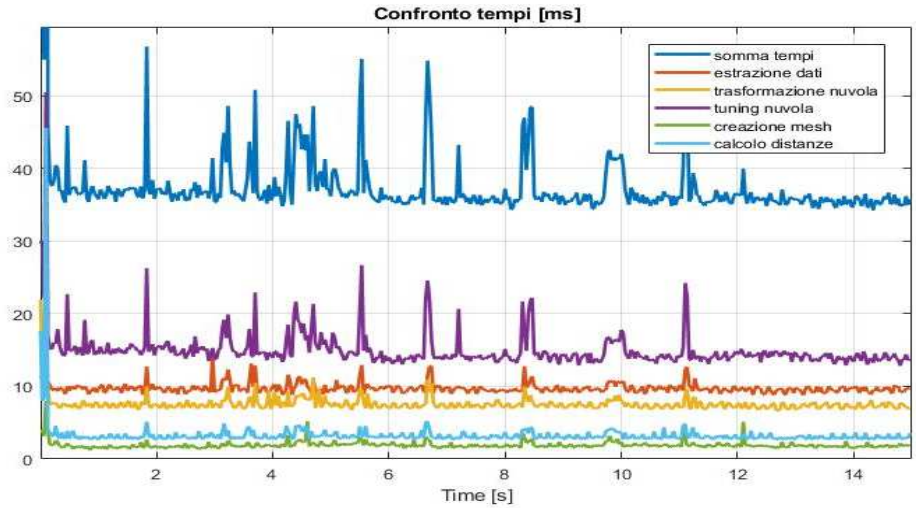


Figura 100 Tempi caratteristici convhull

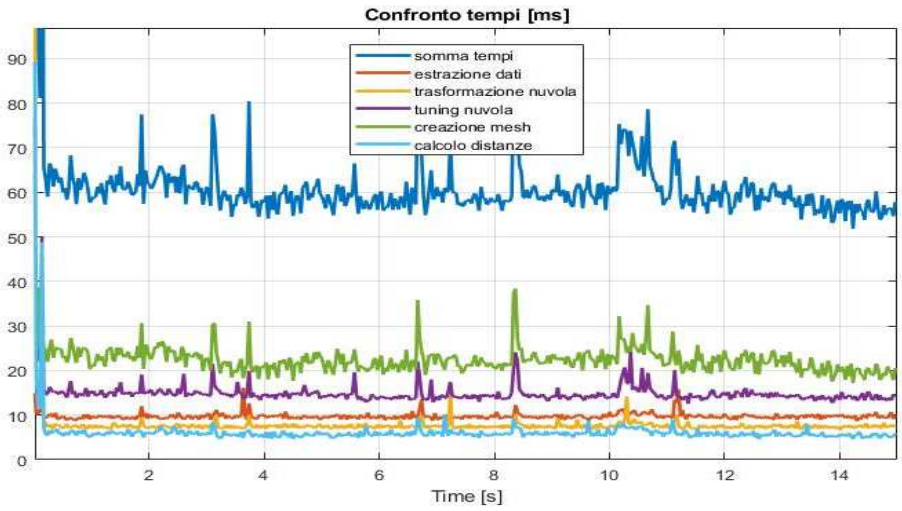


Figura 101 Tempi caratteristici boundary shrink factor = 1

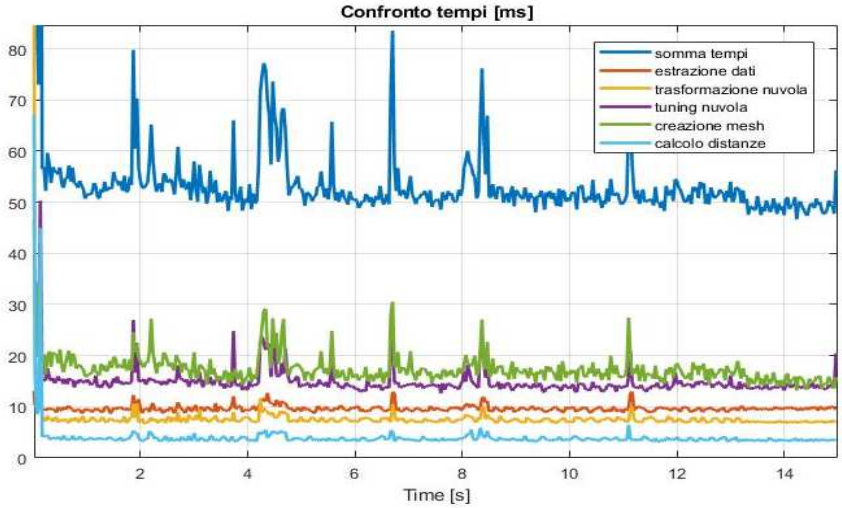


Figura 102 Tempi caratteristici boundary shrink factor = 0.3

Come è lecito aspettarsi con convhull si hanno tempi inferiori nel calcolo della mesh che inviluppa la nuvola di punti. Usando boundary si ha sicuramente una migliore approssimazione della reale forma umana ma le tempistiche di calcolo della mesh sono di molto superiori rispetto all'utilizzo di convhull. Ovviamente riducendo l'accuratezza di rappresentazione della mesh attraverso lo shrink factor si ha una sensibile riduzione del tempo necessario per il calcolo della mesh. Un compromesso tra tempi di calcolo e precisione di rappresentazione risulta usare boundary con un basso valore di shrink factor come ad esempio 0.3 o 0.7. Si noti, inoltre, che in questa prima prova si ha un valore medio di tempo complessivamente impiegato per l'esecuzione del software realizzato di $40 - 60$ ms queso implica una frequenza di aggiornamento di $17 - 25$ Hz. Infine, è possibile osservare che tra i vari tempi, quelli maggiormente incidenti, in ordine di importanza, sul tempo complessivo di esecuzione sono:

- tempo creazione mesh
- tempo per tuning nuvola
- tempo estrazione dati
- tempo trasformazione nuvola

Possibili **miglioramenti** per ridurre i tempi di alcune di queste operazioni sono:

- Tempo creazione mesh: adottare convhull piuttosto che boundary per la generazione della mesh oppure ridurre lo shrink factor quando si utilizza boundary, questo ovviamente inficia sulla fedeltà con cui viene ricostruita la figura umana ma si resta sempre in un margine di sicurezza perché la figura restituita risulta di maggiori dimensioni rispetto alla sola persona. Infine, incidente sul tempo di creazione della mesh è anche il numero di punti da cui è costituita la nuvola e pertanto la dimensione del GridStep con cui si effettua l'operazione di downsampling.
- Tempo impiegato per il tuning della nuvola: per velocizzare questo tempo è possibile incrementare la dimensione del GridStep.
- Estrazione dati: questa operazione è quella sul cui gli interventi di miglioramento sono più limitati.
- Trasformazione della nuvola: per una corretta visualizzazione dei dati acquisiti e per le successive operazioni di post-processamento a questo livello di implementazione sono stati trasferiti i punti dal sistema di riferimento Kinect al sistema di riferimento assoluto nel quale successivamente si pensa di pianificare la traiettoria del manipolatore. Il tempo necessario per eseguire tale operazione si riduce ovviamente riducendo il numero di punti della nuvola ma si può anche annullare, quasi completamente, eseguendo l'operazione inversa, ovvero, è sufficiente riportare i punti caratterizzanti il robot nel sistema di riferimento di Microsoft Kinect.

Ovviamente tali frequenze di calcolo possono anche essere tutte incrementate usando un manipolatore con una maggiore potenza di calcolo rispetto al portatile personale utilizzato ed avente un processore di intel core i7 di quarta generazione rispetto ai moderni processori di ottava generazione aventi più core e maggiore frequenza di calcolo.

6.2 Implementazione degli algoritmi di anticollisione utilizzando la point cloud

Dopo aver analizzato la fattibilità di utilizzare lo strumento della nuvola di punti restituita da Microsoft Kinect in termini di tempi necessari per l'esecuzione delle operazioni fondamentali si sono implementati i due **algoritmi di anticollisione** analizzati in concomitanza di tale strumento, con il fine di ricostruire meglio la figura umana e quindi rendere più sicuro ed affidabile l'intero algoritmo rispetto alla semplificazione dello stesso che si ha semplificando la figura umana con voxel tridimensionali di grandi dimensioni oppure mediante semplici segmenti e/o punti caratteristici.

L'operazione di implementazione dell'algoritmo mediante l'uso della nuvola di punti e della corrispondente mesh che la inviluppa è stata seguita step by step, in particolare: come primo passaggio sono stati scelti 9 dei punti caratteristici individuanti il robot (giunti 2-3-4-5-6-7 e punti intermedi dei link 2-3, 3-4 e 4-5) e in base ad una posa arbitrariamente scelta del manipolatore sono stati visualizzati nello stesso diagramma in cui sono riportati la nuvola di punti e la mesh. Successivamente, si è pianificata la traiettoria del manipolatore ed in particolare sono state pensate due possibili pianificazioni: traiettoria rettilinea dell'EE del manipolatore nello spazio operativo e mantenimento della posizione dell'EE del manipolatore in posizione

fissa. A questo punto, si noti, che avendo a disposizione i punti del manipolatore e la mesh che inviluppa la point cloud, è possibile adottare la funzione Matlab `point2trimesh` per calcolare la **distanza minima** di ciascuno dei 9 punti caratteristici del robot dalla mesh che inviluppa lo point cloud e le coordinate esprimenti la posizione spaziale dei punti di minima distanza della mesh dal manipolatore (sempre in numero pari a 9). Noti i vettori posizione dei punti caratteristici sul robot e sulla mesh è possibile, per integrazione numerica, valutare la velocità degli stessi e pertanto è possibile implementare, non solo l'algoritmo di anticollisione basato sulle sole distanze, ma anche, il più complesso algoritmo basato sul campo di pericolo che tiene conto anche della velocità relativa manipolatore-ostacolo.

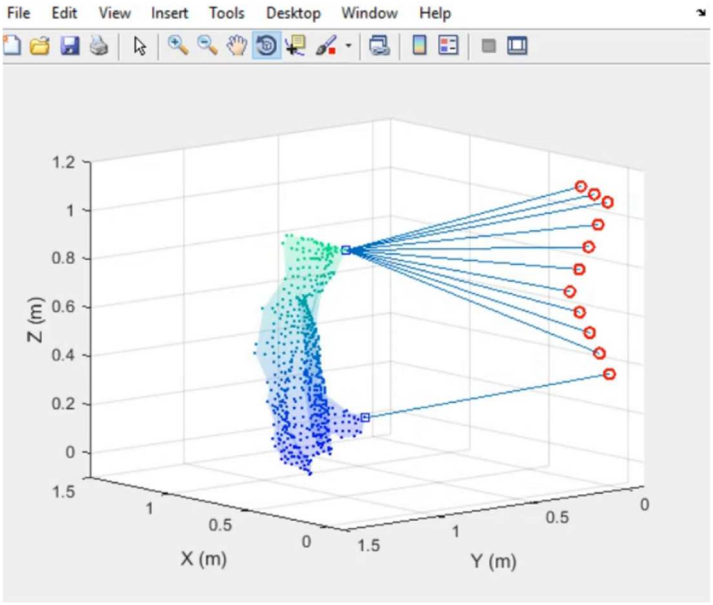


Figura 103 Un frame preso durante l'elaborazione dell'algoritmo di anti-collisione

Sebbene non sia possibile notarlo dal singolo frame preso durante la visualizzazione dell'efficacia dell'algoritmo di anticollisione si è notato che, a causa della continua variabilità della mesh costruita sulla nuvola di punti, le distanze calcolate tra la mesh ed il robot variano in modo non continuo come dovrebbero. Questo si ripercuote in modo ancora più evidente sulle velocità, in quanto, esse sono calcolate mediante rapporto incrementale. In conclusione, risulta che il manipolatore si allontana con eccessiva rapidità dall'ostacolo non avendo un funzionamento regolare.

6.3 Ottimizzazione algoritmo mediante Single Order IIR Filter

6.3.1 Single Order II Filter

A causa dei problemi citati in precedenza, è possibile ridurre il rumore di fondo, nella valutazione delle distanze utilizzando un filtro con funzionamento in real time. In particolare, è possibile adottare un **Single Order IIR Filter** tale da attenuare le componenti ad elevate frequenza nella valutazione della posizione dei punti più vicini al robot ma appartenenti alla mesh. A questo punto, vediamo prima come funziona il filtro adottato e poi come implementarlo numericamente in Matlab.

La funzione di trasferimento di un polo semplice si ricorda può essere scritta come:

$$G(s) = \frac{1}{1 + \tau s} \tag{4.7}$$

Un sistema caratterizzato da questa funzione di trasferimento presenta un'attenuazione di -3dB corrispondentemente ad un fenomeno caratterizzato da una frequenza $\frac{1}{\tau}$ il che si traduce in un'attenuazione allo 0.707 del valore stazionario.

Riportando il diagramma di Bode delle fasi di tale f**unzione di trasferimento** e ricordando che:

$$x\ dB = 20 \log_{10} x \tag{4.8}$$

È possibile comprendere come un filtro di questo tipo attenui le componenti in frequenza del segnale a cui è applicato a valori di frequenza superiori rispetto a quello in cui è posizionato il polo. Oltre all'attenuazione corrispondente a $\frac{1}{\tau}$ risulta anche interessante quella che si ha in corrispondenza di $\frac{10}{\tau}$. In corrispondenza di tale valore è possibile considerare l'andamento del diagramma di Bode delle ampiezze come asintotico con un asintoto obliquo a $-20\ \text{dB/decade}$. Pertanto, una decade dopo il posizionamento del polo del filtro in questione si ha un'attenuazione di circa -20 dB corrispondente ad un valore pari a circa un decimo del valore stazionario. In conclusione, un filtro di questo tipo è un filtro passa-basso che non attenua il modulo a basse frequenza ma che interviene, invece, alle frequenze più alte rispetto a quella caratteristica del polo selezionato.

Di seguito si riporta una rappresentazione schematica e più facilmente comprensibile del concetto che si vuole trasmettere con l'applicazione di un Single Order IIR Filter.

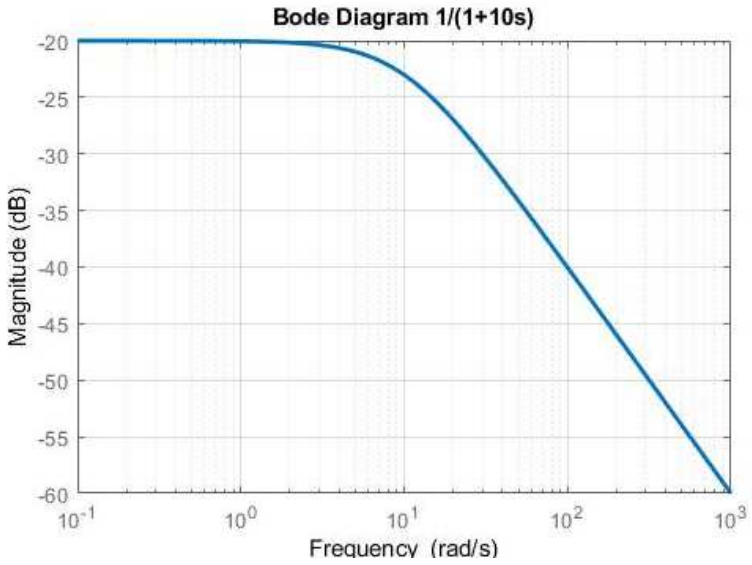


Figura 104 Diagramma di Bode delle ampiezze di $G(s) = \frac{1}{1+10s}$

ω/τ	|G(jω)| in dB	|G(jω)|
1	-3,0	0,707
5	-14,1	0,196
10	-20,0	0,100
50	-34,0	0,020
100	-40,0	0,010

Tabella 7 Tabella riassuntiva dell'attenuazione del filtro passa-basso sul modulo

In questo modo si cerca di eliminare il rumore relativo al segnale di posizione dei punti sulla mesh che sono più vicini al manipolatore e conseguentemente ridurre l'irregolarità di movimento sul manipolatore nell'uso dell'algoritmo di anti-collisione.

6.3.2 Implementazione del IIR Filter

L'**implementazione** del Single Order IIR Filter avviene secondo la seguente formulazione:

$$y(t) = b\,x(t) + b\,x(t-\Delta t) + a\,y(t-\Delta t) \tag{4.9}$$

Dove: $y(t)$ è il segnale filtrato;

$x(t)$ è il segnale non filtrato;

Δt intervallo di discretizzazione temporale del segnale digitale;

Ovviamente, implementando il filtro e l'intero algoritmo al calcolatore risulta tutto applicato ad un segnale digitale discreto nel tempo per effetto del campionamento temporale a 30 Hz dovuto alla frequenza di ripresa della videocamera inserita nel sensore Microsoft Kinect. Definendo:

- $\mathrm{Samplef} = 30\ \mathrm{Hz}$ la frequenza di acquisizione del segnale digitale da filtrare;
- Corner la frequenza di cut-off in corrispondenza della quale è inserito il polo della funzione di trasferimento del Single Order IIR Filter.

È possibile calcolare i coefficienti a e b necessari per l'implementazione del filtro real time mediante le seguenti relazioni:

$$\alpha = \tan(Corner/Samplef) \tag{4.10}$$

$$a = \frac{1-\alpha}{1+\alpha} \tag{4.11}$$

$$b = \frac{1-a}{2} \tag{4.12}$$

Pertanto, definendo la frequenza in corrispondenza della quale si desidera filtrare e conoscendo la frequenza di acquisizione dei dai è possibile passare dal segnale digitale acquisito dotato di rumore ad un segnale digitale filtrato.

6.3.3 Analisi effetto del IIR Filter

Per tarare il low pass filter implementato e osservare l'effetto che lo stesso ha sull'algoritmo di anticollisione è possibile imporre al manipolatore il mantenimento di una certa posizione spaziale dell'EE ed esaminare il comportamento in presenza di una persona in movimento, rilevata con Microsoft Kinect, modellizzata mediante una mesh fornita da boundary con shrink factor pari a 0.7 nel caso di anticollisione realizzata mediante CDF.

Si riportano ora, sovrapposti, i risultati ottenuti in assenza di un filtro e in presenza di due filtri passa basso ad 1 e 5 Hz.

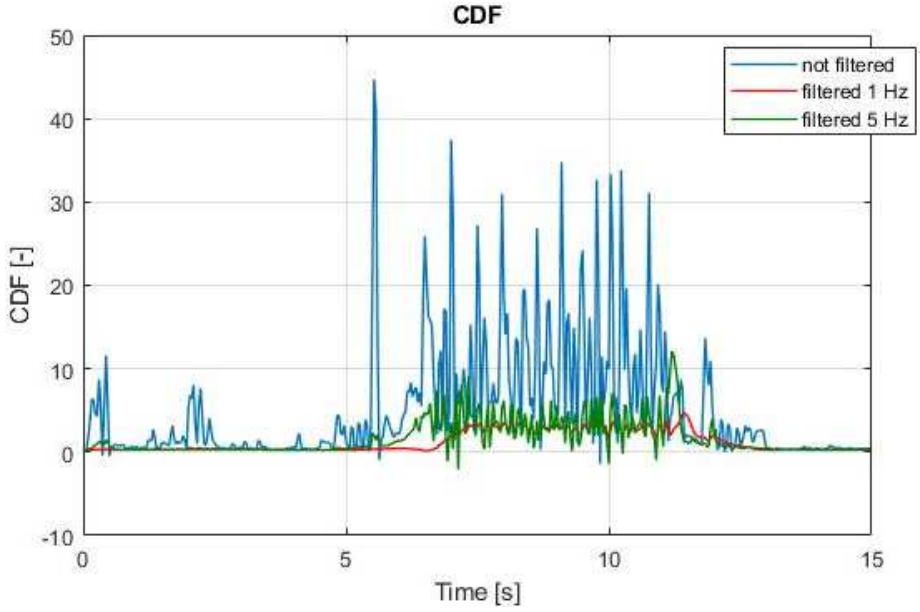


Figura 105 Confronto CDF calcolato al variare del tempo senza filtro, con un filtro a 1 Hz e con un filtro a 5 Hz

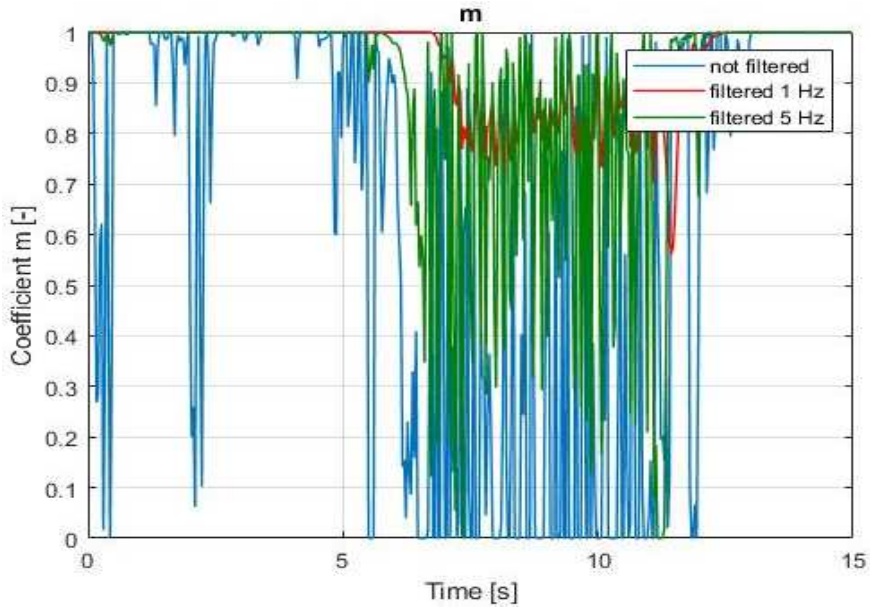


Figura 106 Coefficiente m caratteristico dell'algoritmo di anticollisione al variare del tempo senza filtro, con un filtro a 1 Hz e con un filtro a 5 Hz

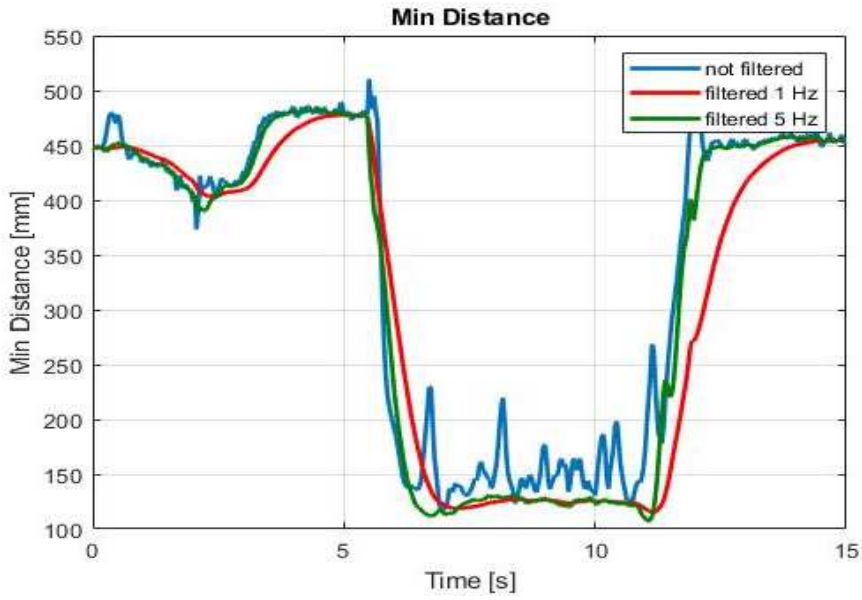


Figura 107 Minima distanza del manipolatore dall'operatore umano al variare del tempo senza filtro, con un filtro a 1 Hz e con un filtro a 5 Hz

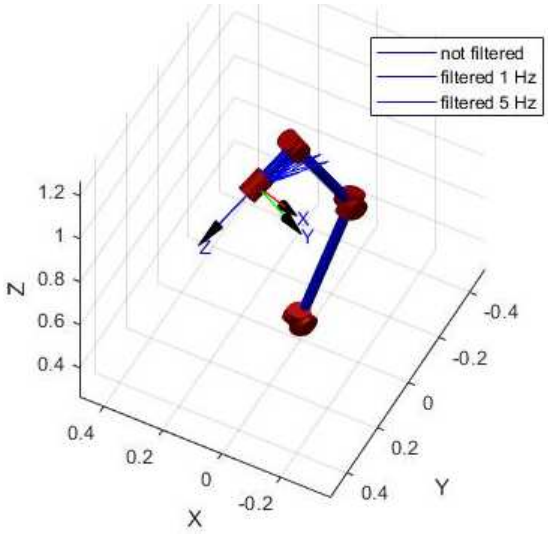


Figura 108 Confronto Percorso manipolatore al variare del tempo senza filtro, con un filtro a 1 Hz e con un filtro a 5 Hz

È possibile constatare l'effetto del filtro passa basso utilizzato, si osservi prima di tutto che i parametri caratteristici dell'algoritmo di anticollisione sono stati tarati in modo tale da permettere un avvicinamento dell'operatore umano a distanze non inferiori a $10 - 12$ cm.

Per quanto concerne sia il CDF, sia la distanza minima che il coefficiente m si può constatare che applicando un filtro che tagli a frequenze via via più bassa generi un **maggiore regolarità** nella variazione delle medesime quantità ma comporta anche un crescente **ritardo** di risposta del manipolatore per allontanarsi dalla figura rilevata come ostacolo e sorgente di pericolo. In particolare, ci si concentri sul parametro di minima distanza del manipolatore dalla mesh che inviluppa la nuvola di punti che rappresenta l'ostacolo da evitare. Questa minima distanza è la sorgente delle azioni repulsive applicate sul manipolatore per ovviare il contatto tra i due corpi. Si nota evidentemente che, a causa della continua variabilità della mesh generata consegue una grande rumorosità di tale distanza minima, stesso aspetto si osserva in modo ancora più evidente

nel CDF che considera anche la velocità relativa ostacolo-manipolatore. Applicando un filtro che attenua il rumore cui sono soggette le minimi distanze manipolatore-mesh si ha un netto miglioramento del comportamento del manipolatore. E l'algoritmo di anticollisione interviene esclusivamente in corrispondenza dell'avvicinamento dell'operatore al manipolatore stesso (il che si verifica tra i 6 ed i 12 secondi della prova che è stata processata e ripotata nei grafici precedenti).

Dalla prova esaminata si nota come un filtro passa-basso con frequenza di taglio pari a 5 Hz (che pertanto attenua del 30% i movimenti con periodicità di 0.2 secondi) è sufficiente per eliminare il rumore di fondo senza compromettere l'efficacia dell'algoritmo di anticollisione sviluppato. Infatti, il filtro a 5 Hz introduce un ritardo nella valutazione delle distanze al più di 0.05 secondi mentre il filtro più attenuante con frequenza di taglio ad 1 Hz genera un ritardo di diversi decimi di secondo che, in termini di sicurezza ed efficacia dell'algoritmo di collision avoidance sono valori eccessivi e non accettabili.

Infine, l'efficacia dell'applicazione del filtro passa-basso per eliminare il rumore si può constatare anche dal percorso compiuto dal manipolatore, in assenza del filtro si può constatare un allontanamento rapido e improvviso dell'EE del manipolatore dalla posizione che è stata impostata da mantenere, mentre, nelle altre due prove, l'EE praticamente non varia la sua posizione, mantenendo comunque una distanza media dall'ostacolo di oltre 10 cm semplicemente sfruttando i propri gradi di ridondanza. Questo è possibile notarlo dalle figure precedenti: Figura 107 e Figura 108.

7 CAPITOLO 7: CONCLUSIONI

Il presente lavoro di tesi ha cercato di analizzare uno dei più moderni ed attuali temi della robotica ovvero l'interazione dei robot con le persone, in modo sicuro e collaborativo.

Come si è analizzato, quello della robotica è un campo relativamente moderno ed in continua espansione, in particolare, la ricerca eseguita per questo lavoro di tesi si è concentrata sull'utilizzo di manipolatori cinematicamente ridondanti per l'esecuzione di task predeterminati in concomitanza con un subtask di collision avoidance.

Ripercorrendo il lavoro fatto, prima di tutto è stato necessario sviluppare una conoscenza di base sui temi della robotica e delle possibili applicazioni dei robot ridondanti. Successivamente, ci si è concentrati sull'argomento dell'anticollisione, ricercando approcci sviluppati negli anni sia in real time che offline. In seguito, compreso che ci fossero diverse tecniche e strade da percorrere per implementare algoritmi di anticollisione su manipolatori ridondanti ci si è concentrati su alcune particolari tecniche sviluppate negli scorsi anni: il metodo dei potenziali repulsivi di De Luca e la tecnica del Cumulative Danger Field di Rocco. A questo punto, avendo poche indicazioni concettuali su come questi autori abbiano sviluppato efficaci metodi di anticollisione si è ricercato di sviluppare dei propri algoritmi in ambiente di programmazione Matlab per implementare simili approcci per poi ottimizzarli step by step. A tal proposito è stato sviluppato un modello di manipolatore che si rifà al robot industriale KUKA LBR iiwa R820 utilizzando le convenzioni tipiche della robotica e le specifiche tecniche del robot industriale prese dal catalogo del produttore. In seguito, disponendo di un modello virtuale di uno dei più utilizzati robot in campo della robotica collaborativa si è passato all'implementazione di due algoritmi di anticollisione, uno basato sulle idee di potenziale repulsivo e l'altro sul concetto di campo di pericolo. Quest'ultimo, in aggiunta al primo, valuta non solo la distanza tra robot ed ostacoli esterni ma ne considera anche la velocità relativa come fattore incidente per il calcolo del livello di pericolo e delle conseguenti azioni sul manipolatore. Solo dopo aver sviluppato gli algoritmi nella loro forma più semplice con applicazione su ostacoli puntiformi (o sferici) fissi si è potuto avanzare e rendere più complessi gli algoritmi sviluppati. In particolare, conducendo test in ambiente virtuale è stato possibile adattare gli algoritmi di anticollisione alla presenza di ostacoli fissi e mobili, con traiettorie pre-implementate o definibili dall'utente in real time. Constando il corretto ed efficace funzionamento degli algoritmi si è reso possibile applicarli ad un robot industriale collaborativo presente in laboratorio. Per procedere in questo senso, è stato realizzato un nuovo modello cinematico del manipolatore industriale UR3 della Universal Robot e sono stati riadattati tutti gli algoritmi realizzati a questo differente robot. Solo dopo un'attenta ripetizione dei test in ambiente virtuali, sviluppati anche con il manipolatore KUKA LBR iiwa R820, è stato possibile condurre dei test con il reale robot con un opportuno set up sperimentale e verificare la corretta ed efficace implementazione degli algoritmi di anticollisione. A questo punto, constato il corretto funzionamento dell'anticollisione con figure semplici quali sfere si è cercato di introdurre la più complessa figura umana. A tale scopo sono state percorse numerose strade, la prima e la più semplice è quella di discretizzare il corpo umano mediante sfere ed utilizzare il

medesimo algoritmo visto in precedenza. Questo comporta però la necessità di un sistema di acquisizione della posizione umana ed a tale scopo è stato utilizzato il sensore di profondità Microsoft Kinect. Conoscendo la posizione dei principali giunti corporei restituiti da Kinect è possibile utilizzare l'algoritmo sviluppato prendendo più punti sul braccio umano. Ciononostante, si sono incontrati problemi nel procedere in questo modo, principalmente legati al rumore nell'acquisizione del segnale di posizione ed ai tempi di calcolo. A questo punto si è deciso di percorrere un'altra strada cercando metodi per ricostruire in modo più completo e veritiero la figura umana, a tale scopo sono stati condotti studi con il metodo di calcolo delle distanze tra figure convesse GJK e mediante algoritmi genetici. Infine, dopo diversi tentativi si è constatato che un'altra strada che utilizza sempre il sensore Microsoft Kinect fosse promettente. In particolare, oltre ai giunti corporei questo sensore di profondità restituisce anche una mappa del campo ripreso mediante nuvola di punti. Sebbene, per un'applicazione real time, è impossibile pensare di calcolare le distanze tra tutti i punti caratteristici del manipolatore e tutti i punti della point cloud acquisita si è adottato un'espediente per ridurre il peso computazionale, ovvero, una mesh tridimensionale che inviluppa tutti i punti della nuvola e permette il calcolo delle distanze e velocità tra manipolatore ed operatore umano in modo molto più fedele e rapido.

Sebbene dai primi test condotti alle ultime prove eseguite in ambiente virtuale siano stati fatti considerevoli progressi di ottimizzazione e aumento della complessità nel sistema di valutazione degli ostacoli presenti nel workspace del robot numerosi miglioramenti sono ancora possibili. In particolare, è opportuno indagare in modo più approfondito l'utilizzo della point cloud e tutti i problemi che ne conseguono e sono stati rilevati, come ad esempio: rumore nel segnale acquisito, riduzione dei tempi di calcolo, miglioramento della ricostruzione della figura umana mediante mesh 3D. Inoltre, oltre questa strada che è stata percorsa e presenta buone prospettive di sviluppo è anche opportuno valutare altre possibili approcci nella risoluzione del medesimo problema e effettuare un confronto non solo sull'efficacia ma anche sull'efficienza dei metodi adottati.

In particolare, come possibili sviluppi futuri, è possibile constatare che per l'eliminazione del rumore causato dall'acquisizione del Kinect è possibile adottare filtri più evoluti del semplice filtro passa basso del primo ordine adottato nella presente trattazione. Inoltre, un considerevole miglioramento potrebbe essere ottenuto utilizzando due sensori di profondità Kinect ed unire le due nuvole di punti rilevate separatamente. Ancora, oltre la strada che prevede l'utilizzo del Microsoft Kinect altri approcci che vale la pena approfondire sono l'utilizzo del linguaggio CUDA per sfruttare le potenzialità della CPU svolgendo calcoli in parallelo per incrementare la rapidità di esecuzione degli algoritmi, l'utilizzo di reti neurali o l'inserimento nella trattazione eseguita di un efficace ed efficiente filtro di Kalman.

BIBLIOGRAFIA

1. **Siciliano B., Sciavicco L., Villani L., Oriolo G.** *Robotica Modellistica, pianificazione e controllo.* Milano : McGraw-Hill, 2008.

2. *Safety of Industrial Robots: From Conventional to Collaborative Applications.* **Fryman J., Matthias B.** Munich : s.n., 2012.

3. **M., Hollerbach J.** *Dynamic Scaling of Manipulator Trajectories.* Cambridge : MIT Artificial Intelligence Laboratory, 1983.

4. *Minimum-Time Control of Robotic Manipulatrs with Geometric Path Constraints.* **Shin K. G., McKay N. D.** Ann Arbor : IEEE, 1985.

5. *Time-optimal control of robotic manipulators along specified paths.* **Bobrow J. E., Dubowsky S., Gibson J. S.** Irvine : Int. J. Robotics Research, 1985.

6. *Torque-limited path following by an-line trajectory time scaling.* **Dahl O., Nielsen L.** Lund : IEEE, 1994.

7. *A contribution to minimum-time task-space path-following problem for redundant manipulators.* **Basile F., Chiacchio P.** Salerno : Robotica, 2002.

8. *Minimum time path planning of robotic manipulator in drilling spot welding tasks.* **Zhang Q., Zhao M. Y.** Beijing : Journal of Computational Sesign and Engineering, 2016.

9. *Minimum Time Path-tracking Control of Redundant Manipulators.* **Ma S., Watanable M.** Ibaraki : IEEE, 2000.

10. *Motion Planning of a Redundant Manipulator for the Purpose of Speed-Up of the Hands Constant Speed Tasks.* **Okabe K., Aiyama Y.** Kobe : IEEE, 2013.

11. *Energy minimization of redundant manipulators using Generalized Pattern Search.* **A., Ata A.** Alexandria : Journal of Information and Optimization Sciences, 2007.

12. *Different-Level Simultaneous Minimization of Joint-Velocity and Joint-Torque for Redundant Robot Manipulators.* **Zhang Y., Guo D., Ma S.** Guangzhou : J intell Robot Syst, 2013.

13. *Joint Torque Reduction of a Three Dimensional Redundant Panar Manipulator.* **Yahaya S, Monghavvemi M., Almurib H. A. F.** Kuala Lumpur : sensors, 2012.

14. *Minimum-torque trajectory planning of redundant manipulators between two joint configurations.* **C., Wu.** Yunlin : Journal of the Chinese Institute of Engineers, 2011.

15. *Redundancy Resolution of Manipulators through Torque Optimization.* **Hollerbach J. M., Suh K. C.** Cambridge : IEEE, 1987.

16. *Redundant manipulator infinity-norm joint torque optimization with actuator constraints using a recurrent neural network.* **S., Tang W.** Hong Kong : IEEE, 2001.

17. *A singularity-free motion.* **Tan J., Xi N., Wang Y.** Houghton : Automatica, 2004.

18. *An on-line task modification method for singularity avoidance of robot manipulators.* **Qiu C., Cao Q. Miao S.** Shanghai : Robotica, 2008.

19. *Modified Jacobian Method of Transversal Passing through Singularities of Nonredundant Manipulators.* **Duleba I., Sasiadek J. Z.** Wroclaw : IEEE, 2002.

20. *A singularities Preservation Approach for Redundant Robot Manipulators.* **Mayorga R. V., Wong A. K. C.** Waterloo : IEEE, 1990.

21. *Singular inverse kinematic problem for robotic manipulators: A normal form approach.* **R., Tcho'n K. Muszy'nski.** Wroclaw : IEEE, 1998.

22. *Contour Tracking of a Redundant Robot Using Integral Variable Structure Control with Output Feedback.* **Lin C., Lee K.** Taipei : J Intell Robot Syst, 2011.

23. *Minimum Infinity-norm Joint Velocity Solutions for Singularity-robust Inverse Kinematics.* **Choi H., Lee S., Lee J.** Gwangju : International Journal of precision engineering and manifacturing, 2011.

24. *Singularity-Robust Task-Priority Redundancy Resolution for Real-Time Kinematic Control of Robot Manipulators.* **S., Chiaverini.** Napoli : IEEE, 1997.

25. *Torque Optimizing Control with Singularity-Robustness for Kinematically Redundant Robots.* **Chung C. Y., Lee B. H., Kim M. S., Lee C. W.** Seoul : Journal of Intelligent and Robotic Systems, 2000.

26. *Kinematic control of redundant robots with guaranteed joint limit avoidance.* **Atawnih A., Papageorigiou D., Doulgeri Z.** Thessaloniki : Robotics and Autonomous Systems, 2016.

27. *Real-Time Implementation of an Optimization Scheme for Seven-Degree-of-Freedom Redundant Manipulators.* **Dubey R. V., Euler J. A., Babcock S. M.** Knoxville : IEEE, 1991.

28. *A coefficient matrix GPM method to avoid joint limits for fault tolerant redundant manipulators.* **Beibei L., Shuli G. , Lina H.** Chengdu, : IEEE, 2016.

29. *A Weighted Least-Norm Solution Based Scheme for Avoiding Joint Limits for Redundant Joint Manipulators.* **Chan T. F., Dubey R. V.** Knoxville : IEEE, 1995.

30. *Avoiding Joint Limits and Obstacles for Kinematically Redundant Manipulators A Fuzzy Approach.* **Ahson S. I., Sharkey N. E., Nicolas B.** Riyadh : IEEE, 2002.

31. *Evolutionary indirect approach to solving trajectory planning problem for industrial robots operating in workspaces with obstacles.* **Abu-Dakka F. J., Rubio F., Valero F., Mata V.** Valencia : European Journal of Mechanics A/Solids, 2013.

32. *5-avoiding obstacles\Trajectory Planning of Redundant Robot Manipulators Using QPSO Algorithm.* **Guo J., Wang. X., Zheng X.** Zhengzhou : World Congress on Intelligent Control and Automation, 2010.

33. *Task-Oriented Redundancy Resolution and Obstacle Avoidance for Kinematically Redundant Robots.* **Cao B., Dodds G. I., Irwin G. W.** Belfast : Conference on Decision & Control, 1997.

34. *A Depth Space Approach to Human-Robot Collision Avoidance.* **Flacco F., Kroge T., De Luca A., Khatib O.** Roma : IEEE, 2010.

35. *A New Index of Serial-Link Manipulator Performance Combining Dynamic Manipulability and Manipulating Force Ellisoids.* **Kurazume R., Hasegawa T.** Fukuoka : IEEE, 2006.

36. *The Use of Kinematic Redundancy in Reducing.* **D., Walker I.** Rice : IEEE, 1990.

37. *Impact and Force Control.* **Youcef-Toumi K., Gutz D. A.** Scottsdale : IEEE, 1989.

38. *Modeling Impact Dynamics for Robotic Operations.* **Wang Y., Mason T.** Raleigh : IEEE, 1987.

39. *Mathematical Modeling of a Robot Collision with its Environment.* **Zheng Y. F., Gemami H.** Clemson : Journal of Robotic Syste, 1985.

40. *An Aaptive Force Control of Redundant Manipulator Based on Joint Acceleration Controller.* **Murakami T., Ohnishi K.** Yokohama : IEEE, 1991.

41. *Admittance control of a redundant industrial manipulator without using forcetorque sensors.* **Kaserer D., Gattringer H. Muller A.** Linz : IEEE, 2016.

42. *Base Reaction Optimization of Robotics Manipulstors for Space Applications.* **de Silva C. W., Chun C. L., Lawrence C.** Sydney : Proc. Int. Symp Robots, 1988.

43. *A Global Approach for Using Kinematic Redundancy to Minimize Base Reactions of Manipulators* . **Chung C. L., Desa S.** Pittsburgh : NASA, 1989.

44. *On the global optimum path planning for redundant space manipulators.* **Agrawal O. P., Xu Y.** Carbondale : IEEE, 1994.

45. *A Global Approach for Using Kinematic Redundancy to Minimize Base Reactions of Manipulators.* **Chung C. L., Desa S.** Carnegie : NASA, 1989.

46. *Redundancy problem in writing: from human to anthropomorphic robot arm.* **Potkonjak V., Popovic M., Lazarevic M., Sianovic J.** Belgrade : IEEE, 1998.

47. *A biomimetic approach to mobility distribution for a human-like redundant arm.* **Caggiano V., De Santis A., Siciliano B., Chianese A.** Pisa : IEEE, 2006.

48. —.**Caggiano V., De Santis A., Siciliano B., Chianese A.** Napoli : IEEE, 2006.

49. *A biomimetic approach to inverse kinematics for a redundant robot arm.* **Artemiadis P. K., Katsiaris P. t., Kyriakopoulos K. J.** Cambridge : Auton Robot, 2010.

50. *Bio-inspired kinematical control of redundant robotic manipulators.* **Shoushtari A. L., Mazzoleni S.** Volterra : Emerald Insight, 2016.

51. *Adapting Human Motion for the Control of a Humanoid Robot.* **Pollard N. S., Hodgins J. K., Riley M. J., Atkeson C. G.** Washington : IEEE, 2002.

52. *Biologically Inspired Optimal Robot Arm Control with Signal-Dependent Noise.* **Simmons G., Demiris Y.** Sandai : IEEE, 2004.

53. *Fault tolerant task execution through global trajectory planning* . **Paredis C.J.J., Khosla P.K.** Pittsburgh : Reliability Engineering & System Safety, 1996.

54. *Fault-tolerantmotionplanningandcontrolofredundantmanipulator.* **Li K., Zhang Y.** Guangzhou : Control EngineeringPractice, 2012.

55. *Fault Tolerance Force for Redundant Manipulators.* **Abdi H., Nahavandi S.** Waurn Ponds : IEEE, 2010.

56. *Fundamental Limitations on Designing Optimally Fault-Tolerant Redundant Manipulators.* **Roberts R. G., Yu H. G., Maciejewski A. A.** s.l. : IEEE, 2008.

57. *Self-motion utilization for reducing vibration of a structurally flexible redundant robot manipulator system.* **Kim S., Park Y.** Taejon : Robotica, 1998.

58. *Simultaneous Control for End-Point Motion and Vibration Suppression of a Space Robot Based on Simple Dynamic Model.* **Hirano D., Fujii Y., Abiko S., Lampariello R., Nagaoka K., Yoshida K.** Hong Kong : IEEE, 2014.

59. *Kinetostatic Danger Field - a Novel Safety Assessment for Human-Robot Interaction.* **Lacevic B., Rocco P.** Taipei : IEEE, 2010.

60. *Safety-Oriented Control of Robotic Manipulators - a Kinematic Approach.* **Lacevic B., Rocco P.** Milano : The International Federation of Automatic Control, 2011.

61. *A computationally efficient safety assessment for collaborative robotics applications.* **Polverini M. P., Zanchettin A. M., Rocco P.** Milano : Robotics and Computer–Integrated Manufacturing, 2017.

62. *Efficient Visual Hull Computation for Real-Time 3D Reconstruction using CUDA.* **Ladikos A., Benhimane S., Navab N.** Anchorage : IEEE, 2008.

63. *Enhancing G JK: Computing Minimum and Penetration Distances between Convex Polyhedra.* **S., Cameron.** Albuquerque, : IEEE, 1997.

65. *Stabilization constraint method for torque optimization of a redundant manipulator.* **Shim I., Yoon Y.** Taejon : IEEE, 1997.

66. *Stabilized minimum infinity-norm torque solution for redundant manipulators.* **Shim I., Yoon Y.** Taejon : Cambridge University Press, 1998.

67. *Development and Experimental Evaluation of a Robust Contact Force Control Strategy for a 7-DOF Redundant Manipulator.* **Shadpeyf I., Ranjbaranff I., Patelf R.V., Robins A. J.** Montreal : ICAR, 1997.