UNIVERSITÀ
POLITECNICA
DELLE MARCHE

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA MECCANICA

Algoritmi per la pianificazione del moto e la prevenzione di collisioni per robot antropomorfi

Algorithms for motion planning and collision avoidance for anthropomorphic robots

Relatore:
Prof. Giacomo Palmieri

Candidato:
Paramvir Singh

A.A. 2020/2021

Indice

Sommario **6**

1 Introduzione alla robotica **10**

1.1 La robotica 10

1.2 Applicazioni industriali 11

1.2.1 Struttura meccanica dei robot 12

1.2.2 Struttura dei manipolatori 14

1.3 Robot collaborativi 17

1.3.1 Misure di sicurezza per i cobot 18

1.3.2 Esempio di robot collaborativo 21

2 Cinematica **23**

2.1 Posizione ed orientamento di un corpo rigido nello spazio 23

2.1.1 Posizione 24

2.1.2 Orientamento 24

2.1.3 Rotazioni elementari 26

2.2 Composizione di matrici di rotazione 28

2.3 Angoli di Eulero 28

2.3.1 Angoli ZYZ o Roll-Pitch-Roll 29

2.4 Roto-traslazione 31

2.5 Trasformazioni omogenee 31

3 Cinematica differenziale **32**

3.1 Jacobiano geometrico 32

3.1.1 Derivata di una matrice di rotazione 33

3.2 Singolarità cinematiche 34

3.3 Problema cinematico inverso 35

4 Pianificazione moto e prevenzione di collisioni **37**

4.1 Damped Least-Squares 37

4.1.1 Definizione del fattore di smorzamento 38

4.2 Correzione del feedback 38

4.3 Strategia per evitare le collisioni 39

4.4 Pianificazione della traiettoria 42

5 Algoritmi **44**

5.1 Pianificazione del moto 44
5.2 Cinematica inversa del manipolatore 48
5.3 Cinematica inversa in presenza di ostacoli 50
5.4 Calcolo della traiettoria 54
5.5 Generazione ostacoli 63
5.6 Determinazione della cinematica simbolica 64
5.7 Calcolo dei parametri di guadagno 77
5.8 Conversione delle matrici di rotazione in angoli di Eulero 78
5.9 Calcolo della velocità attrattiva 79
5.10 Calcolo della velocità repulsiva 80
5.11 Plot delle sfere 80
5.12 Moto 81
5.13 Plot robot 84

6 Risultati e simulazioni **87**

7 Conclusioni **94**

Elenco delle figure

1 Struttura meccanica di un robot antropomorfo 13
2 Manipolatore cartesiano e suo spazio di lavoro 14
3 Manipolatore cilindrico e suo spazio di lavoro 15
4 Manipolatori sferici e loro spazio di lavoro 16
5 Manipolatore antropomorfo e suo soazio di lavoro 16
6 Robot SCARA e suo spazio di lavoro 17
7 Robot collaborativo utilizzato in campo medico: *ROBERT, Life Science Robotics, Aalborg (DK)* 18
8 Stop di sicurezza monitorato 19
9 Esempio di guida manuale per cobot (UR5), *Universal Robots* 20
10 Schema della regolazione della velocità in base alla distanza dall'operatore . 20
11 Esempio di spazio di lavoro condiviso uomo-robot, *Universal Robots* 21
12 Robot UR5 e suo schema cinematico 22
13 Posizione di un punto rispetto una terna di rifermento 24
14 Moto di pura traslazione tra due terne cartesiane 25
15 Rotazione relativa tra due terne 25
16 Rotazione della terna $O - xyz$ intorno all'asse z 27
17 Rappresentazione degli angoli ZYZ 29
18 Moto di traslazione e rotazione tra due terne 31
19 Componenti della velocità per l'end effector e il punto di controllo più vicino del robot 40
20 Parametri a_h e a_v in funzione della distanza d_0 42
21 Campi potenziali per la pianificazione della traiettoria 43
22 Schema delle funzioni implementate su MATLAB 44
23 Schema ostacolo presente sulla traiettoria del manipolatore 60
24 Direzioni di moto in funzione del valore di m 61
25 Schema cinematico del robot 76
26 Punti di controllo presenti sul robot 85
27 Zone rilevabili sensoristicamente dal robot 86
28 Modello CAD del robot con tutti i punti di controllo 87
29 Simulazione moto senza ostacoli 88
30 Simulazione moto con un ostacolo immobile 89
31 Simulazione moto con un ostacolo immobile 91

32 Simulazione moto con un ostacolo mobile 92
33 Simulazione moto con un due ostacoli mobili 93

Elenco delle tabelle

1 Dimensioni dell'UR5 23

Sommario

Negli ultimi anni, una rapida crescita tecnologica associata ad esigenze industriali di realizzazione di prodotti in intervalli di tempo sempre più contenuti con bassi costi di produzione, ha portato all'utilizzo di robot negli stabilimenti industriali. Queste macchine sono in grado, non solo di sostituire l'uomo nelle attività lavorative faticose, ripetitive o, in generale, poco gradite, ma anche di collaborare con l'operatore in attività specifiche dove l'intelligenza umana non è ancora rimpiazzabile da quella artificiale.
La seguente tesi ha come oggetto di studio proprio i robot collaborativi (denominati *cobot*) che tipicamente lavorano a stretto contatto con l'uomo. Dunque, per salvaguardare l'incolumità dell'operatore umano, risulta fondamentale che tali macchine siano dotate di appositi sistemi di sicurezza, i quali devono essere utilizzati con un certo livello di intelligenza.
L'obiettivo finale proposto per questa tesi è di generare degli algoritmi (basandosi sull'articolo *Real Time Strategy for Obstacle Avoidance in Redundant Manipulatros* [2]) per fare in modo che il robot, durante l'esecuzione della propria attività, qualora rilevi la presenza di un operatore umano (oppure di un qualsiasi oggetto fisico in generale), elabori tempestivamente una nuova traiettoria in modo da evitare l'urto con l'oggetto e riuscire a completare il proprio movimento senza doversi arrestare, come avviene solitamente in questi casi per i cobot.
Il seguente trattato è suddiviso in sette capitoli, il primo dei quali contiene una breve introduzione generale al settore della robotica, con argomenti trattati che vanno dall'evoluzione storica di questa disciplina alle applicazioni moderne in campo industriale. Inoltre si introduce il lettore alla componentistica dei robot, alle tipologie di *manipolatori* (cioè il braccio robotico senza il dispositivo terminale) ed infine viene trattato il ramo della robotica collaborativa, focalizzando sugli aspetti legati alla sicurezza.
Il secondo capitolo tratta la cinematica dei corpi rigidi nello spazio. Si parte dalla posizione di un punto espresso rispetto a una terna di riferimento, passando poi all'orientamento di una terna (può essere considerata solidale a un corpo rigido) rispetto a una terna fissa. In seguito si introduce una notazione compatta (*trasformazioni omogenee*) che contiene contemporaneamente informazioni sulla posizione e l'orientamento relativo tra due terne.
Il terzo capitolo tratta la cinematica differenziale, ovvero il calcolo dello spostamento e della velocità dei giunti del robot necessari per muovere il dispositivo terminale lungo la traiettoria desiderata con la velocità prestabilita.
Nel quarto capitolo è riportata la strategia per movimentare l'organo terminale del robot dalla posizione iniziale a quella finale, evitando eventuali ostacoli che a cui il robot andrebbe incontro durante tale moto.

Nel quinto capitolo sono riportati tutti gli algoritmi implementati su MATLAB, applicativi della teoria presente nei capitoli precedenti e a ciascuna di queste funzioni segue una breve descrizione della stessa.
Nel sesto capitolo sono riportate delle simulazioni su MATLAB degli algoritmi scritti, eseguite prendendo come riferimento l'UR5 della Universal Robots. Si tratta di un robot collaborativo con sei gradi di libertà nello spazio, dotato di un sistema sensoristico in grado di rilevare eventuali oggetti in prossimità del robot.
Infine è presente una trattazione conclusiva sul tema affrontato in questa tesi e i risultati ottenuti mediante le simulazioni sul software.

Abstract

In recent years, rapid technological growth associated with industrial realization needs of products in ever shorter time intervals with low production costs, has led to the use of robots in industrial plants. These machines are capable, not only of replacing the man in the activities that may result laborious, repetitive or, generaly, undesired, but can also collaborate with the operator in specific activities where human intelligence is not yet replaceable by machines.

The following thesis focuses on collaborative robots (called *cobots*) which typically work closely with humans. Therefore, to ensure the safety of the human operator, it is essential that these machines are equipped with appropriate security systems, which must be used with a certain level of intelligence.

The final goal proposed for this thesis is to generate algorithms (based on the article *Real Time Strategy for Obstacle Avoidance in Redundant Manipulatros* [2]) to make sure that the robot, during the execution of it's activity, if it detects the presence of an human operator (or any physical object in general), process promptly a new trajectory in order to avoid the collision with the object and be able to complete it's movement without having to stop, as is usually the case for cobots.

The following thesis is divided into seven chapters, the first of which contains a brief general introduction to the robotics industry, with topics covered ranging from the historical evolution of this discipline to modern applications in the industrial field. Furthermore, the reader is introduced to the components of the robots, to the types of manipulators (ie the robotic arm without the end effector) and the branch of the collaborative robotics, focusing on safety aspects.

The second chapter deals with the kinematics of rigid bodies in space. It starts from the position of a point expressed with respect to a reference frame, then passing to the orientation of a frame (can be considered integral with a rigid body) with respect to a fixed frame. Then the reader is introduced to a compact notation (homogeneous transformations) which contains simultaneously information on the position and relative orientation between two frames.

The third chapter deals with differential kinematics, ie the calculation of position and velocity of the robot joints needed to move the end effector along the desired trajectory with the predetermined speed.

In the fourth chapter is reported the strategy for moving the robot's end effector from the initial position to the final position, avoiding any obstacles that the robot would encounter

during such motion.
In the fifth chapter are reported all the algorithms implemented on MATLAB, based on the application of the theory present in the previous chapters and each of these functions is followed by a short description.
In the sixth chapter are reported some simulations on MATLAB of the written algorithms, performed using Universal Robot's UR5 as a reference. It is a collaborative robot with six degrees of freedom in space, equipped with a sensor system capable of detecting any objects near the robot.
In the end, there is a final discussion on the theme dealt within this thesis and the results obtained through software simulations.

1 Introduzione alla robotica

In questo capitolo iniziale della tesi si vuole introdurre il lettore alla *robotica*, una scienza multidisciplinare che include elementi di meccanica, elettronica e informatica. Verranno affrontate le problematiche relative all'introduzione dei robot nelle industrie e le normative (internazionali e/o europee) che tutelano la sicurezza dei lavoratori da tali macchine negli stabilimenti industriali.

1.1 La robotica

In letteratura esistono diversi definizioni di **robot**, ma tutte convergono nella descrizione di macchine *intelligenti*, capaci di sostituire l'uomo nello svolgimento di compiti difficili, ripetitivi o, in generale, poco graditi. Nonostante l'ampio utilizzo delle strutture robotiche nelle attività produttive sia un fenomeno relativamente recente, il concetto teorico di macchine artificiali in grado di compiere in totale autonomia i compiti tradizionalmente svolti dall'uomo è risalente a diversi secoli fa.
Nel tredicesimo secolo, il filosofo e scienziato inglese Ruggero Bacone scrisse la storia fittizia di due frati francescani che, per difendere il territorio inglese dagli invasori, idearono una muraglia d'ottone che corresse lungo i confini. Per realizzare l'enorme impresa, i frati progettarono prima una *testa meccanica* in grado di spiegare come costruire il muro.
Nel sedicesimo secolo, l'alchimista Philippus Theophrastus Von Hohenheim introdusse la figura dell' *Homunculus*, una forma di vita leggendaria creata per mezzo dell'alchimia.
Nel 1817, la scrittrice inglese Mary Shelley attraverso il romanzo *Frankenstein*, introdusse nella letteratura moderna, il concetto di uomo artificiale, ripreso poi numerose volte nella letteratura e cinematografia contemporanea.
Il termine Robot appare per la prima volta nel 1920 nel dramma *R.U.R.: Rossums Universal Robots* dello scrittore ceco Karel Capek. Il termine è coniato dallo scrittore derivandolo dalla parola ceca **robota** (letteralmente schiavitù, lavoro forzato). Nella finzione, i robot, costituiti interamente da materia organica e simili agli esseri umani, vengono costruiti per liberare l'uomo dalla schiavitù del lavoro fisico.
Negli anni '40, un noto scrittore di fantascienza russo, Isaac Asimov, concepì il robot come una macchina di sembianze umanoidi, incapace di provare sentimenti, ma dotato di un cervello artificiale programmato dall'uomo in modo da soddisfare determinate regole di condotta etica. Il termine *robotica* fu così introdotto da Asimov per indicare la scienza devota allo studio dei robot che si fondava sullo studio di tre leggi fondamentali:

1. Un robot non può recar 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 da un essere umano, purché tali ordini non contravvengano alla Prima Legge.

3. Un robot deve proteggere la propria esistenza, purché questa autodifesa non contrasti con la Prima e con la Seconda Legge.

Queste regole corrispondono a leggi da tenere bene in considerazione durante la progettazione di un robot.
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 da vari organi di movimentazione dei corpi rigidi che costituiscono il robot stesso, un **sistema di attuazione** (idraulica, elettrica, pneumatica, ecc.) che permette, da un punto di vista energetico, il movimento degli organi meccanici, i **sensori** (ottici, di contatto, piezoelettrici, piezoresistivi ecc.) che permettono di interfacciare il dispositivo con l'ambiente circostante ed infine di una unità di governo che riceve in ingresso un comando dall'utente e dati dai sensori, e permette di controllare gli attuatori che movimentano le parti meccaniche del robot come se costituisse il cervello della macchina.
Il settore della robotica è oggi grande soggetto di interesse industriale per numerosi motivi, uno dei quali risiede nella necessità di ricorrere ad automi per l'indisponibilità dell'operatore umano a svolgere determinate mansioni oppure per la sicurezza dello stesso.

1.2 Applicazioni industriali

La robotica applicata in campo industriale ha riscontrato un notevole successo per via dei grandi vantaggi che si traggono dall'automazione della produzione, come la riduzione dei costi complessivi di produzione, l'incremento della produttività, possibilità di miglioramento della qualità del prodotto realizzato, e la riduzione del rischio per l'operatore umano nello svolgimento di compiti pericolosi.
Con il *processo di automazione* si intende la sostituzione del lavoro manuale svolto tradizionalmente dall'uomo, dalle macchine in grado di svolgere il medesimo compito con un certo livello di indipendenza dall'operatore umano (in base all'autonomia di tali macchine si stabilisce la tipologia di automazione adottata). Tipicamente le macchine coinvolte in questo processo sono manovrate da sistemi di controllo intelligenti, capaci di gestire una grande mole di dati in tempo reale. Esistono tre tipologie di automazione adottate in campo industriale, ovvero

quella rigida, quella programmabile e quella flessibile.
L'*automazione rigida* riguarda la produzione di massa, ovvero il contesto industriale volto alla produzione di una grande quantità di prodotti identici tra loro. Per soddisfare questa richiesta di grandi volumi di produzione di oggetti pressoché uguali tra loro, sono richieste sequenze prefissate di operazioni di lavorazione/assemblaggio e le macchine coinvolte in questa tipologia di automazione sono *dedicate* alla particolare tipologia di prodotti realizzati, ovvero non si ha molta varietà di prodotti realizzabili con tali macchinari.
L'*automazione programmabile* riguarda il modello di industria dove viene richiesta la produzione di lotti di piccole e medi dimensioni di prodotti che potrebbero essere anche molto diversi tra di loro. Perciò, le macchine saranno di scopo generico, ovvero capaci di lavorare oggetti con differenze sostanziali l'uno rispetto all'altro. Essendo tali macchine non-specializzate come avviene nell'automazione rigida, le velocità di produzione di tali macchinari saranno decisamente più basse se paragonate al caso precedente.
Infine, l'*automazione flessibile* rappresenta l'evoluzione dell'automazione programmabile. Il suo campo di applicazione riguarda la produzione di lotti di dimensioni variabili di un vasta gamma di prodotti, con la minimizzazione dei *tempi morti* che si hanno nel passare dalla produzione di una tipologia di prodotto a un'altra.
Il robot industriale, per le sue caratteristiche di programmabilità, rientra nel campo della automazione programmabile, ma, a causa dell'adattabilità di tale macchina nell'eseguire diverse tipologie di lavorazioni che coinvolgono diversi tipi di prodotti, la robotica può tranquillamente rientrare anche nel campo della automazione flessibile.

1.2.1 Struttura meccanica dei robot

Tipicamente, un robot industriale a catena cinematica aperta è costituito da:

- un manipolatore, ovvero la struttura meccanica del robot (figura 1), costituito da un insieme di corpi rigidi (*bracci*) connessi tra di loro per mezzo di articolazioni (*giunti*), i quali, oltre a connettere i bracci, ne consentono anche un moto relativo, fornendo una o più gradi di libertà alla struttura. Da un punto di vista strutturale, nel manipolatore si individuano una *struttura portante*, che ne consente la mobilità, un *polso*, che consente l'orientamento dell'*organo terminale* con cui il robot esegue il compito che gli viene assegnato;

- gli *attuatori* sono i motori (elettrici, idraulici o pneumatici) che vengono inseriti sui giunti del robot e consentono allo stesso di compiere i movimenti richiesti;

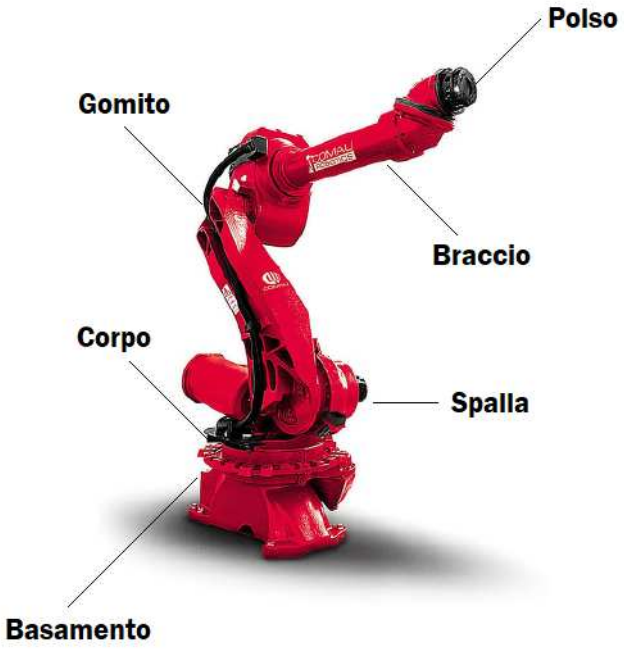


Figura 1: Struttura meccanica di un robot antropomorfo

- la *sensoristica* del robot è costituito dall'insieme dei sensori che rilevano grandezze (posizione, velocità, accelerazione, distanza relativa rispetto a un altro corpo,...) sia interne (in tal caso si parla di sensori propriocettivi) che esterne (vengono allora definiti sensori esterocettivi);

- un'*unità di governo* del robot, che, in base alla traiettoria e velocità desiderata del dispositivo terminale, calcola istante per istante (l'intervallo di tempo tra due istanti di controllo successivi dipende dalla potenza di calcolo di questa unità) lo spostamento da imprimere a ciascun giunto della macchina

Nell'ambito industriale, i robot possono svolgere operazioni di trasporto (per esempio si possono utilizzare nelle fasi di carico/scarico delle merci, nei processi di pallettizzazione delle parti da lavorare, nel set-up delle macchine utensili, ecc...), di manipolazione delle parti (ad esempio possono spostare la torcia di saldatura nelle saldature MIG), oppure possono essere utilizzate nelle operazioni di misura e collaudo dei prodotti.

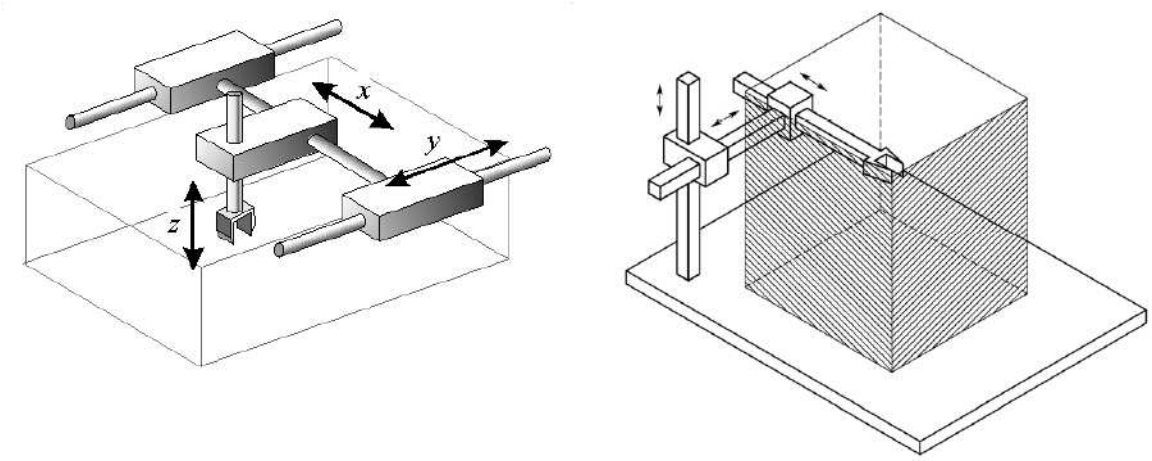


Figura 2: Manipolatore cartesiano e suo spazio di lavoro

1.2.2 Struttura dei manipolatori

Come già detto prima, un manipolatore è una struttura costituita da una serie di corpi rigidi collegati mediante dei giunti che ne consentono la mobilità. I giunti possono essere prismatici o rotoidali. Entrambe le tipologie di giunto lasciano soltanto un grado di libertà tra i corpi che connettono, ma, mentre i giunti prismatici permettono un moto di pura traslazione, i giunti rotoidali consentono una rotazione relativa tra i due bracci.
Per spostare e orientare un corpo nello spazio sono richiesti sei gradi di libertà, quindi se un robot possiede più gradi di libertà di quanto richiesto dal compito che deve svolgere, esso viene definito *ridondante*. Tipicamente, al braccio del robot spetta il compito di spostare l'organo terminale nello spazio, mentre il polso deve orientarlo.
Lo *spazio di lavoro* di un robot rappresenta la porzione dello spazio circostante che esso può raggiungere in base al mobilità posseduta dalla sua struttura; in base alle tipologie di giunti che costituiscono il manipolatore cambia la geometria dello spazio di lavoro, mentre le dimensioni di tale spazio sono dettate anche dalle dimensioni dei corpi che costituiscono il sistema. In base alla tipologia e successione dei giunti che costituiscono la struttura portante, i manipolatori si possono distinguere in *cartesiani, cilindrici, sferici, antropomorfi* e *SCARA*.
Il manipolatore con la configurazione *cartesiana* (figura 2) ha tre giunti prismatici, con assi solitamente ortogonali tra loro. La semplicità costruttiva di questo tipo di manipolatori fa sì che ad ogni grado di mobilità corrisponde un grado di traslazione nello spazio cartesiano, perciò lo spazio di lavoro ha una forma a parallelepipedo, come mostrato nella parte destra della figura 2, e in tale spazio il robot ha ottime caratteristiche di precisione. Di contro, la

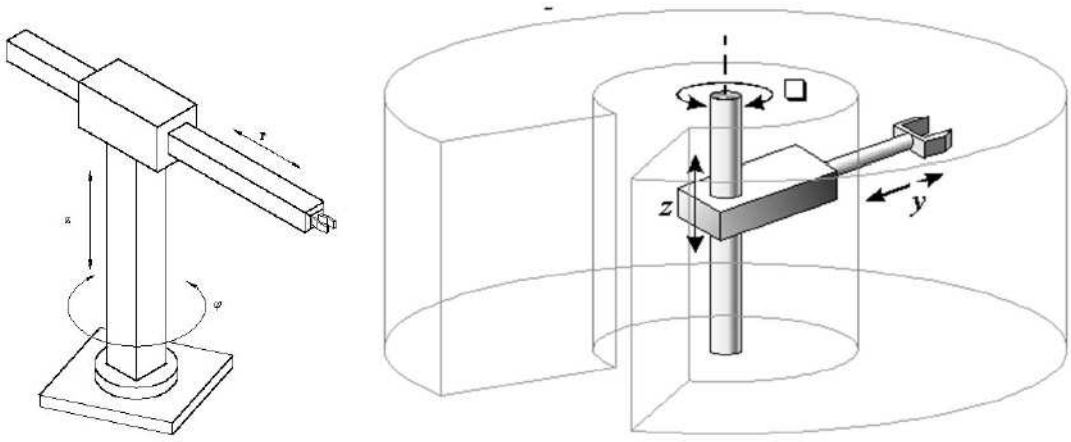


Figura 3: Manipolatore cilindrico e suo spazio di lavoro

presenza dei giunti prismatici comporta una scarsa destrezza del manipolatore.
Il manipolatore *cilindrico* (figura 3) differisce da quello cartesiano per la sostituzione del giunto prismatico di base con un giunto rotoidale. Ne consegue che lo spazio di lavoro assume una forma a cilindro cavo, come riportato nella parte destra della figura 3. In questo caso la precisione di posizionamento diminuisce al crescere dello sbraccio orizzontale. Spesso questa tipologia di robot sono utilizzati per il trasporto di oggetti, anche di masse relativamente elevate, in tal caso è preferibile utilizzare motori idraulici a quelli elettrici.

I manipolatori *sferici* (figura 4) sono costituiti dai primi due giunti dalla base rotoidali e l'ultimo giunto prismatico, e il relativo spazio di lavoro è una sfera cava, come riportato nel lato destro della figura 4. La precisione di posizionamento del polso si riduce al crescere dello sbraccio radiale. Questa tipologia di manipolatore viene utilizzato nelle lavorazioni e, in tal caso, si preferiscono motori elettrici per tutti i giunti del robot.

La geometria *antropomorfa* (figura 5) dei manipolatori prevede la presenza di tre giunti rotoidali, dove l'asse del giunto di base è ortogonale agli altri due tra loro paralleli. Grazie alla geometria composta soltanto da giunti rotoidali, questi robot sono i più destri tra quelli visti finora, mentre come contro hanno la geometria dello spazio di lavoro complessa (figura 5), data da una porzione di sfera il cui volume dipende dalla lunghezza dei bracci del manipolatore. Questi robot sono geometricamente molto simili al braccio umano e hanno un largo utilizzo in campo industriale, in numerose tipologie di lavorazioni.

Infine, i robot *SCARA* (6), sono una tipologia particolare di robot, realizzato in modo che due giunti rotoidali e un prismatico abbiano gli assi di moto tutti paralleli tra loro.

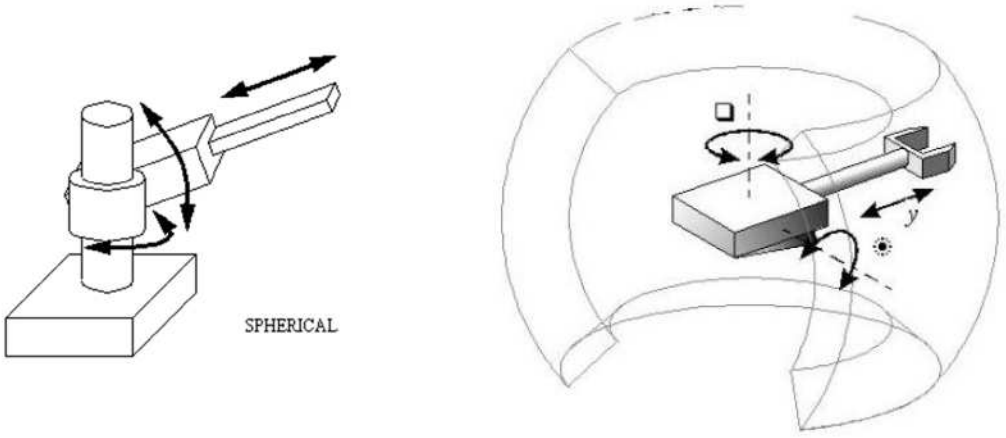


Figura 4: Manipolatori sferici e loro spazio di lavoro

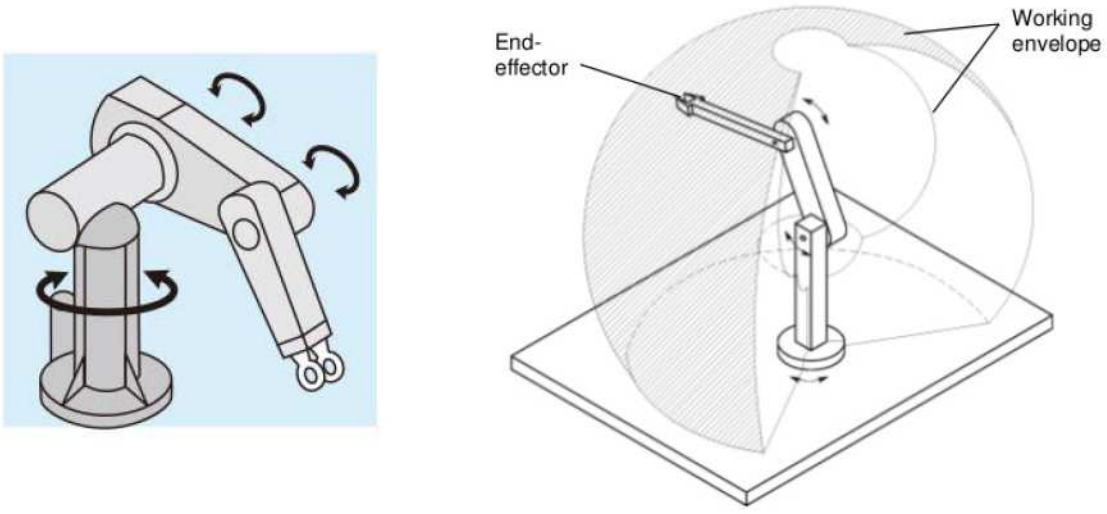


Figura 5: Manipolatore antropomorfo e suo soazio di lavoro

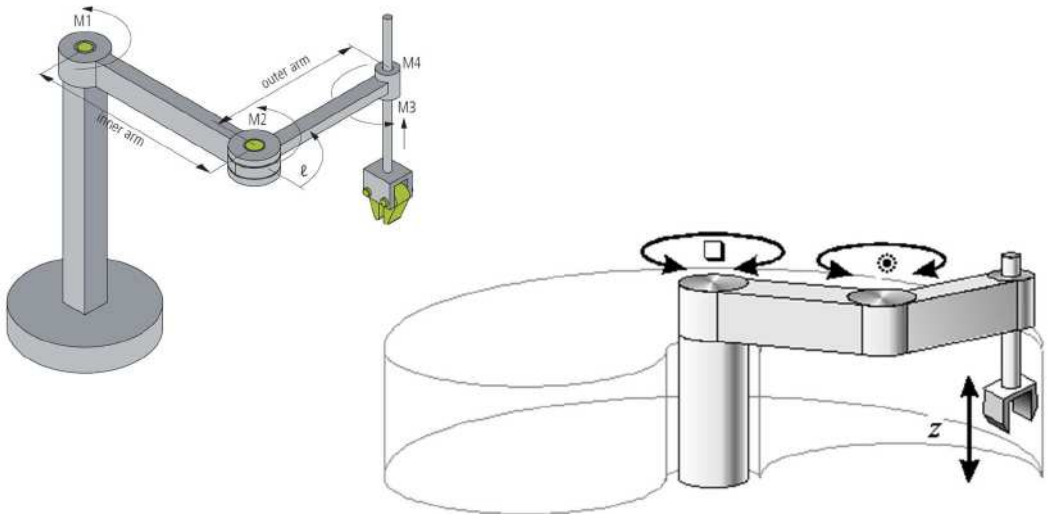


Figura 6: Robot SCARA e suo spazio di lavoro

L'acronimo *SCARA* deriva dalla notazione *Selective Compliance Assembly Robot Arm*, specifica che questa tipologia di robot presentano un'elevata rigidezza ai carichi verticali e cedevolezza ai carichi orizzontali, perciò la struttura SCARA è adatta per operazioni di assemblaggio verticale.

1.3 Robot collaborativi

Mentre i robot tradizionali necessitano di essere isolati in opportune aree di lavoro sicure e protette, lo sviluppo della robotica collaborativa nell'ottica dell'Industria 4.0, ha portato alla nascita di robot capaci di condividere lo spazio di lavoro con l'operatore umano, dotati di sistemi di sicurezza necessari per garantire l'incolumità non solo dell'uomo, ma anche del robot stesso. Questa nuova generazione di robot collaborativi (denominati *cobot*), risultano essere più leggeri e meno potenti rispetti alle alternative, in confronto a quest'ultimi però, garantiscono una maggiore sicurezza e flessibilità nelle lavorazioni. A causa dell'assenza di barriere fisiche tra questa tipologia di macchine e l'uomo, i cobot sono dotati di elevate capacità sensoristiche, necessarie anche per compensare la perdita di accuratezza che si ha per alleggerimento strutturale. In aggiunta, l'interfaccia uomo-macchina per il controllo risulta notevolmente semplificata, consentendo una programmazione intuitiva e veloce, a favore di una maggiore flessibilità di utilizzo.

I robot collaborativi sono utilizzati in ambito industriale per una vasta gamma di lavorazioni, che vanno dalle operazioni di *pick & place*, a operazioni di assemblaggio di componenti

Figura 7: Robot collaborativo utilizzato in campo medico: *ROBERT, Life Science Robotics, Aalborg (DK)*

di dimensioni ridotte (i sensori di forza presenti su queste macchine li rendono adatti a maneggiare componenti delicati e fragili); inoltre, si stanno studiando numerose applicazioni questa tipologia di robot fuori dal campo industriale, ad esempio la figura 7 mostra l'uso di tali macchine per la riabilitazione neuromuscolare, cosa facilitata dalla flessibilità della macchina, che permette esercizi personalizzati, l'elevata ripetibilità e la facilità di programmazione attraverso autoapprendimento guidato dal terapista [11].

1.3.1 Misure di sicurezza per i cobot

Da quanto detto finora si può intuire che aspetto molto importante relativo ai robot collaborativi riguarda la sicurezza. Le normative tecniche riguardo tale caratteristica sono le *EN ISO10218-1/2 e ISO/TS 15066* e sostanzialmente prevedono quattro modalità di gestione dello spazio condiviso tra l'operatore e il cobot:

1. **Stop di sicurezza monitorato (SMS)**
   Questa modalità di collaborazione prevede che l'operatore umano e il cobot condividano lo spazio di lavoro, ma svolgono ognuno una lavorazione diversa. Nel caso l'uomo entri nell'area di lavoro del robot, quest'ultimo arresta immediatamente il proprio moto (figura 8). Questo tipo di precauzione è adottata anche per i robot tradizionali, ma a differenza di quest'ultimi, gli attuatori dei cobot restano accesi durante l'arresto di sicurezza, per poi riprendere immediatamente il moto una volta che l'operatore si sia allontanato dalla macchina.

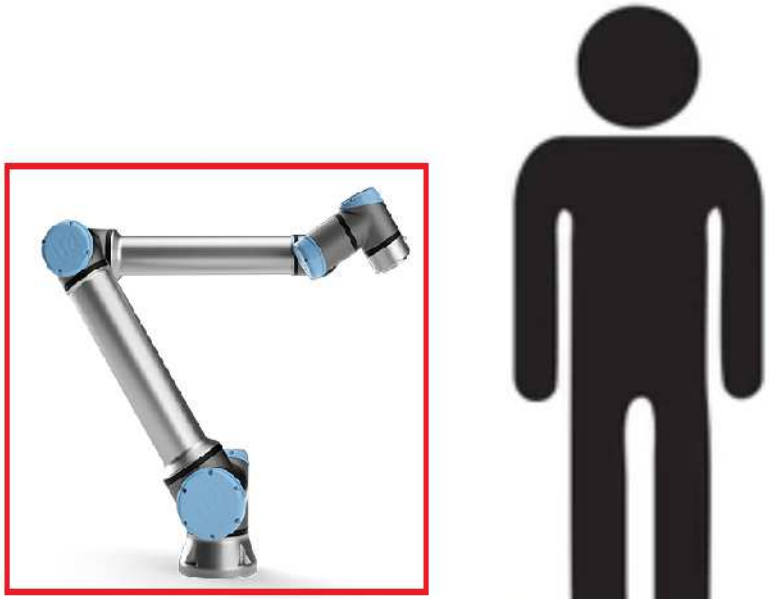


Figura 8: Stop di sicurezza monitorato

2. **Guida manuale (HG)**

   La modalità di *guida manuale* prevede la possibilità per l'utente di insegnare a robot un moto senza l'ausilio del teach pendant (console di controllo). Questa funzione permette all'operatore di muovere l'organo terminale del robot con il minimo sforzo fisico, grazie agli attuatori del manipolatore che lo assistono in tale moto (figura 9). L'unità dii controllo del robot memorizza il moto impartito così dall'esterno, riuscendo poi a replicare tale movimento in completa autonomia durante la fase esecutiva.

3. **Monitoraggio della separazione e della velocità (SSM)**

   Prendendo come riferimento la figura 10, questa modalità di collaborazione consiste nel regolare la velocità del robot in funzione della distanza dall'operatore umano. Nella zona verde l'uomo si trova al di fuori dello spazio di lavoro del robot, perciò questi può continuare indisturbato il proprio lavoro; nella zona gialla l'operatore si è avvicinato alla macchina, ma non abbastanza da trovarsi a rischio di collisione con questa, perciò il robot, come precauzione, riduce la propria velocità senza però fermarsi del tutto; all'interno della zona rossa, invece, il robot può entrare in collisione con l'uomo, perciò, al fine di garantirne la sicurezza, il robot arresta il proprio moto. Tuttavia, durante questo arresto, gli attuatori della macchina restano accesi, in modo da riprendere il moto appena l'operatore si sposta fuori dalla zona rossa (in maniera analoga a quanto descritto nello *stop di sicurezza monitorato*).

Figura 9: Esempio di guida manuale per cobot (UR5), *Universal Robots*

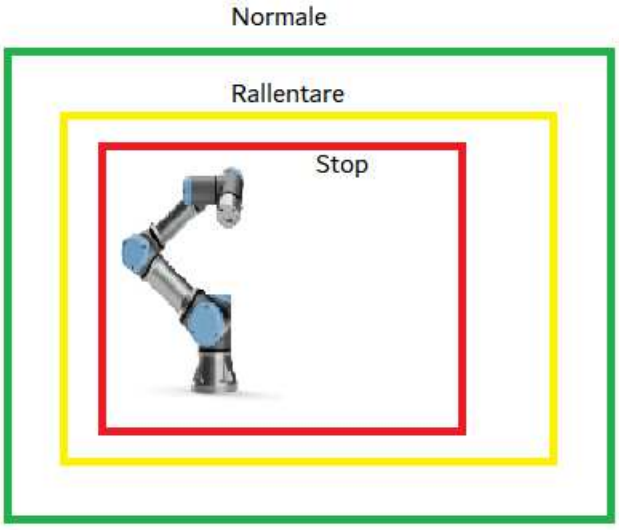


Figura 10: Schema della regolazione della velocità in base alla distanza dall'operatore

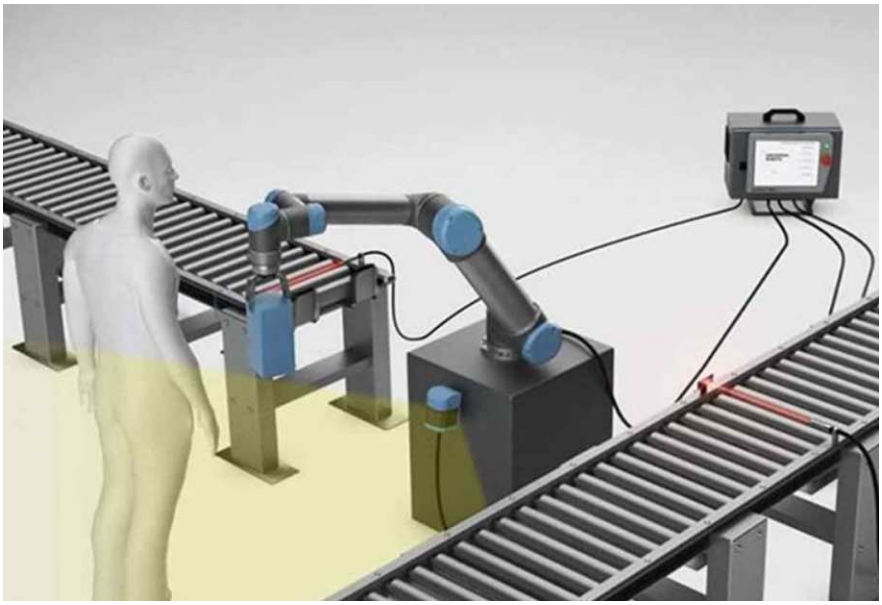


Figura 11: Esempio di spazio di lavoro condiviso uomo-robot, *Universal Robots*

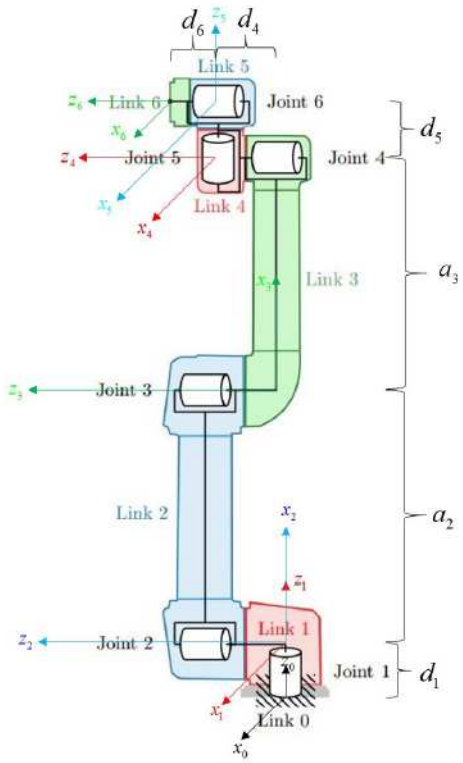
4. **Limitazione della forza e della potenza** In quest'ultimo caso, il robot e l'operatore non solo condividono lo spazio di lavoro, ma lavorano contemporaneamente lo stesso prodotto, come, ad esempio, visibile il caso della figura 11. Per questo genere di lavorazione ci può essere contatto fisico tra l'uomo e il robot, perciò è necessario limitare la potenza degli attuatori della macchina in modo da limitare la forza esercitabile, quindi garantire la sicurezza del lavoratore.

1.3.2 Esempio di robot collaborativo

Come esempio di robot collaborativo, si prenda il modello *UR5* (figura 12a) della Universal Robots, un'azienda specializzata nella produzione di robot industriali collaborativi. Trattandosi di un cobot, la sua massa ha un valore molto bassa se paragonato ai robot tradizionali, pari a 20,6 kg, con una capacità di carico nominale di 5 kg.

In figura 12b è riportato uno schema cinematico del robot, mentre nella tabella 1 sono riportati i valori delle dimensioni dei vari corpi rigidi che costituiscono l'UR5. Nei capitoli successivi di questa trattazione, verrà utilizzato un modello CAD di questo robot per eseguire simulazioni e verificare la validità degli algoritmi di pianificazione moto ed obstacle avoidance.

(a) Robot collaborativo UR5



(b) Schema cinematico e assi di un robot UR5

Figura 12: Robot UR5 e suo schema cinematico

i	a_i	d_i	α_i
0	0	-	0
1	0	0.08916	$\pi/2$
2	0.425	-	0
3	0.39225	0	0
4	0	0.10915	$\pi/2$
5	0	0.09456	$-\pi/2$
6	-	0.0823	-

Tabella 1: Dimensioni dell'UR5

2 Cinematica

Un robot è costituito da un manipolatore, che ha il compito di spostare un dispositivo terminale nello spazio, in modo da compiere un certo lavoro (può essere una semplice operazione di *pick and place*, dove ciò che conta sono solamente le posizioni iniziali e finali raggiunte, oppure una lavorazione più complessa come un processo di saldatura o verniciatura, dove, oltre alle posizioni, è fondamentale anche riuscire a programmare in maniera accurata la traiettoria seguita dal robot).
Come già visto nel paragrafo 1.2.1, da un punto di vista meccanico, il manipolatore è una catena cinematica costituita da una serie di corpi rigidi (detti *bracci*) collegati tramite una serie di giunti, rotoidali o prismatici, che permettono, rispettivamente, un moto rotatorio o di pura traslazione tra i due bracci connessi, riuscendo così a compiere una serie di movimenti nello spazio. Il moto complessivo di questa struttura si ottiene per composizione del moto di ogni singolo corpo rigido che lo compone, rispetto al precedente. L'obiettivo finale è quello di riuscire a mobilitare un dispositivo finale nello spazio, perciò è necessario, istante per istante, riuscire a calcolare la posizione e l'orientamento di tale dispositivo.
Nella successiva trattazione si vedrà come ottenere le equazioni della cinematica diretta, che consente di esprimere la **posa** (posizione + orientamento) dell'organo terminale rispetto alla terna base posta sul telaio.

2.1 Posizione ed orientamento di un corpo rigido nello spazio

Un corpo rigido risulta completamente descritto nello spazio mediante la sua *posa*, ovvero posizione ed orientamento espressi rispetto alla terna base. Nei successivi paragrafi si vedrà

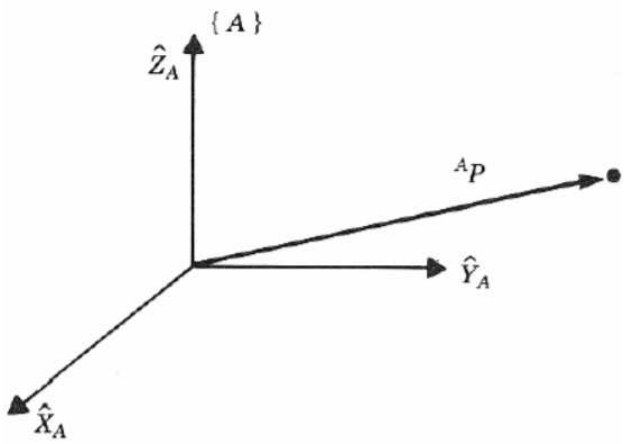


Figura 13: Posizione di un punto rispetto una terna di rifermento

come descrivere la posa di un corpo rigido rispetto a una terna di riferimento (immobile).

2.1.1 Posizione

Con rifermento alla figura 13, si consideri un sistema di riferimento cartesiano ortonormale **A**, la posizione di un corpo rigido nello spazio viene solitamente descritto mediante il *vettore posizione*:

$$^{\mathbf{A}}\mathbf{P} = \mathbf{P} - \mathbf{P}_{\mathbf{AORG}}$$

ovvero il vettore che esprime la distanza tra il punto P e l'origine della terna **A**. La posizione del corpo rigido può essere definita in funzione di tre variabili (in quanto sono possibili tre traslazioni nello spazio), che coincidono con le coordinate cartesiane del punto P nella terna di rifermento:

$$^{\mathbf{A}}\mathbf{P} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$

Considerando invece la situazione espressa in figura 14, dove si ha un moto di pura traslazione tra due terne **A** e **B**, e nota la posizione $^{\mathbf{B}}\mathbf{P}$ del punto P nella terna **B**, allora la posizione del suddetto punto nella terna **A** può essere espresso tramite la seguente relazione:

$$^{\mathbf{A}}\mathbf{P} = {}^{\mathbf{B}}\mathbf{P} + {}^{\mathbf{A}}\mathbf{P}_{\mathbf{BORG}}$$

2.1.2 Orientamento

Come mostrato nella figura 15, risulta comodo fissare una terna solidale al corpo rigido, in modo da poter fare riferimento a tale terna per esprimere l'orientamento del corpo stesso

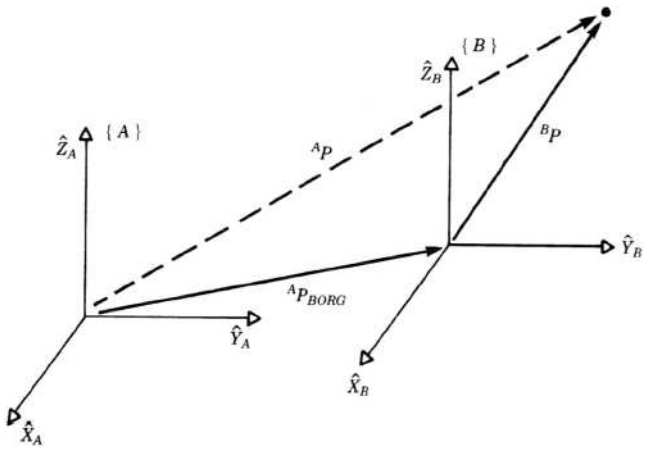


Figura 14: Moto di pura traslazione tra due terne cartesiane

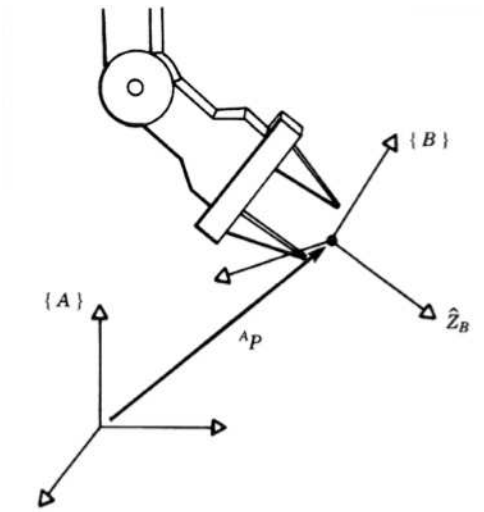


Figura 15: Rotazione relativa tra due terne

rispetto alla terna base.
Oltre ai tre gradi di libertà forniti dalla posizione del corpo rigido nello spazio, l'orientamento dello stesso fornisce altri tre gradi di libertà.
A tal fine si possono prendere i tre versori degli assi principali di $\mathbf{B}$, ovvero $\hat{\mathbf{x}_B}$, $\hat{\mathbf{y}_B}$ e $\hat{\mathbf{z}_B}$, e si possono esprimere rispetto alla terna di riferimento $\mathbf{A}$ dalle seguente equazioni:

$$\begin{aligned} {}^{\mathbf{A}}\hat{\mathbf{x}}_{\mathbf{B}} &= \hat{\mathbf{x}}_{\mathbf{B}} \cdot \hat{\mathbf{x}}_{\mathbf{A}} + \hat{\mathbf{y}}_{\mathbf{B}} \cdot \hat{\mathbf{x}}_{\mathbf{A}} + \hat{\mathbf{z}}_{\mathbf{B}} \cdot \hat{\mathbf{x}}_{\mathbf{A}} \\ {}^{\mathbf{A}}\hat{\mathbf{y}}_{\mathbf{B}} &= \hat{\mathbf{x}}_{\mathbf{B}} \cdot \hat{\mathbf{y}}_{\mathbf{A}} + \hat{\mathbf{y}}_{\mathbf{B}} \cdot \hat{\mathbf{y}}_{\mathbf{A}} + \hat{\mathbf{z}}_{\mathbf{B}} \cdot \hat{\mathbf{y}}_{\mathbf{A}} \\ {}^{\mathbf{A}}\hat{\mathbf{z}}_{\mathbf{B}} &= \hat{\mathbf{x}}_{\mathbf{B}} \cdot \hat{\mathbf{z}}_{\mathbf{A}} + \hat{\mathbf{y}}_{\mathbf{B}} \cdot \hat{\mathbf{z}}_{\mathbf{A}} + \hat{\mathbf{z}}_{\mathbf{B}} \cdot \hat{\mathbf{z}}_{\mathbf{A}} \end{aligned} \tag{2.1}$$

Utilizzando la notazione compatta, i versori presenti in (2.1), possono essere opportunamente combinati al fine di ottenere la **matrice di rotazione** (3x3):

$$\mathbf{{}^{A}_{B}R} = \begin{bmatrix} {}^{\mathbf{A}}\hat{\mathbf{x}}_{\mathbf{B}} & {}^{\mathbf{A}}\hat{\mathbf{y}}_{\mathbf{B}} & {}^{\mathbf{A}}\hat{\mathbf{z}}_{\mathbf{B}} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{x}}_{\mathbf{B}} \cdot \hat{\mathbf{x}}_{\mathbf{A}} & \hat{\mathbf{y}}_{\mathbf{B}} \cdot \hat{\mathbf{x}}_{\mathbf{A}} & \hat{\mathbf{z}}_{\mathbf{B}} \cdot \hat{\mathbf{x}}_{\mathbf{A}} \\ \hat{\mathbf{x}}_{\mathbf{B}} \cdot \hat{\mathbf{y}}_{\mathbf{A}} & \hat{\mathbf{y}}_{\mathbf{B}} \cdot \hat{\mathbf{y}}_{\mathbf{A}} & \hat{\mathbf{z}}_{\mathbf{B}} \cdot \hat{\mathbf{y}}_{\mathbf{A}} \\ \hat{\mathbf{x}}_{\mathbf{B}} \cdot \hat{\mathbf{z}}_{\mathbf{A}} & \hat{\mathbf{y}}_{\mathbf{B}} \cdot \hat{\mathbf{z}}_{\mathbf{A}} & \hat{\mathbf{z}}_{\mathbf{B}} \cdot \hat{\mathbf{z}}_{\mathbf{A}} \end{bmatrix} \tag{2.2}$$

Bisogna notare che nell'espressione (2.2) il sistema di riferimento non risulta più importante, in quanto il prodotto scalare risulta invariante rispetto al sistema di riferimento scelto. Inoltre, dato che $\hat{\mathbf{x}}_{\mathbf{A}} \cdot \hat{\mathbf{x}}_{\mathbf{B}} = cos(\hat{x}_B \hat{x}_A)$, tale matrice viene anche detta *matrice dei coseni direttori*. Le matrici di rotazione sono matrici ortonormali, quindi i vettori colonna di tali matrici sono ortogonali fra loro e hanno modulo unitario:

$$\hat{\mathbf{x}}_{\mathbf{B}} \cdot \hat{\mathbf{x}}_{\mathbf{A}} = 1 \qquad \hat{\mathbf{y}}_{\mathbf{B}} \cdot \hat{\mathbf{y}}_{\mathbf{A}} = 1 \qquad \hat{\mathbf{z}}_{\mathbf{B}} \cdot \hat{\mathbf{z}}_{\mathbf{A}} = 1$$

Perciò la matrice $\mathbf{R}$ risulta *ortogonale*, ovvero gode della proprietà:

$$\mathbf{R}^T \mathbf{R} = \mathbf{I}_3 \tag{2.3}$$

dove $\mathbf{I}_3$ indica la matrice identità di dimensioni (3x3).
Se si moltiplicano da destra entrambi i membri della (2.3) per la matrice inversa $\mathbf{R}^{-1}$, si ottiene:

$$\mathbf{R}^T = \mathbf{R}^{-1} \tag{2.4}$$

ovvero la trasposta della matrice di rotazione coincide con la sua inversa.

2.1.3 Rotazioni elementari

Si consideri ora il caso delle terne ottenibili dalla terna di riferimento mediante delle rotazioni elementari effettuate attorno a un asse coordinato. Ad esempio, nella figura 16 avviene una

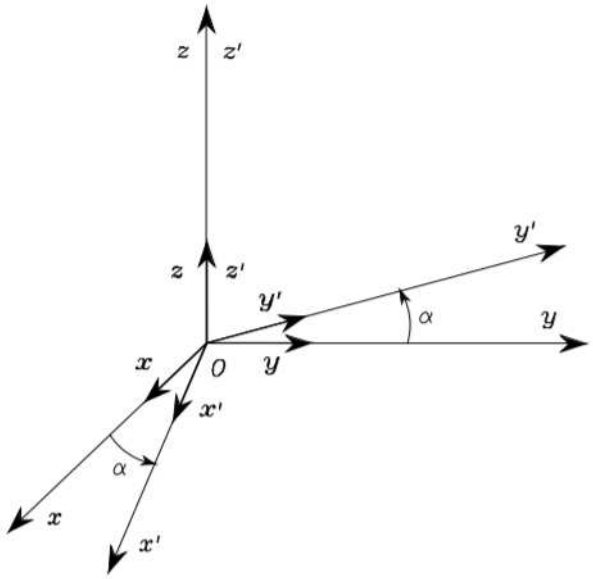


Figura 16: Rotazione della terna $O - xyz$ intorno all'asse z

rotazione attorno all'asse z della terna di riferimento iniziale di un angolo α (le rotazioni sono positive se effettuate in senso antiorario intorno all'asse, e negative viceversa). Si ottiene così la nuova terna $O - x'y'z'$, i cui componenti possono essere caratterizzati rispetto agli assi della terna di riferimento, ovvero:

$$x' = \begin{bmatrix} \cos\alpha \\ \sin\alpha \\ 0 \end{bmatrix} \qquad y' = \begin{bmatrix} -\sin\alpha \\ \cos\alpha \\ 0 \end{bmatrix} \qquad z' = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.5}$$

La matrice composta da tali vettori sarà la matrice di rotazione $\mathbf{R}_z(\alpha)$ della terna $O - x'y'z'$ rispetto alla terna $O - xyz$:

$$\mathbf{R}_z(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

Analogamente si trova che le rotazioni di un angolo β intorno all'asse y e di un angolo γ intorno all'asse x forniscono, rispettivamente, le seguenti matrici di rotazione:

$$\mathbf{R}_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \tag{2.7}$$

$$\mathbf{R}_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix} \tag{2.8}$$

2.2 Composizione di matrici di rotazione

Per spiegare la composizione di rotazioni, si parte da un terna di riferimento **A**, alla quale viene applicata una rotazione, espressa dalla matrice ${}^{\mathbf{A}}_{\mathbf{B}}\mathbf{R}$, ottenendo in questo modo la terna **B**; successivamente, si parte dalla terna **B** appena ricavate (verrà definita *terna corrente*), a cui si applica la matrice di rotazione nota ${}^{\mathbf{B}}_{\mathbf{C}}\mathbf{R}$, ottenendo la terna finale **C**. Applicando la composizione delle matrici a questo caso, si ottiene la matrice di rotazione ${}^{\mathbf{A}}_{\mathbf{C}}\mathbf{R}$, che esprime l'orientamento della terna **C** rispetto alla terna di riferimento **A**:

$${}^{\mathbf{A}}_{\mathbf{C}}\mathbf{R} = {}^{\mathbf{A}}_{\mathbf{B}}\mathbf{R}\,{}^{\mathbf{B}}_{\mathbf{C}}\mathbf{R} \tag{2.9}$$

Dunque, noto il vettore posizione di un punto P nella terna **C** (${}^{\mathbf{C}}\mathbf{P}$), si può ricavare il vettore che esprime la posizione del medesimo punto nella terna **A** dalla seguente notazione:

$${}^{A}P = {}^{\mathbf{A}}_{\mathbf{B}}\mathbf{R}\,{}^{B}P = {}^{\mathbf{A}}_{\mathbf{B}}\mathbf{R}({}^{\mathbf{B}}_{\mathbf{C}}\mathbf{R}\,{}^{B}P) = {}^{\mathbf{A}}_{\mathbf{B}}\mathbf{R}\,{}^{\mathbf{B}}_{\mathbf{C}}\mathbf{R}\,{}^{C}P$$

Si può concludere dicendo che la composizione di matrici successive attorno a terne mobili, si ottiene *post-moltiplicando*, ovvero moltiplicando da sinistra verso destra, le matrici di rotazione.
Alternativamente, le rotazioni, invece di riferirsi alla terna corrente (ovvero quella ruotata), potrebbero anche essere relative alla terna di riferimento (ovvero la terna fissa) e, in tal caso, la composizione delle matrici di rotazione avviene *pre-moltiplicando*, cioè moltiplicando da destra verso sinistra le stesse, come riportato di seguito:

$${}^{\mathbf{A}}_{\mathbf{C}}\mathbf{R} = {}^{\mathbf{B}}_{\mathbf{C}}\mathbf{R}\,{}^{\mathbf{A}}_{\mathbf{B}}\mathbf{R} \tag{2.10}$$

Bisognerebbe notare come l'espressione 2.10 sia diversa da 2.9, perciò eseguire rotazioni successive attorno a terne fisse o mobili porta a risultati differenti.

2.3 Angoli di Eulero

La descrizione dell'orientamento una terna fornita dalle matrici di rotazione risulta ridondante in quanto tali matrici sono formate da nove elementi dipendenti tra loro secondo la legge di ortogonalità espressa in (2.3). Questa considerazione conduce a formulare una *rappresentazione*

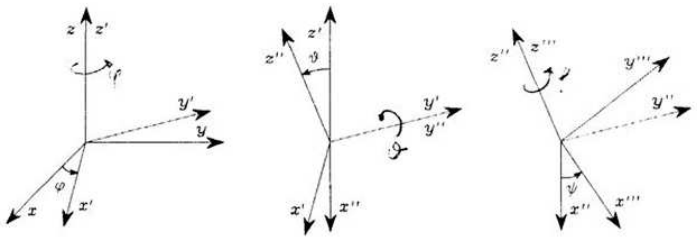


Figura 17: Rappresentazione degli angoli ZYZ

minima dell'orientamento, in quanto il numero di parametri indipendenti effettivamente necessari per la descrizione dell'orientamento di una terna nello spazio sono tre. Dunque, si può ottenere una rappresentazione minima utilizzando una terna di angoli, $\phi = [\varphi \quad \theta \quad \psi]^T$, per la descrizione dell'orientamento. Ciascuno di questi angoli corrisponde a rotazione elementare attorno a uno dei tre assi coordinati della terna corrente; perciò, una generica matrice di rotazione (ovvero che esprima una rotazione attorno a un asse orientato arbitrariamente nello spazio) può essere ricavata come la composizione di tre rotazioni elementari secondo opportune sequenze, a patto che due rotazioni successive non avvengano attorno ad assi paralleli. Questa limitazione comporta che delle $27(=3^3)$ possibili combinazioni di rotazioni, solamente 12 siano considerate fattibili e ciascuna di queste combinazioni rappresenta una terna di *angoli di Eulero*. In questa trattazione verrà applicata la terna di angoli di Eulero data dalle rotazioni successive attorno agli assi ZYZ, detti anche *angoli di Roll-Pitch-Roll*.

2.3.1 Angoli ZYZ o Roll-Pitch-Roll

Gli angoli di Roll-Pitch-Roll sono dati dalle combinazioni delle seguenti matrici di rotazione elementari (Figura 17):

1. rotazione della terna base attorno all'asse z di un angolo φ, con la relativa matrice $\mathbf{R}_z(\varphi)$, data da (2.6);

2. rotazione della terna corrente attorno all'asse y' dell'angolo θ, con la relativa matrice di rotazione $\mathbf{R}_{y'}(\theta)$ data da (2.7);

3. rotazione della corrente attorno all'asse z'' dell'angolo ψ, con la relativa matrice $\mathbf{R}_{z''}(\psi)$ definita nuovamente da (2.6).

La matrice di rotazione finale della terna, dato che le rotazioni sono state eseguite attorno agli assi della terna corrente (ovvero quella mobile), si ottiene post-moltiplicando, cioè

moltiplicando da sinistra verso destra, le matrici di rotazione elementare eseguite

$$\mathbf{R}(\phi) = \mathbf{R}_z(\varphi)\mathbf{R}_{y'}(\theta)\mathbf{R}_{z''} = \begin{pmatrix} c_\varphi c_\theta c_\psi - s_\varphi s_\psi & -c_\varphi c_\theta s_\psi - s_\varphi c_\psi & c_\varphi s_\theta \\ s_\varphi c_\theta c_\psi + c_\varphi s_\psi & -s_\varphi c_\theta s_\psi + c_\varphi c_\psi & s_\varphi s_\theta \\ -s_\theta c_\psi & s_\theta s_\psi & c_\theta \end{pmatrix} \tag{2.11}$$

In seguito tornerà utile la risoluzione del problema inverso, vale a dire determinazione degli angoli di Eulero relativi alle matrici di rotazione note

$$\mathbf{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \tag{2.12}$$

Confrontando tale equazione con quella di $\mathbf{R}(\phi)$ in (2.11), si possono fare le seguenti considerazioni: dagli elementi r_{13} e r_{23}, nell'ipotesi che $r_{13} \neq 0$ e $r_{23} \neq 0$, si ottiene

$$\varphi = atan2(r_{13}, r_{23})$$

Quindi, quadrando e sommando gli elementi r_{13} e r_{23} e utilizzando l'elemento r_{33}, si ottiene:

$$\theta = atan2(\sqrt{r_{13}^2 + r_{23}^2}, r_{33})$$

La scelta del segno positivo per il termine $\sqrt{r_{13}^2 + r_{23}^2}$ restringe a $(0, \pi)$ l'intervallo di appartenenza di θ. Perciò, considerando gli elementi r_{31} e r_{32}, si ha:

$$\psi = atan2(r_{32}, -r_{31})$$

In definitiva, la soluzione cercata risulta:

$$\begin{aligned} \varphi &= atan2(r_{13}, r_{23}) \\ \theta &= atan2(\sqrt{r_{13}^2 + r_{23}^2}, r_{33}) \\ \psi &= atan2(r_{32}, -r_{31}) \end{aligned} \tag{2.13}$$

La precedente soluzione (2.13) degenera quando $s_\theta = 0$, in tal caso è possibile determinare soltanto la somma o la differenza di φ e ψ. Infatti, se $\theta = 0, \pi$, le rotazioni successive di φ e ψ sono effettuate intorno ad assi di terna corrente paralleli fra loro, fornendo così effetti di rotazione equivalenti.

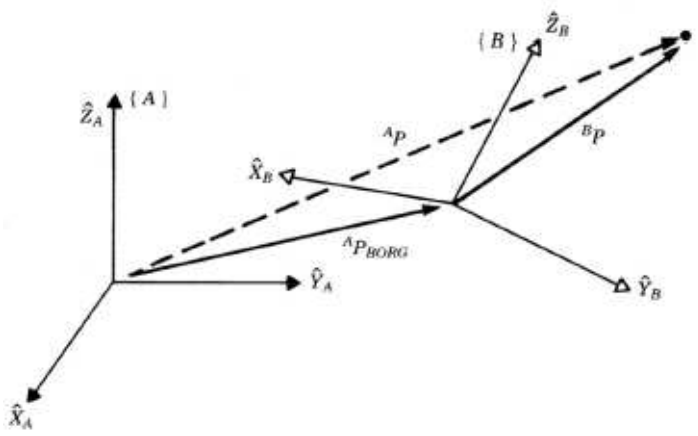


Figura 18: Moto di traslazione e rotazione tra due terne

2.4 Roto-traslazione

Il caso più comune che si può riscontrare nella realtà è quello di avere contemporaneamente una rotazione e una traslazione tra due terne inizialmente coincidenti. Considerando, ad esempio, la situazione mostrata nella figura 18, le origini delle due terne **A** e **B** hanno subito una relativa traslazione, espressa dal vettore $^{\mathbf{A}}\mathbf{P}_{\mathbf{BORG}}$, inoltre è stata effettuata una rotazione relativa tra le due terne, espressa dalla matrice $^{\mathbf{A}}_{\mathbf{B}}\mathbf{R}$. Perciò, nota la posizione del punto P nella terna **B** (pari al vettore $^{\mathbf{B}}\mathbf{P}$), si vuole esprimere la posizione di tale punto nella terna fissa **A**. Considerando che i vettori sono enti le cui caratteristiche sono indipendenti dalla terna in cui vengono espressi, la posizione del punto nella terna **A** può essere espressa come:

$$^{\mathbf{A}}\mathbf{P} = {}^{\mathbf{B}}\mathbf{P} + {}^{\mathbf{A}}\mathbf{P}_{\mathbf{BORG}} \tag{2.14}$$

Risulta chiaro come i termini che compaiono nella (2.14) devono essere proiettati nella terna **A**. Mentre $^{\mathbf{A}}\mathbf{P}$ e $^{\mathbf{A}}\mathbf{P}_{\mathbf{BORG}}$ sono già espressi in tale terna, bisogna proiettare anche il termine $^{\mathbf{B}}\mathbf{P}$ in **A**, e per fare ciò bisogna pre-moltiplicare tale componente per la matrice di rotazione che va da **B** ad **A**:

$$^{\mathbf{A}}\mathbf{P} = {}^{\mathbf{A}}_{\mathbf{B}}\mathbf{R}\,{}^{\mathbf{B}}\mathbf{P} + {}^{\mathbf{A}}\mathbf{P}_{\mathbf{BORG}} \tag{2.15}$$

2.5 Trasformazioni omogenee

Si consideri la situazione di roto-traslazione tra due terne di riferimento, come riportato nella figura 18. Tra due terne vi è una traslazione, indicata dal vettore $^{\mathbf{A}}\mathbf{P}_{\mathbf{BORG}}$, e una relativa rotazione, indicata dalla matrice $^{\mathbf{A}}_{\mathbf{B}}\mathbf{R}$. Nell'equazione (2.15) è riportata la *trasformazione di*

coordinate di un vettore applicato da una terna all'altra.
Allo scopo di voler ottenere una rappresentazione compatta del legame esistente tra la rappresentazione di uno stesso punto in due terne differenti, si può introdurre la *matrice di trasformazione omogenea*, rappresentabile come segue:

$$\mathbf{{}^{A}_{B}T} = \begin{bmatrix} \mathbf{{}^{A}_{B}R} & \mathbf{{}^{A}P_{BORG}} \\ 0 & 1 \end{bmatrix} \tag{2.16}$$

Dunque, secondo quanto espresso nella (2.16), si tratta di una matrice (4x4), avente una componente (3x3) che sarebbe la matrice di rotazione, una componente (3x1) dal vettore posizione relativa tra le due terne, mentre nell'ultima riga ha tre zeri sotto la matrice di rotazione e un 1 sotto il vettore posizione.
Utilizzando tale notazione omogenea, nota la posizione di un punto P nella terna **B**, si può esprimere la posizione dello stesso punto nella terna **A** (figura 18) secondo il seguente calcolo:

$$\mathbf{{}^{A}P} = \mathbf{{}^{A}_{B}T\,{}^{B}P}$$

Nota la matrice di trasformazione tra due terne **A** e **B**, la matrice di trasformazione tra la terna **B** e una terza terna **C**, si può calcolare la matrice di trasformazione tra la terna iniziale e quella finale, applicando la seguente relazione di composizione delle matrici:

$$\mathbf{{}^{A}_{C}T} = \mathbf{{}^{A}_{B}T\,{}^{B}_{C}T} \tag{2.17}$$

3 Cinematica differenziale

La cinematica differenziale stabilisce una relazione lineare tra le velocità del dispositivo terminale nello spazio operativo (o cartesiano) e le velocità dei giunti del manipolatore. Tale relazione lineare è stabilita da una matrice, le cui dimensioni dipendono dalla configurazione del manipolatore, definita *Jacobiano geometrico* [1] .

3.1 Jacobiano geometrico

Considerando un manipolatore di grado n, la cinematica diretta di può scrivere nella notazione compatta (2.16), come riportato di seguito:

$$\mathbf{T(q)} = \begin{bmatrix} \mathbf{R(q)} & \mathbf{p(q)} \\ 0 & 1 \end{bmatrix}$$

dove $\mathbf{q} = [q_1, ..., q_n]^T$ contiene le variabili di giunto, che definiscono la posizione e l'orientamento reale dell'end effector.
Tramite la cinematica differenziale si vuole stabilire una relazione lineare tra la velocità lineare $\dot{\mathbf{x}}$ e angolare $\boldsymbol{\omega}$ dell'end effector in funzione delle variabili di giunto presenti nel vettore $\mathbf{q}$, ottenendo relazioni del tipo:

$$\dot{\mathbf{x}} = \mathbf{J}_x(\mathbf{q})\dot{\mathbf{q}} \tag{3.1}$$

$$\boldsymbol{\omega} = \mathbf{J}_O(\mathbf{q})\dot{\mathbf{q}} \tag{3.2}$$

Dove $\mathbf{J}_x$ e $\mathbf{J}_O$ sono delle matrici $(3 \times n)$ che rappresentano, rispettivamente, il contributo della velocità dei giunti alla velocità lineare $\dot{\mathbf{p}}$ e velocità angolare $\boldsymbol{\omega}$ del dispositivo terminale. Le relazioni (3.1) e (3.2) si possono si possono riscrivere in forma compatta come:

$$\mathbf{v} = \begin{bmatrix} \dot{\mathbf{x}} \\ \boldsymbol{\omega} \end{bmatrix} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{3.3}$$

L'equazione (3.3) rappresenta la *cinematica differenziale* del manipolatore, dove la matrice $\mathbf{J}$ (6 x n) rappresenta lo *Jacobiano geometrico* del manipolatore e generalmente risulta in funzione delle variabili di giunto

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_x \\ \mathbf{J}_O \end{bmatrix} \tag{3.4}$$

Per calcolare lo Jacobiano geometrico è opportuno richiamare delle proprietà delle matrici di rotazione.

3.1.1 Derivata di una matrice di rotazione

In questo capitolo si vuole calcolare la derivata della matrice di rotazione in funzione del tempo. Prendendo una matrice di rotazione $\mathbf{R}(t)$, variabile nel tempo, dalla proprietà di ortogonalità si ottiene della matrice si ottiene:

$$\mathbf{R}(t)\mathbf{R}^T(t) = \mathbf{I}$$

che, derivata nel tempo, fornisce

$$\dot{\mathbf{R}}(t)\mathbf{R}^T(t) + \mathbf{R}(t)\dot{\mathbf{R}}(t) = \mathbf{0}$$

Ponendo una matrice $\mathbf{S}(t)$ pari a :

$$\mathbf{S}(t) = \dot{\mathbf{R}}(t)\mathbf{R}^T(t) \tag{3.5}$$

Tale matrice risulta essere *anti-simmetrica*, in quanto

$$\mathbf{S}(t) + \mathbf{S}^T(t) = \mathbf{0} \tag{3.6}$$

Moltiplicando entrambi i membri della (3.5) per la matrice di rotazione $\mathbf{R}(t)$, si ottiene

$$\dot{\mathbf{R}}(t) = \mathbf{S}(t)\mathbf{R}(t) \tag{3.7}$$

da notare che la precedente equazione esprime la derivata di $\mathbf{R}(t)$ in funzione della matrice stessa.
Per avere una interpretazione fisica della (3.7), si consideri un vettore costante $\mathbf{x}'$ ed il vettore $\mathbf{x}(t) = \mathbf{R}(t)\mathbf{x}'$. La derivata temporale di $\mathbf{x}(t)$ si può scrivere come

$$\dot{\mathbf{x}}(t) = \dot{\mathbf{R}}(t)\mathbf{x}'$$

per cui la (3.7) si può scrivere come

$$\dot{\mathbf{x}}(t) = \mathbf{S}(t)\mathbf{R}(t)\mathbf{x}'$$

Con il vettore $\boldsymbol{\omega}(t)$ si indica la velocità angolare relativa alla matrice di rotazione all'istante di tempo t rispetto alla terna di riferimento, dalla meccanica di ha che

$$\dot{\mathbf{x}}(t) = \boldsymbol{\omega}(t) \times \mathbf{R}(t)\mathbf{x}'$$

Dunque, l'operatore matriciale $\mathbf{S}(t)$ descrive il prodotto vettoriale tra il vettore $\boldsymbol{\omega}$ e il vettore $\mathbf{R}(t)\mathbf{x}'$. Gli elementi simmetrici della matrice $\mathbf{S}(t)$ rispetto alla diagonale principale rappresentano le componenti del vettore $\boldsymbol{\omega}(t) = [\omega_x \quad \omega_y \quad \omega_z]^T$, nella forma

$$\mathbf{S} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}. \tag{3.8}$$

3.2 Singolarità cinematiche

Lo Jacobiano geometrico presente nell'equazione della cinematica differenziale (3.3) stabilisce una relazione lineare tra la velocità nello spazio cartesiano ($\mathbf{v}$) e la velocità nello spazio dei giunti ($\mathbf{q}$). Sempre nell'equazione (3.3) si può notare come lo Jacobiano sia in funzione della vettore $\mathbf{q}$ delle variabili di giunto, dunque si definisce *singolarità cinematica* di un

manipolatore, tutte le configurazioni in corrispondenza delle quali lo Jacobiano diminuisce di rango. Nell'utilizzo di un robot è molto importante conoscerne le singolarità, per via del problema della perdita di gradi di libertà della struttura, ovvero non tutti i moti sono possibili; inoltre, nelle configurazioni singolari, potrebbero esistere infinite soluzioni al problema cinematico inverso; infine, nell'intorno di una singolarità, pur mantenendo velocità contenute nello spazio cartesiano, si possono avere velocità molto elevate nello spazio dei giunti.
Esistono due tipologie di singolarità:

1. *Singolarità ai confini dello spazio di lavoro raggiungibile*: come intuibile dal titolo, queste singolarità si presentano qualora il manipolatore tenti di raggiungere punti al di fuori o sul bordo dello spazio di lavoro del robot, ovvero la zona che esso può raggiungere con il dispositivo terminale. Queste configurazioni singolari si possono evitare facilmente a condizione che il robot lavori sempre seguendo una traiettoria ben contenuta nel suo spazio di lavoro.

2. *Singolarità all'interno dello spazio di lavoro raggiungibile*: queste si verificano qualora il manipolatore assuma delle configurazioni particolari, come quelle ottenute dall'allineamento di due o più assi di moto. Queste classe di singolarità risultano più difficili da evitare rispetto a quelle precedenti, in quanto possono verificarsi anche all'interno dello spazio di lavoro per determinate configurazioni nello spazio dei giunti.

3.3 Problema cinematico inverso

É stato già visto come la relazione $\mathbf{v} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$, espressa in (3.3), stabilisca una relazione lineare tra la velocità cartesiana del dispositivo terminale ($\mathbf{v}$) e le velocità dei giunti ($\mathbf{q}$). Si può utilizzare sempre la 3.3 anche per affrontare il problema dell'inversione cinematica. Dunque, si supponga di assegnare una traiettoria all'end effector e di stabilire anche la velocità che esso dovrà muoversi lungo tale traiettoria, l'obiettivo è quello di calcolare le posizioni e le velocità nello spazio dei giunti ($\mathbf{q}$ e $\dot{\mathbf{q}}$) in modo da riuscire a seguire la traiettoria desiderata con la velocità stabilita.
Ipotizzando che il numero di gradi di libertà del robot siano pari ai gradi di libertà richiesti per l'esecuzione del moto (ovvero lo Jacobiano sia una matrice quadrata), la cinematica inversa può essere risolta con la seguente forma della equazione 3.3:

$$\dot{\mathbf{q}} = \mathbf{J}^{-1}(\mathbf{q})\mathbf{v} \tag{3.9}$$

Partendo dalla relazione (3.9), nota la posizione iniziale nello spazio dei giunti $\mathbf{q}_0$, le posizioni possono essere ottenute per integrazione nel dominio del tempo, come riportato di seguito:

$$\mathbf{q}(t) = \int_0^t \dot{\mathbf{q}} dt + \mathbf{q}_0$$

L'integrazione può essere effettuata a tempo discreto, ricorrendo a metodi numerici, tra cui il più utilizzato è il metodo di Eulero, dove, fissato in intervallo di integrazione Δt e note le posizioni e velocità dei giunti all'istante di tempo t_k, le posizioni dei giunti all'istante $t_{k+1} = t_k + \Delta t$ sono calcolate come:

$$\mathbf{q}(t_{k+1}) = \mathbf{q}(t_k) + \dot{\mathbf{q}}(t_k)\Delta t \tag{3.10}$$

Il calcolo della cinematica inversa con questa metodologia ha come prerequisito la risolubilità della struttura cinematica, inoltre, al fine di poter applicare la (3.9), lo Jacobiano deve essere quadrato e di rango pieno. Perciò bisogna elaborare ulteriori strategie per le situazioni di singolarità (4.1) e manipolatori ridondanti (non è di interesse per questa trattazione).

4 Pianificazione moto e prevenzione di collisioni

In questo capitolo si riporta la teoria riguardante la strategia elaborata per pianificazione di moto e prevenzione di collisione. Si inizia introducendo il lettore ai Damped Least Squares, una strategia utilizzata per il calcolo della cinematica inversa in corrispondenza delle zone di singolarità [3]. Successivamente sono riportati le metodologie utilizzate nel capitolo 5 per l'elaborazione della traiettoria e prevenzione delle collisioni [2].

4.1 Damped Least-Squares

Come già accennato, l'equazione della cinematica inversa (3.9) può essere calcolata solo nell'ipotesi di rango pieno dello Jacobiano, perciò essa perde significato quando il manipolatore si trova in una configurazione singolare. L'inversione dello Jacobiano può costituire un problema significativo non solo in corrispondenza di una configurazione singolare, ma anche nell'intorno di essa dove si rischierebbe di perdere il controllo del robot in quanto nel calcolo dell'inversa bisogna passare per il calcolo del determinante di $\mathbf{J}$ che, nell'intorno di singolarità assume un valore relativamente piccolo, il che potrebbe portare a elevate velocità dei giunti. Una soluzione a tale problema è data dall'uso di una *inversa ai minimi quadrati* o **Damped Least Squares**:

$$\mathbf{J}^* = \mathbf{J}^\mathbf{T}(\mathbf{J}\mathbf{J}^\mathbf{T} + \lambda^2\mathbf{I})^{-1}$$

dove λ è un fattore di smorzamento che rende l'operazione di inversione meglio condizionata dal punto di vista numerico.
L'implementazione del Damped Least-Squares per lo studio della cinematica inversa in corrispondenza di una singolarità è basato sulla risoluzione della seguente equazione:

$$\mathbf{J}^\mathbf{T}\mathbf{v} = (\mathbf{J}^\mathbf{T}\mathbf{J} + \lambda^2\mathbf{I})\dot{\mathbf{q}} \tag{4.1}$$

Da tale equazione si ottiene:

$$\dot{\mathbf{q}} = (\mathbf{J}^\mathbf{T}\mathbf{J} + \lambda^2\mathbf{I})^{-1}\mathbf{J}^\mathbf{T}\mathbf{v} \tag{4.2}$$

Da notare che quando $\lambda = 0$, l'equazione (4.1) diventa equivalente a:

$$\mathbf{v} = \mathbf{J}\dot{\mathbf{q}}$$

e la Damped Least-Square si riduce all'inversione di una matrice vicino a una singolarità.
L'equazione (4.2) può essere ulteriormente elaborata nel seguente modo:

$$\dot{\mathbf{q}} = \mathbf{J}^*\dot{\mathbf{x}} + \mathbf{N}\dot{\mathbf{q}}_0 \tag{4.3}$$

dove $\mathbf{J}^*$ sarebbe il Damped Least Square, $\mathbf{N}$ sarebbe la matrice ortogonale ottenuta come proiezione ortogonale di $\mathbf{J}$ nello spazio *nullo*, mentre $\dot{\mathbf{q}}_0$ sarebbe la velocità nello spazio nullo che ha l'effetto di generare moto nei giunti interni del manipolatore, lasciando invariata la posizione del dispositivo terminale.
Perciò, nota la posizione nello spazio dei giunti ad un generico istante di tempo $\mathbf{q}(t)$, si può calcolare la posizione nel successivo istante di controllo, calcolata la velocità inversa con (4.3), tramite la seguente espressione già vista in (3.10)

$$\mathbf{q}(t + dt) = \mathbf{q}(t) + \dot{\mathbf{q}}dt \tag{4.4}$$

dove l'intervallo di tempo dt tra due successivi istanti di controllo, dipende dalla potenza di calcolo del controller del robot.

4.1.1 Definizione del fattore di smorzamento

Dunque è importante selezionare un opportuno valore del fattore di smorzamento λ [3]:

- bassi valori di λ forniscono soluzioni accurate, ma meno precise nelle configurazioni singolari e loro prossimità
- elevati valori di λ comportano bassa accuratezza di tracciamento, anche quando si ha una soluzione accurata

Una regione singolare può essere definita sulla stima del minore valore singolare di $\mathbf{J}$; fuori di tale regione viene utilizzata la soluzione esatta, mentre al suo interno viene usato un opportuno fattore di smorzamento che consente di ottenere una soluzione approssimata. Tale fattore di smorzamento deve essere scelto in modo da rispettare la continuità della velocità dei giunti $\dot{\mathbf{q}}$ durante la transizione ai confini della regione singolare. Una volta ottenuto il minore valore singolare (s_{min}), si può ricavare il valore di λ con la seguente formulazione:

$$\lambda = \begin{cases} 0, & \text{se } s_{min} > \epsilon, \\ (1 - (\frac{s_{min}}{\epsilon})^2) \cdot \lambda_{max}^2, & \text{se } s_{min} \leq \epsilon. \end{cases} \tag{4.5}$$

dove il valore λ_{max} è una scelta dell'utente per modellare la soluzione in prossimità della singolarità.

4.2 Correzione del feedback

Le precedenti soluzioni della cinematica inversa tendono ad essere soggette a deviazione numerica quando vengono implementate per intervalli discreti di tempo. Per evitare ciò il

termine $\mathbf{v}$ nell'equazione (4.2) viene sostituito con [1]:

$$\mathbf{v} = \mathbf{v_d} + \mathbf{Ke} \tag{4.6}$$

dove il pedice "*d*" sottolinea il fatto che ci si riferisca alla velocità dell'end effector, $\mathbf{K}$ è una matrice (6x6) positiva e diagonale, ed $\mathbf{e}$ esprime lo scostamento tra la posizione desiderata e quella attuale dell'end effector. L'errore $\mathbf{e}$ può essere rappresentato come:

$$\mathbf{e} = \begin{pmatrix} \mathbf{e_t} \\ \mathbf{e_0} \end{pmatrix} = \begin{pmatrix} \mathbf{p_d} - \mathbf{p} \\ \frac{1}{2}(\mathbf{n} \times \mathbf{n_d} + \mathbf{s} \times \mathbf{s_d} + \mathbf{a} \times \mathbf{a_d} \end{pmatrix} \tag{4.7}$$

dove l'errore di traslazione è dato dal vettore $\mathbf{e_t}$ (3x1), mentre l'errore di orientamento è dato dal vettore $\mathbf{e_0}$ (3x1); la matrice di rotazione è data da $\mathbf{R}$=[$\mathbf{n}$ $\mathbf{s}$ $\mathbf{a}$], dove $\mathbf{n}$, $\mathbf{s}$ e $\mathbf{a}$ sono i vettori che compongono la matrice di rotazione (3x3).

4.3 Strategia per evitare le collisioni

Il manipolatore, per portare a termine il proprio compito, deve spostare il dispositivo terminale dalla posizione iniziale O_i alla posizione finale O_f. In condizioni generali, il robot passa da una posizione all'altra seguendo il percorso più breve, ovvero la traiettoria rettilinea che collega O_i e O_f, ma si desidera esplicitare una via alternativa da seguire qualora non fosse possibile percorrere il tratto lineare per via di una imminente collisione con un ostacolo (potrebbe trattarsi di un oggetto fisico oppure di una persona), fisso o mobile, che il braccio robotico incontra seguendo tale tracciato. Ovviamente la collisione con l'ostacolo va evitato non solo con l'end effector connesso al manipolatore, bensì con ogni parte della sua struttura. Al fine di formulare la strategia per evitare le collisioni, bisogna innanzitutto definire due raggi, r e r_{min}, dove il primo è il raggio di controllo del robot, ovvero rappresenta una sfera virtuale all'interno della quale il robot è in grado di percepire eventuali oggetti esterni tramite i propri sensori, mentre r_{min} rappresenta il raggio minimo di sicurezza, vale a dire che se un corpo estraneo al robot dovesse avvicinarsi alla struttura dello stesso ad una distanza minore di r_{min}, il manipolatore si blocca immediatamente per garantire la sicurezza dell'oggetto in questione, questo in accordo con le normative per la sicurezza di strutture collaborative. Per una distanza intermedia ai due raggi, il robot deve modificare la propria traiettoria, in modo da portare a termine il proprio compito senza urtare contro l'oggetto rilevato.
La strategia di *Collsion Avoidance* [2] pensata per risolvere questa situazione consiste, appena rilevata la presenza di un oggetto esterno all'interno della zona di controllo, nel

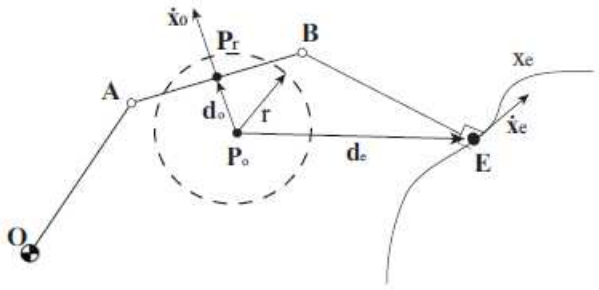


Figura 19: Componenti della velocità per l'end effector e il punto di controllo più vicino del robot

calcolare la distanza tra l'ostacolo e il punto della struttura del manipolatore più vicina all'oggetto in questione (il punto del robot più vicino al manipolatore può essere visto come *critico* per il collision avoidance). Ad esempio, nella figura 19, il punto P_0 rappresenta l'ostacolo, mentre P_r è il punto situato sulla struttura del manipolatore con la distanza minore rispetto a P_0. Perciò si può definire d_0 come:

$$d_0 = P_r - P_0$$

e se tale distanza dovesse risultare minore al raggio di controllo r, un vettore velocità repulsiva $\dot{x}_0$ viene applicato al robot nel punto P_r, come mostrato in figura. Tale vettore velocità repulsiva viene tende ad allontanare il punto critico dall'ostacolo. Inoltre un ulteriore vettore velocità, $\dot{x}_e$, viene applicato all'end effector al fine di riportarlo sulla traiettoria predefinita per portarlo nella posizione finale desiderata. Dunque sul robot agiscono due componenti di velocità, una repulsiva che tende ad allontanarlo da eventuali corpi esterni che vengono rilevati dai dispositivi sensoristici, mentre la seconda velocità è attrattiva e tende a riportare l'end effector sulla traiettoria rettilinea presente tra la posizione iniziale (O_i) e finale (O_f). Tali velocità possono essere calcolate con le seguenti relazioni:

$$\dot{\mathbf{x}}_e = \mathbf{J}\dot{\mathbf{q}} \tag{4.8}$$

$$\dot{\mathbf{x}}_0 = \mathbf{J}_0\dot{\mathbf{q}} \tag{4.9}$$

dove $\mathbf{J}_0$ in (4.9) è lo Jacobiano relativo al punto P_r.
Inserendo i valori di $\dot{\mathbf{x}}_e$ e $\dot{\mathbf{x}}_0$ nell'equazione (4.3), si ottiene:

$$\dot{\mathbf{q}} = \mathbf{J}^*\dot{\mathbf{x}}_c + (\mathbf{J}_0\mathbf{N})^*(\dot{\mathbf{x}}_0 - \mathbf{J}_0\mathbf{J}^*\dot{\mathbf{x}}_e) \tag{4.10}$$

dove il termine $\dot{\mathbf{x}}_c = \dot{\mathbf{x}}_e + \mathbf{K}e$ sarebbe la velocità dell'end effector corretta dello scostamento tra la posizione desiderata e quella attuale dello stesso, definita più nello specifico in (4.6).

Il primo termine dell'equazione (4.10) contribuisce a muovere il dispositivo terminale alla velocità desiderata, con la minima velocità dei giunti; il secondo termine guida il punto P_r del robot in base al moto repulsivo dall'ostacolo.
L'equazione (4.10) può essere ulteriormente semplificata, in quanto secondo i ragionamenti appena fatti la strategia di *Collision avoidance* applicata richiede soltanto il moto nella direzione che collega il punto critico del robot con l'ostacolo e ciò richiede un solo grado di libertà del manipolatore. Dunque è possibile semplificare la risoluzione della (4.10) e avere minori singolarità, sostituendovi lo jacobiano $\mathbf{J}_0$ con:

$$\mathbf{J}_{d0} = \hat{\mathbf{d}}^T \mathbf{J}_0 \tag{4.11}$$

dove $\hat{\mathbf{d}} = \mathbf{d}_0/||\mathbf{d}_0||$ sarebbe la direzione lungo la quale viene applicata la velocità repulsiva. Le velocità $\dot{\mathbf{x}}_0$ e il termine $\mathbf{J}_{d0}\mathbf{J}^*\dot{\mathbf{x}}_e$ diventano così dei valori scalari, perciò l'equazione (4.10) può essere riscritta come:

$$\dot{\mathbf{q}} = \mathbf{J}^*\dot{\mathbf{x}}_c + \mathbf{N}\mathbf{J}_{d0}^T(\mathbf{J}_{d0}\mathbf{N}\mathbf{J}_{d0}^T)^{-1}(\dot{x}_0 - \mathbf{J}_{d0}\mathbf{J}^*\dot{\mathbf{x}}_e) \tag{4.12}$$

La scelta della velocità repulsiva è fondamentale in questa fase dell'algoritmo, e la strategia adottata propone di variare il valore assoluto di $\dot{\mathbf{x}}_0$ in funzione della distanza dall'ostacolo. Per evitare fenomeni di discontinuità e dare scorrevolezza al moto, sono stati adottati tre fattori di guadagno a_h, a_v e a_e, in modo che:

$$\dot{x}_0 = a_v v_{rep} \tag{4.13}$$

Perciò sostituendo la (4.13) in (4.12) si ottiene:

$$\dot{\mathbf{q}} = \mathbf{J}^*(\dot{\mathbf{x}}_c + a_e v_{rep}\hat{\mathbf{d}}_e) + a_h\mathbf{N}\mathbf{J}_{d0}^T(\mathbf{J}_{d0}\mathbf{N}\mathbf{J}_{d0}^T)^{-1}(a_v v_{rep} - \mathbf{J}_{d0}\mathbf{J}^*\dot{\mathbf{x}}_e) \tag{4.14}$$

Nella precedente equazione, il modulo della velocità repulsiva v_{rep} è definita dal coefficiente a_v in funzione della distanza d_0 dall'ostacolo, mentre il fattore a_h agisce come peso per bilanciare l'effetto della soluzione omogenea relativa al collision avoidance. Inoltre, il termine $a_e v_{rep}\hat{\mathbf{d}}_e$ è stato aggiunto nella (4.14) alla velocità applicata al dispositivo terminale in modo da tenere in considerazione la possibilità di collisione con un ostacolo che si sposta dalla sua posizione iniziale. I tre coefficienti a_h, a_v, a_e sono mostrati graficamente nella figura 20 in funzione della distanza d_0 tra l'ostacolo e il punto critico del robot. Innanzitutto, sull'asse delle ascisse di tale figura si possono distinguere i raggi r, r_{min} e r_m, ovvero, rispettivamente, il raggio massimo di controllo dei sensori del robot, il raggio minimo sotto al quale si ha l'interruzione di moto (per evitare l'urto con l'oggetto rilevato), e un raggio intermedio tra i

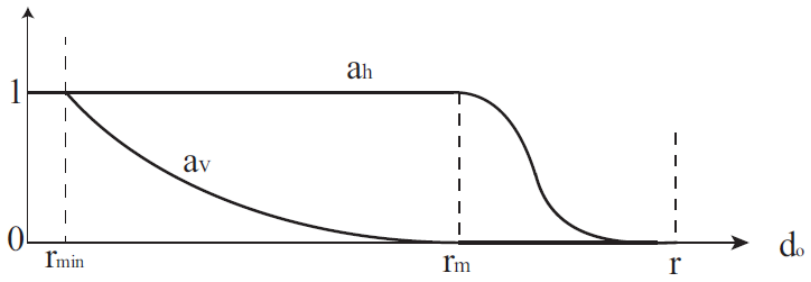


Figura 20: Parametri a_h e a_v in funzione della distanza d_0

due, che rappresenta una distanza critica per il funzionamento del robot. Per una distanza d_0 maggiore di r, i coefficienti tendono tutti e tre al valore nullo, in quanto il robot non rileva nessun ostacolo e segue la traiettoria rettilinea che ne collega la posizione iniziale e finale; per una distanza minore di r_{min}, i parametri assumono un valore unitario e si ha il blocco totale del robot. Per valori di d_0 compresi tra r_{min} e r_m, i parametri assumono valori espressi dai sistemi di equazioni:

$$a_v = \begin{cases} (\frac{d_0 - r_m}{r_{min} - r_m})^2, & se\ d_0 < r_m \\ 0, & se\ d_0 \leq r_m \end{cases} \tag{4.15}$$

$$a_h = \begin{cases} 1, & se\ d_0 \geq r_m \\ \frac{1}{2}[1 - cos(\pi \frac{d_0 - r_m}{r - r_m})], & se\ r_m < d_0 < r \\ 0, & se\ d_0 \leq r \end{cases} \tag{4.16}$$

Il coefficiente a_e viene definito in modo analogo a a_v, sostituendo in (4.15) d_e al posto di d_0.

4.4 Pianificazione della traiettoria

Oltre alla strategia per evitare la collisione con ostacoli, un algoritmo per la pianificazione della traiettoria è richiesto innanzitutto per definire il moto $\mathbf{x}_e(t)$ da assegnare al dispositivo terminale, in modo che questo possa raggiungere la posizione designata ed evitare gli ostacoli nella loro posizione iniziale. Per fare ciò si adotta un metodo basato sull'applicazione di campi potenziali artificiali, basato su una tecnica utilizzata tipicamente per calcolare la traiettoria di robot mobili.

Un campo potenziale agisce sull'end effector con delle forze virtuali, associabili alle componenti di velocità responsabili di condurre il dispositivo terminale lungo la traiettoria a minimo potenziale verso la posizione target. Riferendosi alla figura 21, **P** è la posizione iniziale

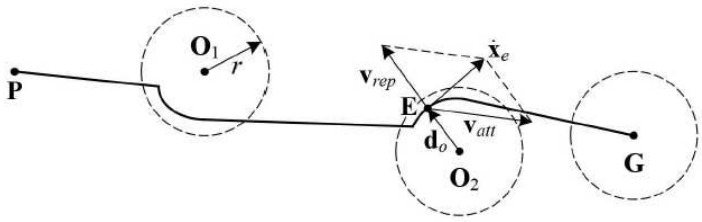


Figura 21: Campi potenziali per la pianificazione della traiettoria

dell'end effector, **E** ne è la posizione attuale e **G** sarebbe la posizione finale, mentre $\mathbf{O}_1$ e $\mathbf{O}_2$ sono due ostacoli (r sarebbe il raggio delle sfere entro le quali il robot è in grado di rilevare oggetti esterni, in questa parte si suppone per semplicità che tali sfere siano poste sugli ostacoli piuttosto che sul robot, ma ciò non comporta nessuna variazione funzionale nell'algoritmo). Si possono definire le distanze $\mathbf{d}_O = \mathbf{E} - \mathbf{O}$, ovvero la distanza tra l'end effector e l'ostacolo più vicino, e $\mathbf{d}_G = \mathbf{G} - \mathbf{E}$ che sarebbe la distanza tra la posizione finale dell'end effector e quella attuale. Le forze attrattive e repulsive agenti su **E** si possono definire secondo i seguenti sistemi di equazioni:

$$\mathbf{v}_{att} = \begin{cases} v_{att}\frac{\mathbf{d}_G}{r}, & se\ d_G < r \\ v_{att}\hat{\mathbf{d}}_G, & se\ d_G \leq r \end{cases} \tag{4.17}$$

$$\mathbf{v_{rep}} = \begin{cases} \frac{v_{rep}}{\mathbf{d}_O}(\frac{1}{\mathbf{d}_O} - \frac{1}{r})\nabla\mathbf{d}_O, & se\ d_O < r \\ 0, & se\ d_O \leq r \end{cases} \tag{4.18}$$

dove $\nabla\mathbf{d}_O$ rappresenta il gradiente della distanza d_O in funzione delle coordinate cartesiane.

5 Algoritmi

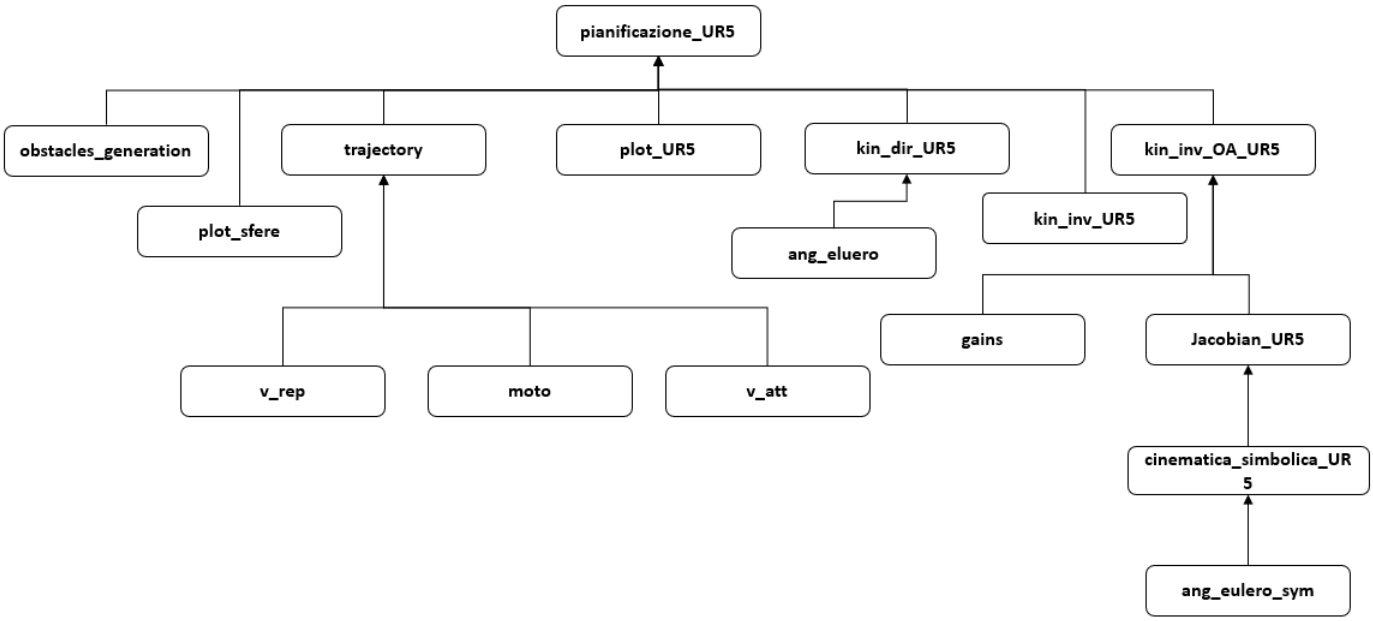


Figura 22: Schema delle funzioni implementate su MATLAB

5.1 Pianificazione del moto

```
close all
clear
clc

global X_i X_f dt T r r_min...
       v0_rep v0_att k_e Q_i O dO k_v lambda_max eps...
       dQ_max  UR5 ik weights ...

dQ_max=5;
r=0.12;
r_min=0.09;
v0_rep=10;
v0_att=1;
k_e=100;
```

```
T=1;
steps=1000;
dt=T/steps;
k_v=100;
order_Bezier_curve = 3;
lambda_max=10^-1;
eps=10^-1;
UR5 = importrobot('ur5_robot.urdf');
UR5.DataFormat='row';
ik = robotics.InverseKinematics('RigidBodyTree', UR5);
weights = [1, 1, 1, 1, 1, 1];
X_i=[0.8173;0.1842;-0.0054;0;0;0];
X_f=[0.4173;0.1842;0.856;0;0;0];
Oi=[0.68 0.6; 0.38 0.12; 0.3 0.8];
Of=[0.68 0.6; 0 0.17; 0.3 0.4];
dOi=[0 0; 0 0; 0 0];
eul_i=X_i(4:6);
eul_f=X_f(4:6);
R_i=eul2rotm(eul_i','ZYZ');
R_f=eul2rotm(eul_f','ZYZ');
X_model_i=X_i;
X_model_f=X_f;
X_model_i(4:6)=rotm2eul(R_i,'ZYZ');
X_model_f(4:6)=rotm2eul(R_f,'ZYZ');
%%%%%%%%%%%%%%%%%%%
initialguess=UR5.homeConfiguration;
t_form=getTransform(UR5, initialguess,'ee_link','base_link');
Q_i = ik('ee_link',t_form,weights,initialguess);
Q_i=Q_i';
pose=[eul2rotm(X_model_f(4:6)','ZYZ') X_model_f(1:3); 0 0 0 1];
Q_f = ik('ee_link',pose,weights,initialguess);
Q_f=Q_f';
[O,dO]=obstacles_generation(Oi,Of,dOi,dt,T);
tic
[t,X,dX,X_bez,dX_bez]=trajectory(order_Bezier_curve);
toc
%cinematica inversa
[Q]=kin_inv_OA_UR5(t,X_bez,dX_bez,O,dO);
% [Q]=kin_inv_UR5(X,dX,Q_i,dt);

%scrittura video
```

```
vid = VideoWriter('video5.avi');
open(vid);
figure(2)
view(135,45)
camva(9);
cameratoolbar('Toggle')
axis equal

for i=1:10:size(Q,2)
cla
[X_test,punti]=kin_dir_UR5_3(Q(:,i));
%subplot(1,2,2)
hold on
plot3(X(1,:),X(2,:),X(3,:),'g')
plot3(X_bez(1,:),X_bez(2,:),X_bez(3,:),'m')
plot_UR5(Q(:,i),r_min);
plot_sfere(O(:,:,i), r_min/4)
P=X_i(1:3,1);
eul=X_i(4:6,1);
R= eul2rotm(eul','ZYZ');
quiver3(P(1),P(2),P(3),R(1,1),R(2,1),R(3,1),0.2,'r')
quiver3(P(1),P(2),P(3),R(1,2),R(2,2),R(3,2),0.2,'g')
quiver3(P(1),P(2),P(3),R(1,3),R(2,3),R(3,3),0.2,'b')
P=X_f(1:3,1);
eul=X_f(4:6,1);
R= eul2rotm(eul','ZYZ');
quiver3(P(1),P(2),P(3),R(1,1),R(2,1),R(3,1),0.2,'r')
quiver3(P(1),P(2),P(3),R(1,2),R(2,2),R(3,2),0.2,'g')
quiver3(P(1),P(2),P(3),R(1,3),R(2,3),R(3,3),0.2,'b')
axis equal
%F(i)= getframe;
frame = getframe(gcf);
   writeVideo(vid,frame);
end

close(vid);
%movie(F)

figure(4)
plot(Q(1,:));hold on
plot(Q(2,:))
```

```
plot(Q(3,:))
plot(Q(4,:))
plot(Q(5,:))
plot(Q(6,:))

figure(5)
plot(diff(Q(1,:))./dt);hold on
plot(diff(Q(2,:))./dt);
plot(diff(Q(3,:))./dt);
plot(diff(Q(4,:))./dt);
plot(diff(Q(5,:))./dt);
plot(diff(Q(6,:))./dt);
```

La funzione riportata mostra il moto del robot lungo una certa traiettoria (può trattarsi di una traiettoria preimpostata o generata in maniera random dal programma); inizialmente viene definito valore della velocità massima di rotazione dei giunti (dq_{max}), il raggio delle sfere di controllo r, il raggio minimo al di sotto del quale la simulazione fallisce r_{min}, le velocità iniziali di attrazione e repulsione (v_{0att} e v_{0rep}), i coefficienti k_e e k_i, l'intervallo dei tempi di controllo, l'ordine delle curve di Bezier (verrà inserito come input per la funzione *trajectory*, riportata nel capitolo 5.4), λ_{max} (per il calcolo del fattore di smorzamento del Damped Least-squares (4.5)) e ϵ, ovvero il valore con cui confrontare il minimo valore singolare, come riportato nell'equazione (4.5). Inizialmente la funzione si importa il modello CAD del robot dal file *urdf* e ne viene definito un resolver per il calcolo della cinematica inversa utilizzando il comando "InverseKinematics" presente nel toolboox *Robotics* di MATLAB. Il passo successivo consiste nel definire il punto iniziale e quello finale che il manipolatore deve raggiungere con il suo end effector, la posizione iniziale e finale degli ostacoli e la loro velocità iniziale (posta nulla lungo tutte le direzioni). In seguito, la funzione esegue la cinematica inversa di orientamento, convertendo gli angoli di Eulero contenuti negli ultimi tre termini dei vettori posizione iniziale e finale (X_i e X_f, rispettivamente) usando il comando "eul2rotm" del programma, ottenendo così la matrice di rotazione del manipolatore nella configurazione iniziale (come configurazione iniziale è stata impostata quella presente nel file *urdf*, ovvero il robot si trova in posizione perfettamente orizzontale, questo in quanto qualunque variazione da tale posizione comporterebbe uno scostamento tra la struttura studiata cinematicamente e quella presente nel file CAD, con la quale verranno eseguite le successive simulazioni) e configurazione finale. Il passo successivo consiste nel ricavare le posizioni e le velocità degli ostacoli utilizzando la funzione *obstacle_ generation* (capitolo

5.5), e la traiettoria e la velocità dell'end effector (nello spazio cartesiano) utilizzando la funzione *trajectory* (capitolo 5.4). Da tali valori si ricava la cinematica inversa, ovvero la posizione nello spazio dei giunti (**q**) utilizzando la funzione *kin_inv_OA_UR5* (capitolo 5.3), la quale tiene conto anche delle forze attrattive e repulsive dovute alla presenza degli ostacoli. In seguito, vengono ricavate le posizioni dei vari punti di controllo dalla funzione della cinematica diretta *kin_dir_UR5* (capitolo 5.6) e in tali punti vengono visualizzate le sfere di controllo; viene anche visualizzata la traiettoria rettilinea che il robot deve seguire in assenza di ostacoli e vengono posizionate due terne nella posizione iniziale e finale che l'end effector deve raggiungere (tali terne mostrano anche l'orientamento che l'end effector deve avere una volta arrivato nel punto che deve raggiungere). L'ultimo passo è quello di mostrare due grafici contenenti l'andamento della posizione e velocità dei vari giunti nel tempo di simulazione, e ciò viene effettuato utilizzando il comando *plot* di MATLAB.

5.2 Cinematica inversa del manipolatore

```
function Q = kin_inv_UR5 (X,dX,Q_i,dt)
global k_e lambda_max eps

n=size(X,2);
Q(:,1)=Q_i;
for i=1:n-1
    X_ac=kin_dir_UR5_3(Q(:,i));
    e(1:3,1)=X(1:3,i)-X_ac(1:3);
    eul_pl=X(4:6,i);
    eul_ac=X_ac(4:6);
    R_pl=eul2rotm(eul_pl','ZYZ');
    R_ac=eul2rotm(eul_ac','ZYZ');
    e(4:6,1)=(0.5*cross(R_ac(:,1),R_pl(:,1))+...
    cross(R_ac(:,2),R_pl(:,2))+...
    cross(R_ac(:,3),R_pl(:,3)))/(2*pi);

    J=Jacobian_UR5_2(Q(:,i),13);
    %[U,S,V] = svd(J);
    %s_min=S(6,6);
    S=svd(J);
```

```
    s_min=S(6);
        if s_min<eps
    lambda=(1-(s_min/eps)^2)*lambda_max^2;
    else
        lambda=0;
    end
    pinv_JD=J'*inv(J*J'+lambda^2*eye(6));
    Q(:,i+1)=Q(:,i)+dt*(pinv_JD*(dX(:,i)+k_e*e));


end
end
```

La funzione richiama in ingresso il vettore $\mathbf{Q_i}$ delle posizioni iniziali dei giunti ed esegue un ciclo *for* facendo variare un parametro i tra il valore 1 e la dimensione del vettore X delle posizioni nello spazio cartesiano. Dalla funzione della cinematica diretta *kin_dir_UR5* (capitolo 5.6) richiama $\mathbf{X_{ac}}$, ovvero un vettore che contiene dati sulla posizione e orientamento dell'end effector nella configurazione finale. Si definisce con il parametro **e** la differenza tra di posizione (ovvero solo le prime tre righe dei due vettori) tra la posizione desiderata dell'end effector e quella effettiva. Successivamente, dalle ultime tre righe dei vettori posizione, utilizzando il comando *eul2rotm* di MATLAB, vengono ricavate le matrici di rotazione $\mathbf{R}_{pl}$ e $\mathbf{R}_{ac}$ relative all'orientamento desiderato dell'end effector e l'orientamento attuale; le ultime tre righe del vettore **e** vengono calcolate come prodotto vettoriale delle colonne delle matrici $\mathbf{R}_{ac}$ e $\mathbf{R}_{pl}$ (che, essendo matrici di rotazione, hanno entrambe dimensione 3x3). Successivamente, dalla funzione *Jacobian_UR5* contenete i Jacobiani di tutti i punti di controllo del manipolatore, si ricava lo Jacobiano relativo al punto di controllo situato in prossimità dell'end effector, fornendo come input anche la i-esima colonna della matrice **Q**. Si utilizza il comando *svd* di MATLAB per calcolare i valori singolari della matrice jacobiana e generare un vettore **S** che contenga tali valori disposti in ordine decrescente, perciò l'ultimo termine di **S** conterrà il minimo valore singolare (s_{min}). Paragonando tale valore minimo con la costante ϵ, si può ottenere il valore di λ, come da equazione (4.5). A questo punto la velocità del giunto i-esimo può essere calcolato applicando la Damped Least-Squares:

$$\mathbf{J}^* = \mathbf{J}^T(\mathbf{J}^T\mathbf{J} + \lambda^2\mathbf{I})^{-1}$$

Perciò la configurazione dei giunti all'istante i+1 può essere calcolata con la seguente equazione, in accordo con la (3.10):

$$q(i+1) = q(i) + dt(\mathbf{J}^* \cdot (v + k_e e))$$

dove $k_e\mathbf{e}$ è un termine correttivo dato dalla differenza tra la posizione desiderata dell'end effector e quella reale (4.6).

5.3 Cinematica inversa in presenza di ostacoli

```
function [Q]=kin_inv_OA_UR5(t,X,dX,O,dO)
global  dt  r r_min v0_rep k_e Q_i k_v lambda_max eps dQ_max
tolleranza=0.90;
Q(:,1)=Q_i;

for i=1:size(t,2)-1
    o=O(:,:,i);
    V_o=dO(:,:,i);
    X_pl=X(:,i);
    dt=t(i+1)-t(i);
    J=Jacobian_UR5_2(Q(:,i),13);
    S=svd(J);
    s_min=S(6);
        if s_min<eps
    lambda=(1-(s_min/eps)^2)*lambda_max^2;
    else
        lambda=0;
    end
    pinvd_J=J'*inv(J*J'+lambda^2*eye(6));
    J_par=J(1:3,:);
    S=svd(J_par);
    s_min=S(3);
        if s_min<eps
    lambda=(1-(s_min/eps)^2)*lambda_max^2;
    else
        lambda=0;
    end
    pinvd_J_par=J_par'*inv(J_par*J_par'+lambda^2*eye(3));
    [X_ac,punti_robot]=kin_dir_UR5_3(Q(:,i));
    punti_robot=punti_robot(1:3,:);
```

```
[k,dist] = dsearchn(o',punti_robot');

d_ee=dist(13);
P_o_ee=o(:,k(13));
v_o_ee=V_o(:,k(13));
P_r_ee=punti_robot(:,13);
k=k(4:12);
dist=dist(4:12);
[d_min,index]=min(dist);
if  or (d_ee<tolleranza*r_min,d_min<tolleranza*r_min)
    disp 'fail'
end
P_r=punti_robot(:,index+3);
P_o=o(:,k(index));
v_o=V_o(:,k(index));
[a_v, a_h, a_e]=gains(d_min,d_ee,r,r_min);
J0=Jacobian_UR5_2(Q(:,i),index+3);

vettore=(P_r-P_o-k_v*v_o*dt);
d0=vettore/norm(vettore);

N=eye(6)-pinvd_J*J;
J0_par=J0(1:3,:);
A=J0_par*N;
S=svd(A);
s_min=S(3);
if s_min<eps
    lambda=(1-(s_min/eps)^2)*lambda_max^2;
else
    lambda=0;
end
pinvd_A=A'*inv(A*A'+lambda^2*eye(3));

e(1:3,1)=X_pl(1:3)-X_ac(1:3);
eul_pl=X_pl(4:6);
eul_ac=X_ac(4:6);
R_pl=eul2rotm(eul_pl','ZYZ');
R_ac=eul2rotm(eul_ac','ZYZ');
e(4:6,1)=(0.5*cross(R_ac(:,1),R_pl(:,1))+...
cross(R_ac(:,2),R_pl(:,2))+...
cross(R_ac(:,3),R_pl(:,3)))/(2*pi);
```

```
    vettore=(P_r_ee-P_o_ee-k_v*v_o_ee*dt);
    d_rep=vettore/norm(vettore);
    dX_rep=a_e*v0_rep*d_rep;
    dQ=(pinvd_J*(dX(:,i)+k_e*e)+pinvd_J_par*dX_rep...
        +a_h*pinvd_A*(a_v*v0_rep*d0-J0_par*pinvd_J*dX(:,i)));
for j=1:6
    if abs(dQ(j))>dQ_max
        dQ(j)=dQ_max*sign(dQ(j));
    end
end
    Q(:,i+1)=Q(:,i)+dt*dQ;
end

end
```

Questa funzione richiede come input il vettore tempo t (fornito dalla funzione *trajectory*), i vettori posizione e la velocità dell'end effector ($\mathbf{X}$ e $\mathbf{dX}$), e i vettori posizione e la velocità degli ostacoli ($\mathbf{O}$ e $\mathbf{dO}$).
Si inizia con un ciclo *for* facendo variare un parametro i tra il valore 1 e $t-1$; si ricava l'intervallo di tempo dt come la differenza tra il tempo all'istante $i+1$ e il tempo all'istante i ($dt = t(i+1) - t(i)$); dalla funzione *Jacobian_UR5* di ricava lo jacobiano dell'end effector inserendovi la i-esima riga della matrice $\mathbf{Q}$, poi, con il comando *svd* di MATLAB si ricavano i valori singolari del Jacobiano e si genera un vettore $\mathbf{S}$ che contenga tali valori disposti in ordine decrescente. Perciò s_{min} sarà il valore singolare più piccolo e sarà contenuto nell'ultimo termine del vettore $\mathbf{S}$. Successivamente si calcola il parametro λ tramite il sistema di equazioni (4.5). Fino a qui il procedimento è analogo a quanto riportato nel capitolo 5.2. A questo punto è possibile ricavare la Damped Least-Square associato allo Jacobiano $\mathbf{J}$ come:

$$\mathbf{J}^* = \mathbf{J}^T(\mathbf{J}^T\mathbf{J} + \lambda^2\mathbf{I})^{-1} \tag{5.1}$$

Dallo Jacobiano $\mathbf{J}$ si estraggono le prime tre righe da cui viene generata la matrice $\mathbf{J_{par}}$, per la quale il procedimento precedente viene ripetuto fino ad ottenere la Damped Least-Square relativa a tale matrice. In seguito, dalla funzione della cinematica diretta *kin_dir_UR5* (capitolo 5.6), si ottengono i vettori associati ai vari punti di controllo del manipolatore e quello dell'end effector ($\mathbf{X}$). Usando il comando *dsearchn* di MATLAB, si ricavano il vettore $\mathbf{k}$ e il parametro *dist*, dove la prima contiene le distanze tra l'ostacolo e i vari punti di controllo del robot e *dist* sarebbe la minima tra tali distanze; successivamente si ricava la distanza, posizione e velocità dell'ostacolo rispetto all'end effector. Viene imposto che se la distanza minima (*dist*) tra il robot e l'ostacolo dovesse risultare inferiore al raggio minimo

r_{min} (vedi capitolo 4.3) moltiplicato per una certa tolleranza, il cui valore viene definito a priori, allora il programma restituisce l'avvertimento di *fail*, in quanto non viene rispettata la condizione di sicurezza preimpostata riguardo alla distanza minima tra robot e l'ostacolo. Si introduce un indice (*index*) il cui valore dipende da quale dei vari punti di controllo si trova più vicino all'ostacolo, quindi si ricavano la posizione e la velocità dell'ostacolo in corrispondenza del punto indicato da tale indice; dalla funzione *gains* si ricavano i valori dei parametri a_v, a_h e a_e, il cui significato è spiegato specificamente nel capitolo 4.3. $\mathbf{J_0}$ è lo Jacobiano corrispondente al punto del manipolatore che si trova più vicino all'ostacolo; poi viene generato un vettore per avere una componente di velocità in senso opposto a v_0 (velocità dell'ostacolo) e d_0 è il versore associato a tale vettore. Successivamente viene generata una matrice N (già vista nell'equazione 4.3) come:

$$\mathbf{N} = \mathbf{I} - \mathbf{J}^*\mathbf{J}$$

Tale matrice viene utilizzata per generare una matrice $\mathbf{A}$ data da:

$$\mathbf{A} = \mathbf{J_{0par}}\mathbf{N}$$

dove $\mathbf{J_{0par}}$ è la matrice contenente le prime tre righe dello jacobiano $\mathbf{J_0}$. Il passo successivo consiste nel calcolarsi il Damped Least Square della matrice $\mathbf{A}$ con il medesimo procedimento visto nella (5.1).
Poi si introduce un vettore $\mathbf{e}$, che serve per definire l'errore di posizione dell'end effector. I primi 3 termini di $\mathbf{e}$ sono dati dalla differenza tra la posizione desiderata dell'end effector e quella reale, mentre gli ultimi tre termini possono essere calcolati come il prodotto vettoriale tra le colonne delle matrici di rotazione relative alla posizione iniziale e finale dell'end effector, in accordo con la seguente definizione di tale vettore:

$$\mathbf{e} = \begin{bmatrix} \mathbf{e_t} \\ \mathbf{e_0} \end{bmatrix} = \begin{bmatrix} \mathbf{p_d} - \mathbf{p} \\ \frac{1}{2}(\mathbf{n} \times \mathbf{n_d} + \mathbf{s} \times \mathbf{s_d} + \mathbf{a} \times \mathbf{a_d} \end{bmatrix}$$

dove l'errore di traslazione è dato dal vettore $\mathbf{e_t}$ (3x1), mentre l'errore di orientamento è dato dal vettore $\mathbf{e_0}$ (3x1); la matrice di rotazione è data da $\mathbf{R}$=[$\mathbf{n}$ $\mathbf{s}$ $\mathbf{a}$], dove $\mathbf{n}$, $\mathbf{s}$ e $\mathbf{a}$ sono i vettori che compongono tale matrice.
A questo punto la velocità repulsiva agente sull'end effector viene calcolata come:

$$\dot{x}_{rep} = a_e \cdot v_{0rep} \cdot d_{rep}$$

da cui si può calcolare la velocità nello spazio dei giunti come:

$$\dot{\mathbf{q}} = \mathbf{J}^*(\dot{\mathbf{x}}(i) + k_e\mathbf{e}) + \mathbf{J^*_{par}}\dot{x}_{rep} + a_h\mathbf{A}^*(a_v v_{0rep} d_0 - \mathbf{J_{0par}}\mathbf{J}^*\dot{\mathbf{x}}(i))$$

Da notare che la precedente è equazione è uguale alla (4.14), vista nel capitolo 4.3.
Se la velocità di un qualunque giunto del sistema tende a superare la velocità massima imposta ($\dot{\mathbf{q}}_{\mathbf{max}}$), in tal caso la velocità del giunto considerato viene posta pari a tale valore massimo.
Dunque, lo spostamento nello spazio dei giunti nell'istante successivo all'attuale può essere calcolato come nella (3.10), ovvero:

$$\mathbf{q}(i+1) = \mathbf{q}(i) + \dot{\mathbf{q}}dt$$

e il valore $\mathbf{q}(i+1)$ sarà l'output di questa funzione.

5.4 Calcolo della traiettoria

```
function [t,X,dX,X_bez,dX_bez]=trajectory(order_Bezier_curve)
    global X_i X_f dt T r r_min v0_rep v0_att k_e Q_i O

    x_i=X_i(1:3);
    x_f=X_f(1:3);
    d_O=r;
    d_G=r;
    soglia=10^-5;
    Oi=O(:,:,1);

    X_prel1(:,1)=x_i;
    X_prel2(:,1)=x_i;
    X_prel3(:,1)=x_i;
    X_prel4(:,1)=x_i;

    dX_prel1(:,1)=zeros(3,1);
    dX_prel2(:,1)=zeros(3,1);
    dX_prel3(:,1)=zeros(3,1);
    dX_prel4(:,1)=zeros(3,1);

    s_prel1(1)=0;
    s_prel2(1)=0;
    s_prel3(1)=0;
    s_prel4(1)=0;
```

```
for m=1:1:4
    X_exit=x_i;
    while norm(X_exit-x_f)>soglia
        switch m    %m va da 1 a 4 e sceglie il verso dei casi
            case 1
                v_a=v_att(X_prel1(:,end),x_f,d_G,v0_att); %v_attrattiva
                v_r=v_rep(X_prel1(:,end),Oi,d_O,v0_rep);  % v_repulsiva
                v=v_a+v_r;                                % somma delle v
                if v_r/norm(v_r)+v_a/norm(v_a)==0
                    vers_v=v/norm(v); %aumenta di una dimensione
                    vers_v_ort = null(vers_v(:).');
                    v=v+(-1)^m*v0_rep*vers_v_ort(:,1);
                end
                X_new1=X_prel1(:,end)+dt*v;
                X_prel1=[X_prel1 X_new1];
                s_new1=s_prel1(:,end)+norm(dt*v);
                s_prel1=[s_prel1 s_new1];
                X_exit=X_new1;
                dX_prel1=[dX_prel1 dt*v];

            case 2
                v_a=v_att(X_prel2(:,end),x_f,d_G,v0_att); %v_attrattiva
                v_r=v_rep(X_prel2(:,end),Oi,d_O,v0_rep);  % v_repulsiva
                v=v_a+v_r;                                % somma delle v
                if v_r/norm(v_r)+v_a/norm(v_a)==0
                    vers_v=v/norm(v); %aumenta di una dimensione
                    vers_v_ort = null(vers_v(:).');
                    v=v+(-1)^m*v0_rep*vers_v_ort(:,2); %[-vers_v(2); vers_v(1)];
                end
                X_new2=X_prel2(:,end)+dt*v;
                X_prel2=[X_prel2 X_new2];
                s_new2=s_prel2(:,end)+norm(dt*v);
                s_prel2=[s_prel2 s_new2];
                X_exit=X_new2;
                dX_prel2=[dX_prel2 dt*v];

            case 3
                v_a=v_att(X_prel3(:,end),x_f,d_G,v0_att);
                v_r=v_rep(X_prel3(:,end),Oi,d_O,v0_rep);
                v=v_a+v_r;
```

```
                if v_r/norm(v_r)+v_a/norm(v_a)==0
                    vers_v=v/norm(v);
                    vers_v_ort = null(vers_v(:).');
                    v=v-(-1)^m*v0_rep*vers_v_ort(:,1);
                end
                X_new3=X_prel3(:,end)+dt*v;
                X_prel3=[X_prel3 X_new3];
                s_new3=s_prel3(:,end)+norm(dt*v);
                s_prel3=[s_prel3 s_new3];
                X_exit=X_new3;
                dX_prel3=[dX_prel3 dt*v];

            case 4
                v_a=v_att(X_prel4(:,end),x_f,d_G,v0_att);
                v_r=v_rep(X_prel4(:,end),Oi,d_O,v0_rep);
                v=v_a+v_r;
                if v_r/norm(v_r)+v_a/norm(v_a)==0
                    vers_v=v/norm(v);
                    vers_v_ort = null(vers_v(:).');
                    v=v-(-1)^m*v0_rep*vers_v_ort(:,2);
                end
                X_new4=X_prel4(:,end)+dt*v;
                X_prel4=[X_prel4 X_new4];
                s_new4=s_prel4(:,end)+norm(dt*v);
                s_prel4=[s_prel4 s_new4];
                X_exit=X_new4;
                dX_prel4=[dX_prel4 dt*v];

        end
    end

end

scelta = [s_prel1(end) s_prel2(end) s_prel3(end) s_prel4(end)];
[m, ind] = min(scelta);
switch ind
    case 1
        X_prel=X_prel1;
        dX_prel=dX_prel1;
        s_prel=s_prel1;
```

```
    case 2
        X_prel=X_prel2;
        dX_prel=dX_prel2;
        s_prel=s_prel2;

    case 3
        X_prel=X_prel3;
        dX_prel=dX_prel3;
        s_prel=s_prel3;

    case 4
        X_prel=X_prel4;
        dX_prel=dX_prel4;
        s_prel=s_prel4;
end

s_prel=s_prel/s_prel(end);
out = moto(0,0,0,1,0,0,0,1);
t=out(:,1)';
s=out(:,2)';
ds=out(:,3)';
X(1,:)=interp1(s_prel,X_prel(1,:),s);
X(2,:)=interp1(s_prel,X_prel(2,:),s);
X(3,:)=interp1(s_prel,X_prel(3,:),s);  %aggiunta questa X

dX(1,:)=interp1(s_prel,dX_prel(1,:),s);
dX(2,:)=interp1(s_prel,dX_prel(2,:),s);
dX(3,:)=interp1(s_prel,dX_prel(3,:),s); %aggiunta questa dX

%%%% Interpolazione con Bezier

switch order_Bezier_curve
    case 3
        for i=1:size(s,2)
            S1(i,:)=[(1-s(i))^3  (s(i))^3];
            S2(i,:)=[3*s(i)*(1-s(i))^2  3*(s(i))^2*(1-s(i))];
            N(i,:)=[X(1,i) X(2,i) X(3,i)];
        end
        C1=[x_i';x_f'];
        C2=pinv(S2)*(N-S1*C1);
        coeff=[C1; C2]';
```

```
    X_A = coeff(:,1);
    X_B = coeff(:,2);
    X_A1 = coeff(:,3);
    X_B1 = coeff(:,4);
    V_A = 0;
    V_B = 0;
    X_bez = X_A.*((1-s).^3) +3*X_A1.*s.*((1-s).^2) +...
    +3*X_B1.*(s.^2).*(1-s) + X_B.*s.^3;
    dX_bez = (3*X_B.*s.^2).*ds - (3*X_B1.*s.^2).*ds +...
    - (3*X_A.*(s - 1).^2).*ds +...
    (3*X_A1.*(s - 1).^2).*ds + (3*X_A1.*s.*(2*s - 2)).*ds +...
    +(6*X_B1.*s.*(s - 1)).*ds;

case 4
    for i=1:size(s,2)
        S1(i,:)=[(1-s(i))^4  (s(i))^4 ];
        S2(i,:)=[4*s(i)*(1-s(i))^3  6*(s(i))^2*(1-s(i))^2 4*s(i)^3*(1-s(i))];
        N(i,:)=[X(1,i) X(2,i) X(3,i)];
    end
    C1=[x_i';x_f'];
    C2=pinv(S2)*(N-S1*C1);
    coeff=[C1; C2]';
    X_A = coeff(:,1);
    X_B = coeff(:,2);
    X_A1 = coeff(:,3);
    X_B1 = coeff(:,4);
    X_B2 = coeff(:,5);
    V_A = 0;
    V_B = 0;
    X_bez = X_A.*((1-s).^4) +4*X_A1.*s.*((1-s).^3) +...
        6*X_B1.*(s.^2).*(1-s).^2 + 4*X_B2.*(s.^3).*(1-s) +X_B.*s.^4;
    dX_bez = ds.*(4*X_B.*s.^3 - 4*X_B2.*s.^3 ...
    + 4*X_A.*(s - 1).^3 - 4*X_A1.*(s - 1).^3 +...
        - 12*X_A1.*s.*(s - 1).^2 + 12*X_B1.*s.*(s - 1).^2 +...
        - 12*X_B2.*(s.^2).*(s - 1) + 6*X_B1.*(s.^2).*(2*s - 2));
case 5
    for i=1:size(s,2)
    S1(i,:)=[(1-s(i))^5  (s(i))^5 ];
    S2(i,:)=[5*s(i)*(1-s(i))^4  ...
    10*(s(i))^2*(1-s(i))^3 10*s(i)^3*(1-s(i))^2 5*s(i)^4*(1-s)];
    N(i,:)=[X(1,i) X(2,i) X(3,i)];
```

```
        end
        C1=[x_i';x_f'];
        C2=pinv(S2)*(N-S1*C1);
        coeff=[C1; C2]';
        X_A = coeff(:,1);
        X_B = coeff(:,2);
        X_A1 = coeff(:,3);
        X_B1 = coeff(:,4);
        X_B2 = coeff(:,5);
        X_C1 = coeff(:,6);
        V_A = 0;
        V_B = 0;
        X_bez = X_A.*((1-s).^5) +5*X_A1.*s.*((1-s).^4) +...
        +10*X_B1.*(s.^2).*(1-s).^3 +10*X_B2.*(s.^3).*(1-s).^2+...
        +5*X_C1*(s.^4).*(1-s)+X_B.*s.^5;
        dX_bez = ds.*(5*X_B.*s.^4 - 5*X_C1.*s.^4 - 5*X_A.*(s - 1).^4 +...
            5*X_A1.*(s - 1).^4 + 20*X_A1.*s.*(s - 1).^3 - 20*X_B1.*s.*(s - 1).^3 +...
            - 20*X_C1.*(s.^3).*(s - 1) + 10*X_B2.*(s.^3).*(2*s - 2)+...
            - 30*X_B1.*(s.^2).*(s - 1).^2 + 30*X_B2.*(s.^2).*(s - 1).^2);
end
eul_i=X_i(4:6);
eul_f=X_f(4:6);
E=eul_i+(eul_f-eul_i).*s;
A=E(1,:);
B=E(2,:);
C=E(3,:);
dA=diff(A)/dt;
dA=[dA(1) dA];
dB=diff(B)/dt;
dB=[dB(1) dB];
dC=diff(C)/dt;
dC=[dC(1) dC];
W =[dC.*cos(A).*sin(B) - dB.*sin(A);...
    dB.*cos(A) + dC.*sin(A).*sin(B);...
    dA + dC.*cos(B)];

X=[X;E];
X_bez=[X_bez;E];
dX=[dX;W];
dX_bez=[dX_bez;W];
```

Figura 23: Schema ostacolo presente sulla traiettoria del manipolatore

```
end
```

La funzione riportata richiede in input l'ordine delle curve di Bézier e fornisce in output l'intervallo di tempo t che intercorre tra due successivi istanti di controllo del robot, la posizione e la velocità ($\mathbf{X}$ e $\mathbf{dX}$) tra due istanti di tempo successivi quando l'end effector segue un percorso rettilineo, vale a dire la via più breve per raggiungere la posizione finale desiderata, e la posizione e la velocità interpolata con le curve di Bézier ($\mathbf{X_{bez}}$ e $\mathbf{dX_{bez}}$) che definiscono traiettorie alternative necessarie per evitare urti contro eventuali ostacoli che si potrebbero presentare lungo il tratto rettilineo. Viene specificato un valore di soglia pari a $1 \cdot 10^{-5}$ che rappresenta la distanza minima tra la posizione desiderata dell'end effector e quella reale, per cui il task assegnato al robot viene considerato completato. Dalla funzione *pianificazione* vengono richiamati i vettori posizione iniziale ($\mathbf{X_i}$) da cui parte il robot e il vettore posizione finale ($\mathbf{X_f}$) da raggiungere, e dalla funzione *obstacles_generation* viene richiamata la posizione $\mathbf{O}$ degli ostacoli.
Per prima cosa vengono definiti quattro vettori $\mathbf{X\_prel_1}$, $\mathbf{X\_prel_2}$, $\mathbf{X\_prel_3}$ e $\mathbf{X\_prel_4}$, posti tutti e quattro pari al vettore posizione iniziale $\mathbf{X_i}$; inoltre vengono definiti quattro vettori $\mathbf{dX\_prel_1}$, $\mathbf{dX\_prel_2}$, $\mathbf{dX\_prel_3}$ e $\mathbf{dX\_prel_4}$ i quali hanno tutti i termini con valore pari a zero, in quanto la velocità iniziale viene considerata nulla.
A questo punto si inizia un ciclo *for* facendo variare un parametro m tra i valori 1 e 4, facendo lo switch tra quattro casi in base al valore di m. In ciascuno dei casi, in primo luogo si ricava le velocità attrattive e repulsive dalle funzioni *v_att* e *v_rep* rispettivamente. Successivamente la velocità assoluta dell'end effector viene definita come la somma delle velocità attrattive e repulsive:

$$\mathbf{v} = \mathbf{v_{att}} + \mathbf{v_{rep}}$$

Poi si studia il caso presente nella figura 23, dove l'ostacolo (riportato in colore rosso) si trova esattamente sopra la traiettoria che il manipolatore deve percorrere per andare

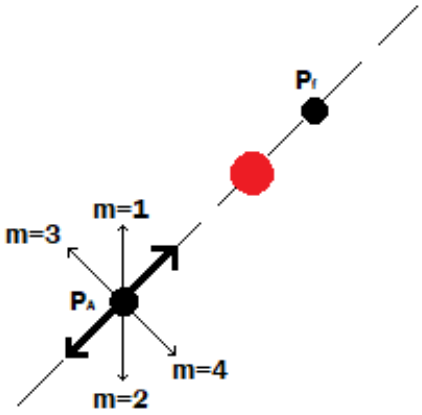


Figura 24: Direzioni di moto in funzione del valore di m

dalla posizione attuale ($\mathbf{P_A}$) alla posizione finale ($\mathbf{P_f}$), perciò sull'end effector agiscono contemporaneamente due forze opposte, la prima attrattiva ($\mathbf{v_{att}}$) che tende a portarlo verso la posizione finale seguendo il percorso più breve, ovvero la traiettoria rettilinea che collega il punto iniziale e quello finale, poi vi agisce una forza repulsiva che tende ad allontanarlo dall'ostacolo. Dunque, come osservabile in figura 23, nasce la situazione dove la forza attrattiva e repulsiva tendono a muovere il manipolatore lungo la traiettoria rettilinea, non riuscendo effettivamente a raggiungere la posizione desiderata. Perciò si introduce un ciclo *if* dove, se la somma del versore associato alla velocità attrattiva e del versore della velocità repulsiva risulta pari a 0, ovvero il manipolatore si muove in loop lungo la retta comune alla posizione finale e all'ostacolo, in tal caso l'algoritmo considera quattro casi, dipendenti dal valore del parametro m, dove il robot esce dalla traiettoria rettilinea lungo i quattro percorsi ortogonali alla stessa, mostrati nella figura 24. Per ciascuna di queste direzioni viene calcolata la posizione dell'end effector come la posizione dello stesso all'istante precedente a cui si somma la velocità (si ricorda che la velocità assoluta viene calcolata coma la somma algebrica delle velocità attrattive e repulsive) per l'intervallo di tempo tra due punti di controllo successivi; inoltre si calcola lo spazio percorso in tale frazione di tempo come la somma dello spazio percorso fino all'istante precedente più il modulo della velocità per il tempo.

Alla fine del ciclo *for*, si genera un vettore chiamato **scelta** che ha come componenti i

valori di spazio **s** che il manipolatore deve percorrere lungo ciascuna delle quattro direzioni mostrate nella figura 24. Utilizzando il comando *min* del software, si ricava il minimo valore incluso nel vettore **scelta** e si crea un indice (*ind*), il cui valore specifica lungo quale delle quattro direzioni considerate si percorre lo spazio minore per arrivare alla posizione finale (ad esempio se il secondo elemento del vettore **scelta** ha il valore minore, questo significa che l'indice avrà valore pari a 2 e si raggiunge la posizione desiderata con il percorso dato dal medesimo valore di m). In base al valore di *ind*, i nuovi valori di posizione, velocità e spazio percorso diventano pari a quelli ottenuti dal caso corrispondente. Dalla funzione *moto* si ottiene il vettore tempo **t** che contiene tutti gli intervalli di controllo da quando inizia il moto a quando finisce, la posizione **s** e la velocità **ds**. Utilizzando il comando *interpl1* di MATLAB, si interpola il percorso che l'end effector deve seguire, conoscendo la direzione in cui deve muoversi (viene dettato dalle velocità attrattive e repulsive agenti sul robot) e lo spazio percorso tra due punti di controllo successivi.
Successivamente si calcola la traiettoria che il manipolatore deve percorrere interpolando con le curve di Bézier. Si studiano tre casi in funzione dell'ordine delle curve di Bézier scelto dall'utente (tipicamente una curva del terzo ordine è in grado di fornire risultati accettabili). Si riporta il caso della curva del terzo ordine, mentre per le curve di ordine maggiore il procedimento è analogo, a differenza del grado dell'equazione delle curve. Innanzitutto, per tutte le colonne del vettore posizione **s** si calcola le due matrici $\mathbf{S_1}$ e $\mathbf{S_2}$, dati dai seguenti polinomi

$$\mathbf{S_1} = [(1-\mathbf{s})^3 \quad \mathbf{s}^3]$$

$$\mathbf{S_2} = [3\mathbf{s}(1-\mathbf{s})^2 \quad 3\mathbf{s}^2(1-\mathbf{s})]$$

Inoltre si definisce la matrice **N** formata dai componenti della i-esima colonna della matrice **X**, ovvero la matrice ottenuta per interpolazione lineare, contenente i punti controllo del percorso che il manipolatore deve eseguire.
Successivamente si definiscono due matrici $\mathbf{C_1}$ e $\mathbf{C_2}$, dove la prima è formata dai vettori posizione iniziale e posizione finale, la seconda è data dalla seguente relazione:

$$\mathbf{C_2} = \mathbf{S_2^*} \cdot (\mathbf{N} - \mathbf{S_1 C_1})$$

A questo punto si genera una matrice **coeff**, formato dalle matrici $\mathbf{C_1}$ e $\mathbf{C_2}$, come riportato di seguito

$$\mathbf{coeff} = \begin{bmatrix} \mathbf{C_1} & \mathbf{C_2} \end{bmatrix}, \tag{5.2}$$

i vettori $\mathbf{X_A}, \mathbf{X_B}, \mathbf{X_{A1}}, \mathbf{X_{B1}}, \mathbf{X_{B2}}$ e $\mathbf{X_{C1}}$ costituiscono le colonne della matrice **coeff**. Con tali vettori si può ricavare la posizione e velocità interpolate con le curve di Bézier, applicando

le relative espressioni

$$\mathbf{X_{bez}} = \mathbf{X_A}(1-\mathbf{s})^3 + 3\mathbf{X_{A1}}\mathbf{s}(1-\mathbf{s})^2 + 3\mathbf{X_{B1}}\mathbf{s}^2 \tag{5.3}$$

$$\begin{aligned}\mathbf{dX_{bez}} =&(3\mathbf{X_B}\mathbf{s}^2)d\mathbf{s} - (3\mathbf{X_{B1}}\mathbf{s}^2)d\mathbf{s} - (3\mathbf{X_A}(\mathbf{s}-1)^2)d\mathbf{s} + (3\mathbf{X_A}\mathbf{1}(\mathbf{s}-1)^2)d\mathbf{s} + \\ &+ (3\mathbf{X_{A1}}\mathbf{s}(2\mathbf{s}-2))d\mathbf{s} + 6(\mathbf{X_{B1}}\mathbf{s}(\mathbf{s}-1))d\mathbf{s}\end{aligned} \tag{5.4}$$

In seguito, una volta che il procedimento descritto finora è stato completato, per qualunque ordine delle curve di Bézier scelto, si definiscono due vettori $\mathbf{eul_i}$ e $\mathbf{eul_f}$, di dimensioni (3x1), contenenti gli angoli di Eulero relativi all'orientamento del dispositivo terminale nella posizione iniziale e finale. La matrice $\mathbf{E}$ è data da:

$$\mathbf{E} = \mathbf{eul_i} + (\mathbf{eul_f} - \mathbf{eul_i})\mathbf{s}$$

Partendo da tale matrice, si ricavano i vettori $\mathbf{A}, \mathbf{B}$ e $\mathbf{C}$ come le righe di $\mathbf{E}$ e si derivano tali vettori ottenendo $\mathbf{dA}, \mathbf{dB}$ e $\mathbf{dC}$, con le quali si forma il vettore $\mathbf{W}$, utilizzando la seguente relazione:

$$\mathbf{W} = \begin{bmatrix} \mathbf{dC}\cdot\cos(\mathbf{A})\sin(\mathbf{B}) - \mathbf{dB}\cdot\sin(\mathbf{A}) \\ \mathbf{dB}\cdot\cos(\mathbf{A}) + \mathbf{dC}\cdot\sin(\mathbf{A})\sin(\mathbf{B}) \\ \mathbf{dA} + \mathbf{dC}\cos(\mathbf{B}) \end{bmatrix}$$

dove le operazioni di moltiplicazione presenti in tale matrice sono prodotti di Hadamard, ovvero moltiplicazioni tra due matrici delle stesse dimensioni (i,j) e fornisce in uscita una matrice prodotto delle stesse dimensioni (i,j) delle matrici di partenza.
Infine, la funzione fornisce in output la traiettoria $\mathbf{X_{bez}}$, calcolata come una curva di Bézier in (5.3), e la velocità lungo tale traiettoria $\mathbf{dX_{bez}}$, calcolata in (5.4).

5.5 Generazione ostacoli

```
function [O,dO]=obstacles_generation(Oi,Of,dOi,dt,T)

t=0:dt:T;

for i=1:size(t,2)
    O(:,:,i)=(Of-Oi)*t(i)/T+Oi;
    if i==1
        dO(:,:,i)=dOi;
```

```
    else
        dO(:,:,i)=(O(:,:,i)-O(:,:,i-1))/dt;
    end
end
```

Questa funzione viene utilizzata per generare ostacoli mobili che il manipolatore dovrà evitare durante la simulazione. Viene richiesto in input il tempo complessivo di simulazione, gli intervalli di controllo in cui si suddivide tale arco di tempo, la posizione iniziale e finale dell'ostacolo (possono anche essere generati in maniera casuale) e la velocità iniziale dell'ostacolo $\mathbf{dO_i}$ (posta nulla in tutte le direzioni in modo da semplificare la simulazione). Si genera un vettore **v** che contiene ciascun istante di controllo del manipolatore, e per ciascun valore contenuto all'interno di tale vettore si calcola la posizione dell'ostacolo come la posizione iniziale $\mathbf{O_i}$ più lo spazio percorso fino al momento attuale. Lo spazio percorso viene calcolato come la distanza tra la posizione iniziale e finale diviso gli intervalli di controllo totali. Per quanto riguarda la velocità dell'ostacolo, questa viene considerata nulla nell'istante iniziale, come specificato nel vettore $\mathbf{dO_i}$, mentre per gli istanti successivi viene calcolata come la differenza tra la posizione attuale e quella all'istante precedente, diviso il tempo *dt* passato tra le due posizioni.
La funzione fornisce come output, istante per istante, la posizione e la velocità dell'ostacolo.

5.6 Determinazione della cinematica simbolica

```
clc
clear
close all

syms d_1 d_4 d_5 d_6 d5 d4 ...
     q1 q2 q3 q4 q5 q6 ...
     a_3 a_4 ...
     dq1 dq2 dq3 dq4 dq5 dq6...
     q dq d a dR R P S w J JO JP real

q=[q1; q2; q3; q4; q5; q6];
dq=[dq1; dq2; dq3; dq4; dq5; dq6];
```

```
d=[d_1; 0; 0; d_4; d_5; d_6];
a=[0; 0; a_3; a_4; 0; 0];

a_0=a(1);a_1=a(2);a_2=a(3);a_3=a(4);a_4=a(5);a_5=a(6);
d_1=d(1);d_2=d(2);d_3=d(3);d_4=d(4);d_5=d(5);d_6=d(6);
q_1=q(1);q_2=q(2);q_3=q(3);q_4=q(4);q_5=q(5);q_6=q(6);


%Matrici di rotazione

R_1=[cos(q_1)       -sin(q_1)         0;
     sin(q_1)        cos(q_1)         0;
     0                 0              1];

 R_2=[-sin(q_2)     0        cos(q_2);
      0            1              0 ;
     -cos(q_2)     0        -sin(q_2)];


R_3= [cos(q_3)     0        sin(q_3);
      0            1            0   ;
     -sin(q_3)    0        cos(q_3)];

R_B1= [1          0           0;
       0          1           0;
       0          0           1];

R_4=[-sin(q_4)     0          +cos(q_4);
    0             1             0    ;
    -cos(q_4)      0         -sin(q_4)];


 R_5=[cos(q_5)    -sin(q_5)     0;
     sin(q_5)      cos(q_5)     0;
       0              0         1];


R_6=[cos(q_6)      0        sin(q_6);
    0              1             0 ;
    -sin(q_6)      0        cos(q_6)];
```

```
 R_7=[ 0   -1   0;
       1    0   0;
       0    0   1];


 %vettori di traslazione dei giunti
p_1=[0;0;d_1];                                          %punto A
p_2=[0;d4;0];                                           %punto B
p_3=[0;0;a_2/3];                                        %punto C
p_4=[0;0;a_3/3];                                        %punto D
p_5=[0;d_4;0];                                          %punto E
p_6=[0;0;d_5];                                          %punto F
p_7=[0;d_6;0];                                          %punto G


%Matrici di trasformazione dei giunti
T_1 = [R_1,p_1;zeros(1,3),1];
T_2 = [R_2,p_2;zeros(1,3),1];
T_3 = [R_3,p_3;zeros(1,3),1];
T_4 = [R_4,p_4;zeros(1,3),1];
T_5 = [R_5,p_5;zeros(1,3),1];
T_6 = [R_6,p_6;zeros(1,3),1];


%base point (terna al telaio)
O =[0 0 0 0 0 0];

%punto A (giunto 1)
A(1:3)=p_1;
A(4:6)=ang_eulero_sym(R_1);
A=simplify(A');

%punto B (giunto 2)
T_02=T_1*T_2;
T_02=simplify(T_02);
B(1:3)=T_02(1:3,4);
B(4:6)=ang_eulero_sym(T_02(1:3,1:3));
B=simplify(B');

%punto B1 (sul link 2)
```

```
p_B1=[0;0;a_2/3];
T_B1=[R_B1,p_B1;zeros(1,3),1];
T_0B1=T_02*T_B1;
B1(1:3)=T_0B1(1:3,4);
B1(4:6)=ang_eulero_sym(T_0B1(1:3,1:3));
B1=simplify(B1');

%punto B2 (sul link 2)
p_B2=[0;0;a_2/3];
T_B2=[R_3,p_B2;zeros(1,3),1];
T_0B2=T_0B1*T_B2;
T_0B2=simplify(T_0B2);
B2(1:3)=T_0B2(1:3,4);
B2(4:6)=ang_eulero_sym(T_0B2(1:3,1:3));
B2=simplify(B2');

%punto C (giunto 3)
T_03=T_0B2*T_3;
T_03=simplify(T_03);
C(1:3)=T_03(1:3,4);
C(4:6)=ang_eulero_sym(T_03(1:3,1:3));
C=simplify(C');

%punto C1 (intermedio tra C e D)
p_c1=[0;-d5;0];
R_c1=R_B1;
T_c1=[R_c1,p_c1;zeros(1,3),1];
T_0c1=T_03*T_c1;
T_0c1=simplify(T_0c1);
C1(1:3)=T_0c1(1:3,4);
C1(4:6)=ang_eulero_sym(T_0c1(1:3,1:3));
C1=simplify(C1');

%punto C2 (intermedio tra C e D)
p_C2=[0;0;a_3/3];
T_C2=[R_B1,p_C2;zeros(1,3),1];
T_0C2=T_0c1*T_C2;
T_0C2=simplify(T_0C2);
C2(1:3)=T_0C2(1:3,4);
C2(4:6)=ang_eulero_sym(T_0C2(1:3,1:3));
C2=simplify(C2');
```

```
%punto C3 (intermedio tra C e D)
p_C3=[0;0;a_3/3];
T_C3=[R_B1,p_C3;zeros(1,3),1];
T_0C3=T_0C2*T_C3;
T_0C3=simplify(T_0C3);
C3(1:3)=T_0C3(1:3,4);
C3(4:6)=ang_eulero_sym(T_0C3(1:3,1:3));
C3=simplify(C3');

%punto D (giunto 4)
T_04=T_0C3*T_4;
T_04=simplify(T_04);
D(1:3)=T_04(1:3,4);
D(4:6)=ang_eulero_sym(T_04(1:3,1:3));


%punto E (giunto 5)
T_05=T_04*T_5;
T_05=simplify(T_05);
E(1:3)=T_05(1:3,4);
E(4:6)=ang_eulero_sym(T_05(1:3,1:3));
E=simplify(E');

%punto F (ginto 6)
 T_06=T_05*T_6;
 T_06=simplify(T_06);
 F(1:3)=T_06(1:3,4);
 F(4:6)=ang_eulero_sym(T_06(1:3,1:3));
 F=simplify(F');

 %punto G (end effector)
%  R7=R_6;
 T_7=[R_7,p_7;zeros(1,3),1];
 T_07=T_06*T_7;
 G(1:3)=T_07(1:3,4);
 G(4:6)=ang_eulero_sym(T_07(1:3,1:3));
 G=simplify(G');


% %%%%%%%%%%%%%%%
```

```
% %Jacobiani
%
% %J_A
 P=A(1:3);
 R=R_1;
 JP=jacobian(P,q);
 JP=simplify(JP);
 for i=1:3
     for j=1:3
         dR(i,j)=0;
         for k=1:length(q)
            dR(i,j)=dR(i,j)+diff(R(i,j),q(k))*dq(k);
         end
     end
 end
 S=dR*R.';
w=[S(3,2); S(1,3); S(2,1)];
JO=jacobian(w,dq); %Jacobiano di orientazione
JO=simplify(JO);
J_A=[JP; JO]


%J_B
P=B(1:3);
R=T_02(1:3,1:3);
JP=jacobian(P,q);
JP=simplify(JP);
for i=1:3
    for j=1:3
        dR(i,j)=0;
        for k=1:length(q)
           dR(i,j)=dR(i,j)+diff(R(i,j),q(k))*dq(k);
        end
    end
end
S=dR*R.';
w=[S(3,2); S(1,3); S(2,1)];
JO=jacobian(w,dq); %Jacobiano di orientazione
JO=simplify(JO);
J_B=[JP; JO]
```

```
%J_B1
P=B1(1:3);
R=T_0B1(1:3,1:3);
JP=jacobian(P,q);
JP=simplify(JP);
for i=1:3
    for j=1:3
        dR(i,j)=0;
        for k=1:length(q)
           dR(i,j)=dR(i,j)+diff(R(i,j),q(k))*dq(k);
        end
    end
end
S=dR*R.';
w=[S(3,2); S(1,3); S(2,1)];
JO=jacobian(w,dq); %Jacobiano di orientazione
JO=simplify(JO);
J_B1=[JP; JO]

%J_B2
P=B2(1:3);
R=T_0B2(1:3,1:3);
JP=jacobian(P,q);
JP=simplify(JP);
for i=1:3
    for j=1:3
        dR(i,j)=0;
        for k=1:length(q)
           dR(i,j)=dR(i,j)+diff(R(i,j),q(k))*dq(k);
        end
    end
end
S=dR*R.';
w=[S(3,2); S(1,3); S(2,1)];
JO=jacobian(w,dq); %Jacobiano di orientazione
JO=simplify(JO);
J_B2=[JP; JO]

%J_C
P=C(1:3);
```

```
R=T_03(1:3,1:3);
JP=jacobian(P,q);
JP=simplify(JP);
for i=1:3
    for j=1:3
        dR(i,j)=0;
        for k=1:length(q)
           dR(i,j)=dR(i,j)+diff(R(i,j),q(k))*dq(k);
        end
    end
end
S=dR*R.';
w=[S(3,2); S(1,3); S(2,1)];
JO=jacobian(w,dq); %Jacobiano di orientazione
JO=simplify(JO);
J_C=[JP; JO]

%J_C1
P=C1(1:3);
R=T_0c1(1:3,1:3);
JP=jacobian(P,q);
JP=simplify(JP);
for i=1:3
    for j=1:3
        dR(i,j)=0;
        for k=1:length(q)
           dR(i,j)=dR(i,j)+diff(R(i,j),q(k))*dq(k);
        end
    end
end
S=dR*R.';
w=[S(3,2); S(1,3); S(2,1)];
JO=jacobian(w,dq); %Jacobiano di orientazione
JO=simplify(JO);
J_C1=[JP; JO]

%J_C2
P=C2(1:3);
R=T_0C2(1:3,1:3);
JP=jacobian(P,q);
JP=simplify(JP);
```

```
for i=1:3
    for j=1:3
        dR(i,j)=0;
        for k=1:length(q)
           dR(i,j)=dR(i,j)+diff(R(i,j),q(k))*dq(k);
        end
    end
end
S=dR*R.';
w=[S(3,2); S(1,3); S(2,1)];
JO=jacobian(w,dq); %Jacobiano di orientazione
JO=simplify(JO);
J_C2=[JP; JO]

%J_C3
P=C3(1:3);
R=T_0C3(1:3,1:3);
JP=jacobian(P,q);
JP=simplify(JP);
for i=1:3
    for j=1:3
        dR(i,j)=0;
        for k=1:length(q)
           dR(i,j)=dR(i,j)+diff(R(i,j),q(k))*dq(k);
        end
    end
end
S=dR*R.';
w=[S(3,2); S(1,3); S(2,1)];
JO=jacobian(w,dq); %Jacobiano di orientazione
JO=simplify(JO);
J_C3=[JP; JO]

%J_D
P=D(1:3);
R=T_04(1:3,1:3);
JP=jacobian(P,q);
JP=simplify(JP);
for i=1:3
    for j=1:3
        dR(i,j)=0;
```

```
        for k=1:length(q)
           dR(i,j)=dR(i,j)+diff(R(i,j),q(k))*dq(k);
        end
    end
end
S=dR*R.';
w=[S(3,2); S(1,3); S(2,1)];
JO=jacobian(w,dq); %Jacobiano di orientazione
JO=simplify(JO);
J_D=[JP; JO]

%J_E
P=E(1:3);
R=T_05(1:3,1:3);
JP=jacobian(P,q);
JP=simplify(JP);
for i=1:3
    for j=1:3
        dR(i,j)=0;
        for k=1:length(q)
           dR(i,j)=dR(i,j)+diff(R(i,j),q(k))*dq(k);
        end
    end
end
S=dR*R.';
w=[S(3,2); S(1,3); S(2,1)];
JO=jacobian(w,dq); %Jacobiano di orientazione
JO=simplify(JO);
J_E=[JP; JO]

%J_F
P=F(1:3);
R=T_06(1:3,1:3);
JP=jacobian(P,q);
JP=simplify(JP);
for i=1:3
    for j=1:3
        dR(i,j)=0;
        for k=1:length(q)
           dR(i,j)=dR(i,j)+diff(R(i,j),q(k))*dq(k);
        end
```

```
    end
end
S=dR*R.';
w=[S(3,2); S(1,3); S(2,1)];
JO=jacobian(w,dq); %Jacobiano di orientazione
JO=simplify(JO);
J_F=[JP; JO]

%J_G
P=G(1:3);
R=T_07(1:3,1:3);
JP=jacobian(P,q);
JP=simplify(JP);
for i=1:3
    for j=1:3
        dR(i,j)=0;
        for k=1:length(q)
           dR(i,j)=dR(i,j)+diff(R(i,j),q(k))*dq(k);
        end
    end
end
S=dR*R.';
w=[S(3,2); S(1,3); S(2,1)];
JO=jacobian(w,dq); %Jacobiano di orientazione
JO=simplify(JO);
J_G=[JP; JO]
```

Nella funzione della cinematica diretta sono presenti tutti i dati geometrici riguardanti il robot, ovvero le lunghezze dei corpi che lo costituiscono e l'orientamento reciproco tra due terne presenti su due giunti successivi, espresso dalle matrici di rotazione (nella configurazione iniziale, invece di rispettare la convenzione di Denavit-Hartenberg, come avviene solitamente in questo tipo di studio, viene invece considerato l'orientamento delle terne presenti nel modello CAD incluso nel file *urdf*). Partendo dai vettori posizione (che esprimono la posizione di un giunto o punto di controllo rispetto a quello precedente, andando dalla terna base all'end effector) e dalle relative matrici di rotazione (2.2), si possono costruire le matrici di trasformazione (2.16), ovvero delle matrici 4x4 aventi sulle prime tre colonne la matrice di rotazione e zeri nell'ultima riga, mentre l'ultima colonna è formata dal vettore posizione come primi tre elementi, mentre l'ultimo termine ha valore unitario.
Partendo da tali matrici di trasformazione, si possono ricavare dei vettori 6x1 che esprimono

la posizione e l'orientamento di ciascun punto di controllo. Per la terna base tale vettore ha tutti i valori nulli, in quanto si tratta di una terna fissa che non ha né moto di traslazione né moto di rotazione. Si consideri la terna immediatamente successiva a quella base, nominata con la lettera A nella figura 25; per tale terna si vuole creare un vettore (6x1) che contenga dati sulla posizione e orientamento di tale terna rispetto alla terna base. Tale vettore ha come primi tre elementi il vettore posizione $\mathbf{p_1}$ rispetto alla terna base e gli ultimi tre termini di tale vettore sono gli angoli di Eulero relativi alla matrice di rotazione $\mathbf{R_1}$ di tale punto rispetto al punto di controllo precedente. Per ricavare gli angoli di Eulero, partendo dalla matrice di rotazione, si utilizza la funzione *ang_eulero_sym*, riportata nel capitolo 5.8.
Per il punto successivo B, il calcolo è leggermente più complesso, in quanto la matrice di trasformazione di tale terna si ottiene applicando la relazione (2.17), ovvero moltiplicando le matrici di trasformazione di tutti i punti di controllo presenti tra la terna base fino al punto considerato (questo procedimento viene poi applicato per tutti i punti successivi, fino all'end effector). Una volta ottenuta la matrice di trasformazione, il vettore $\mathbf{B}$ che esprime la *posa* di tale punto, avrà come primi tre elementi le coordinate del punto rispetto alla terna base (tali valori si possono ricavare come le prime tre righe dell'ultima colonna della matrice di trasformazione), mentre gli ultimi tre termini del vettore saranno gli angoli di Eulero ottenuti dalla conversione della matrice di rotazione (prime tre righe delle prime tre colonne della matrice di trasformazione).
Questo procedimento va ripetuto per tutti i punti di controllo imposti sulla struttura del manipolatore.
Finito questo processo, questa funzione fornisce in output i vettori contenenti la posizione e orientamento (in termini di angoli di Eulero) per tutti i punti di controllo del manipolatore (i punti di controllo sono posizionati su tutti i giunti e ce ne sono alcuni presenti sui bracci di lunghezza elevata, per i quali non risulta più sufficiente il volume delle sfere di controllo poste sui giunti alle estremità, in quanto vi restano delle zone "cieche", non in grado di rilevare eventuali collisioni con oggetti esterni). Lo schema dei punti di controllo e l'orientamento delle relative terne viene riportato nella figura 25.
Fatto tale procedimento per tutti i punti di controllo del sistema, il passo successivo consiste nel calcolare i Jacobiani per tutti i punti del sistema; nello specifico, i Jacobiani geometrici si calcolano come la "combinazione" dei Jacobiani di posizione e quelli di orientamento, come indicato nella (3.4). Per il calcolo dello Jacobiano di posizione ($\mathbf{J_P}$) si utilizza la funzione *jacobian* di MATLAB, fornendo in input la posizione del punto considerato (primi tre termini del vettore associato a quel punto) e il vettore $\mathbf{q}$ di posizione nello spazio dei

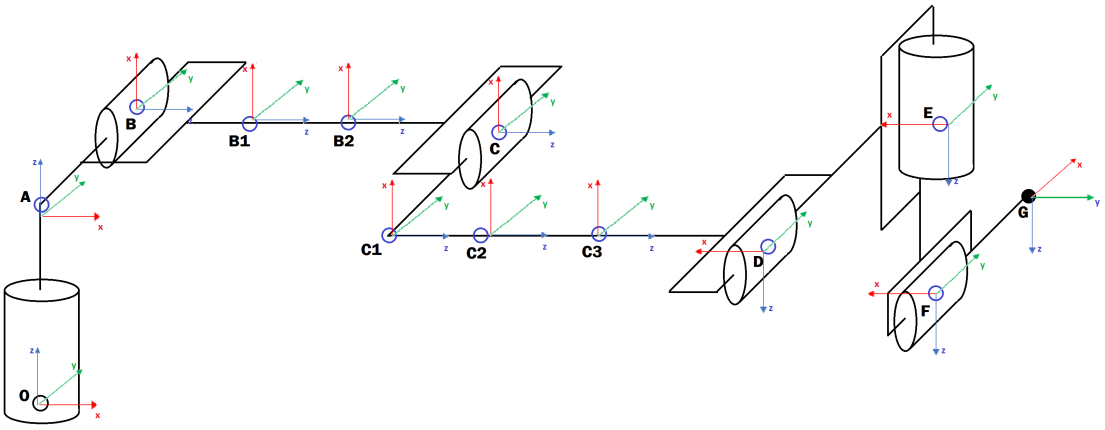


Figura 25: Schema cinematico del robot

giunti. Per il calcolo dello Jacobiano di orientamento ($\mathbf{J_O}$) la procedura è leggermente più complessa. Innanzitutto, come già visto nel capitolo 3.1, bisogna ricavare la matrice **S**, data come il prodotto tra la derivata della matrice di rotazione per la trasposta della stessa, come riportato nella seguente equazione:

$$\mathbf{S}(t) = \dot{\mathbf{R}}(t)\mathbf{R}^{\mathbf{T}}(t)$$

Ricavare tale matrice è comodo in quanto si tratta di una matrice antisimmetrica, che ha sugli elementi fuori dalla diagonale le tre componenti della velocità angolare:

$$\mathbf{S} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & \omega_x \\ \omega_y & \omega_x & 0 \end{bmatrix}$$

Dunque, si estraggono i valori positivi delle velocità angolari da tale matrice e si genera un vettore **w** che li contenga tutti e tre; in seguito, il vettore **w** appena generato, insieme a **dq** che esprime la velocità nello spazio dei giunti, viene introdotta nella funzione *jacobian* di MATLAB, ottenendo così la matrice jacobiana di orientamento. La matrice jacobiana (complessiva) sarà costituita dallo Jacobiano di posizione e quello di orientamento.

$$\mathbf{J} = \begin{bmatrix} \mathbf{J_P} & \mathbf{J_O} \end{bmatrix}$$

Questa funzione fornisce in output le matrici jacobiane per tutti i punti del sistema, le quali verranno poi ricopiate in un'altra funzione MATLAB e verranno utilizzate per calcolare la cinematica inversa dei vari punti di controllo del robot.

5.7 Calcolo dei parametri di guadagno

```
function     [a_v, a_h, a_e]=gains(d,d_ee,r,r_min)

d1=r;
d2=(r_min+r)/2;
d3=r_min;

if d<=d2
    a_h=1;
else
    if d>=d1
        a_h=0;
    else
        a_h=0.5*(1+cos(pi*(d-d2)/(d1-d2)));
    end
end



if d>=d2
    a_v=0;
else
    a_v=((d-d2)/(d3-d2))^2;
end



if d_ee<=d2
    a_e=1;
else
    if d_ee>=d1
        a_e=0;
    else
        a_e=0.5*(1+cos(pi*(d_ee-d2)/(d1-d2)));
    end
end

end
```

Questa funzione calcola i coefficienti a_v, a_h e a_e necessari per risolvere l'equazione della

cinematica inversa (4.14), la quale tiene conto anche delle componenti della velocità attrattive e repulsive che agiscono sull'end effector e il punto del robot più vicino all'ostacolo rispettivamente. Per calcolare tali fattori, occorre innanzitutto definire la distanza r, cioè la distanza limite entro la quale le componenti sensoristiche del robot sono in grado di percepire corpi estranei presenti nello spazio circostante; r_{min}, cioè il raggio minimo, superato il quale il robot si arresta per garantire la sicurezza di chiunque o qualunque cosa si sia avvicinato troppo a esso, mentre r_m è un raggio intermedio tra i due precedenti, e per una distanza rispetto all'ostacolo compresa tra r_m e r_{min}, il robot è in grado di modificare la propria traiettoria a seconda delle velocità attrattive e repulsive che vi agiscono, regolate appunto dai tre parametri calcolati con la questa funzione MATLAB, i quali hanno il compito di rendere più scorrevole il moto dell'organo terminale, evitando fenomeni di discontinuità. Definiti i tre raggi (presi in input dalla funzione *pianificazione*), il programma si calcola a_h, a_v e a_e applicando le equazioni (4.15) e (4.16).

5.8 Conversione delle matrici di rotazione in angoli di Eulero

```
function E= ang_eulero_sym(R)

b=atan2(sqrt(R(3,1)^2+R(3,2)^2), R(3,3));
if b==0
    a=0;
    c=atan2(R(2,1),R(1,1));
else
   a= atan2(R(2,3)/sin(b),R(1,3)/sin(b));
   c=atan2(R(3,2)/sin(b),-R(3,1)/sin(b));
end
E=[a b c];

end
```

Questa funzione richiede in input la matrice di rotazione **R** del giunto considerato (le matrici di rotazione per i vari joint del sistema sono riportati nella funzione *cinematica simbolica*, Capitolo 5.6) e fornisce in uscita i relativi angoli di Eulero (**Roll-Pitch-Roll)**, ottenuti con rotazioni successive attorno agli assi ZYZ. Per calcolare gli angoli di Eulero relativi alla matrice di rotazione considerata, questa funzione utilizza le equazioni (2.13), fornendo in

uscita la terna di angoli φ, θ e ψ. Viene anche considerato il caso di singolarità cinematica di rappresentazione in cui $\theta = 0$, e in tale situazione vengono forniti in output valori nulli per gli angoli φ e θ, mentre l'ultimo elemento della terna assume il valore:

$$\psi = atan2(r_{21}, r_{11})$$

Il codice scritto sopra è utile soltanto quando si hanno dei valori *simbolici* (ovvero dei valori non numerici) per gli elementi della matrice di rotazione, espressi in funzione degli angoli di rotazione dei giunti q_i. Qualora fossero noti i valori precisi delle rotazioni dei giunti, si può applicare, per le matrici di rotazione riportate nella funzione *pianificazione*, il seguente comando di MATLAB:

```
rotm2eul(R,'ZYZ')
```

Questa funzione (*rotm2eul*) richiede in input la matrice di rotazione **R** per la quale si vogliono calcolare gli angoli di Eulero, e gli assi intorno ai quali si vogliono fare le rotazioni successive, e fornisce in uscita gli angoli di Eulero relativi alla matrice data.

5.9 Calcolo della velocità attrattiva

```
function v=v_att(P,G,d_G,v_0)

v_0=v_0/d_G;
d=norm(G-P);
if d<=d_G
    v=v_0*(G-P);
else
    v=v_0*d_G/d*(G-P);
end
end
```

In questa funzione viene generata la velocità attrattiva agente sull'end effector che tende a riportarlo sulla traiettoria diretta verso la posizione finale che esso deve raggiungere. Viene fornito in ingresso la posizione **P** del dispositivo terminale del robot, la posizione finale **G** che esso deve raggiungere, il raggio d_G di controllo, e la velocità di attrazione assoluta v_{att}. Con tali valori si applica il sistema di equazione (4.17), fornendo in uscita dal programma il

valore di velocità attrattiva ($\mathbf{v}_{att}$), calcolata confrontando la distanza d_G con la distanza tra la posizione finale la posizione attuale dell'end effector.

5.10 Calcolo della velocità repulsiva

```
function v=v_rep(P,O,d_O,v_0)

dimension=size(P,1);
v=zeros(dimension,1);

for i=1:size(O,2)
    d=norm(P-O(:,i));
    nabla_d=(P-O(:,i))/d;
    if d<=d_O
        v=v+v_0*(1/d-1/d_O)/d^2*nabla_d;
    end
end
end
```

Per completare il procedimento di calcolo della traiettoria, previsto nel capitolo 4.4, ci si calcola la velocità repulsiva agente sul dispositivo terminale del robot. Per fare ciò viene applicata l'equazione (4.18). Viene richiesto in input la posizione del dispositivo terminale, la posizione dell'ostacolo, il raggio d_O di controllo del robot e la velocità repulsiva nominale v_{rep}. Perciò si applica il sistema di equazioni riportato in (4.18), confrontando la distanza tra il punto critico e l'ostacolo con il raggio di controllo della sensoristica del robot.

5.11 Plot delle sfere

```
function plot_sfere(ostacoli, d_soglia)

for i=1:size(ostacoli, 2)

centro=ostacoli(:,i);
```

```
plot3(centro(1), centro(2), centro(3),'or');hold on
[x,y,z] = sphere;
surf(d_soglia*x+centro(1),d_soglia*y+centro(2),d_soglia*z+centro(3),...
   'FaceColor',[1 0 0],'EdgeColor',[1 0 0])
end
```

La finalità di questa funzione è di tracciare delle sfere intorno agli ostacoli generati dal programma. Per fare ciò, innanzitutto si prendono le posizioni degli ostacoli dalla funzione *obstacles generation* e vi si tracciano delle sfere di diametro prestabilito (d_{soglia}), questo per dare un volume maggiore alle sfere, in modo che queste possano essere più verosimili ad oggetti reali.

5.12 Moto

```
function [Output_moto] = moto(qi,vi,ai,qf,vf,af,ti,scelta)
global dt T

tf=ti+T;
samples=round(T/dt);
t = (ti:T/samples:tf);

if scelta == 1
a0 = qi;
a1 = vi;
a2 = ai/2;
a3 = (20*(qf-qi)-(8*vf+12*vi)*T-(3*af-ai)*T^2)/(2*T^3);
a4 = (30*(qi-qf)+(14*vf+16*vi)*T+(3*af-2*ai)*T^2)/(2*T^4);
a5 = (12*(qf-qi)-6*(vf+vi)*T-(af-ai)*T^2)/(2*T^5);

for k = 1:size(t,2)
  M(k,:) = a0 + a1*(t(k)-ti) + a2*(t(k)-ti)^2 +...
  + a3*(t(k)-ti)^3 + a4*(t(k)-ti)^4 + a5*(t(k)-ti)^5;
  V(k,:) = a1 + 2*a2*(t(k)-ti) + 3*a3*(t(k)-ti)^2 + 4*a4*(t(k)-ti)^3 + 5*a5*(t(k)-ti)^4;
  A(k,:) = 2*a2 + 6*a3*(t(k)-ti) + 12*a4*(t(k)-ti)^2 + 20*a5*(t(k)-ti)^3;
end
```

```
elseif scelta == 2  %v_trapezoidale
    Ta = T/4;
    vcost = (qf-qi)/(T-Ta);
    for k = 1:size(t,2)
        if t(k)<=(ti+Ta)
           M(k,:) = qi + vcost*(t(k)-ti)^2/(2*Ta);
           V(k,:) = vi + vcost*(t(k)-ti)/Ta;
           A(k,:) = vcost/Ta;
        elseif t(k)>(ti+Ta) && t(k)<=(tf-Ta)
           M(k,:) = qi + vcost*(t(k)-ti-Ta/2);
           V(k,:) = vi + vcost;
           A(k,:) = zeros(1,7);
        elseif t(k)>(tf-Ta)
           M(k,:) = qf - vcost*(tf-t(k))^2/(2*Ta);
           V(k,:) = vf + vcost*(tf-t(k))/Ta;
           A(k,:) = -vcost/Ta;
        end
    end

end

Output_moto = [t',M,V,A];

end
```

In questa funzione si calcolano le posizioni ($\mathbf{M}$), velocità ($\mathbf{V}$) e accelerazioni ($\mathbf{A}$) nello spazio dei giunti, noto il tempo T totale di moto, fornendo in input i valori della posizione iniziale ($\mathbf{q_i}$), velocità iniziale ($\mathbf{v_i}$), accelerazione iniziale ($\mathbf{a_i}$), posizione finale ($\mathbf{q_f}$), velocità finale ($\mathbf{v_f}$), accelerazione finale ($\mathbf{a_f}$) e il tempo iniziale (t_i). Bisogna evidenziare subito che le velocità e accelerazioni iniziali e finali, così come il tempo iniziale, sono tutti termini nulli.
Si vuole calcolare la traiettoria nello spazio dei giunti applicando una espressione polinomiale come quella riportata di seguito [5]

$$q(t) = a_0 + a_1(t - t_0) + a_2(t - t_0)^2 + ... + a_n(t - t_0)^n \tag{5.5}$$

In (5.5) n esprime il grado del polinomio, dato dal numero di condizioni al contorno che si desidera soddisfare, e dal grado del polinomio dipende la *scorrevolezza* della traiettoria ottenuta. Bisogna dire che le trattorie ricavate in questa maniera sono piuttosto semplici, sia da un punto di vista della valutazione dei coefficienti che da quello della valutazione puntuale nei vari istanti di tempo.

Per la risoluzione del polinomio (5.5), si consideri la seguente espressione matriciale:

$$\mathbf{Xa} = \mathbf{y} \tag{5.6}$$

dove:

- $\mathbf{X}$ è una matrice nota di dimensioni $(n+1) \times (n+1)$
- $\mathbf{y}$ è il vettore dei termini noti
- $\mathbf{a} = [a_0, a_1, ..., a_n]^T$ è il vettore dei parametri incogniti da calcolare

Invertendo la (5.6) si ottiene:

$$\mathbf{a} = \mathbf{X}^{-1}\mathbf{y} \tag{5.7}$$

Inoltre, per derivazione dell'equazione (5.5), si possono ottenere le espressioni delle velocità e accelerazioni nello spazio dei giunti:

$$v(t) = \dot{q}(t) = a_1 + 2a_2(t - t_0) + ... + na_n(t - t_0)^{n-1} \tag{5.8}$$

$$a(t) = \ddot{q}(t) = 2a_2 + 6a_3(t - t_0) + ... + n(n-1)a_n(t - t_0)^{n-2} \tag{5.9}$$

Nel caso specifico considerato il numero di equazioni di condizioni al contorno sono sei, ovvero:

$$q_i = 0 \qquad q_f = 1$$

$$v_i = 0 \qquad v_f = 0$$

$$a_i = 0 \qquad a_f = 0$$

ponendo $T = t_f - t_0$, dalla equazione (5.7) si possono ricavare i seguenti coefficienti:

$$a_0 = q_0$$

$$a_1 = v_0$$

$$a_2 = \frac{1}{2}a_0$$

$$a_3 = \frac{1}{2T^3}[20(q_f - q_i) - (8v_f + 12v_i)T - (3a_i - a_f)T^2]$$

$$a_4 = \frac{1}{2T^4}[-30(q_f - q_i) + (14v_f + 16v_i)T + (3a_i - 2a_f)T^2]$$

$$a_5 = \frac{1}{2T^5}[12(q_f - q_i) - 6(v_f + v_i)T + (a_f - a_i)T^2]$$

Inserendo questi sei coefficienti trovati possono essere inseriti nel seguente polinomio del quinto ordine, derivante da (5.5):

$$q(t) = a_0 + a_1 T + a_2 T^2 + a_3 T^3 + a_4 T^4 + a_5 T^5 \tag{5.10}$$

Derivando la (5.10) si possono ottenere le espressioni polinomiali della velocità e dell'accelerazione. La funzione fornisce in output il vettore dei tempi **t**, la traiettoria nello spazio dei giunti **M**, le velocità e accelerazioni nello spazio dei giunti (**V** e **A**).

5.13 Plot robot

```
function plot_UR5(Q,r)
global UR5

 [X, punti]=kin_dir_UR5_3(Q);

plot3(punti(1,:),punti(2,:),punti(3,:),'-ok')
hold on

 for i=[3,4,5,6,7,8,9,10,11,12,13]

centro=punti(:,i);
[x,y,z] = sphere;

surf(r*x+centro(1),r*y+centro(2),r*z+centro(3),...
    'FaceColor',[0.8 0.8 0.8],'FaceAlpha',0.05,'EdgeColor',[0.8 0.8 0.8],'EdgeAlpha',0.25)
end

P=punti(1:3,1);
eul=punti(4:6,1);
R= eul2rotm(eul','ZYZ');
quiver3(P(1),P(2),P(3),R(1,1),R(2,1),R(3,1),0.2,'r')
quiver3(P(1),P(2),P(3),R(1,2),R(2,2),R(3,2),0.2,'g')
quiver3(P(1),P(2),P(3),R(1,3),R(2,3),R(3,3),0.2,'b')

P=punti(1:3,13);
eul=punti(4:6,13);
R= eul2rotm(eul','ZYZ');
```

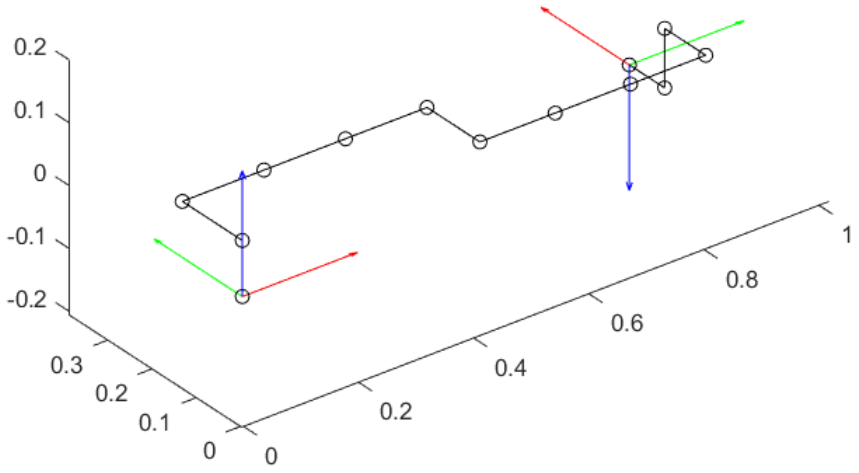


Figura 26: Punti di controllo presenti sul robot

```
quiver3(P(1),P(2),P(3),R(1,1),R(2,1),R(3,1),0.2,'r')
quiver3(P(1),P(2),P(3),R(1,2),R(2,2),R(3,2),0.2,'g')
quiver3(P(1),P(2),P(3),R(1,3),R(2,3),R(3,3),0.2,'b')

show(UR5,Q', 'Frames','off');

end
```

Per visualizzare correttamente il robot su un *plot* di MATLAB, bisogna innanzitutto riportare i punti di controllo, ricavati tramite la cinematica diretta del manipolatore (Capitolo 5.6) raffigurati schematicamente nella figura 25. La cinematica iniziale del robot è riportata nella figura 26, dove vengono visualizzati tutti e tredici i punti di controllo posizionati sulla sua struttura.
In seguito, su tali punti di controllo si vogliono visualizzare delle sfere che rappresentano lo spazio di controllo, ovvero la zona circostante al robot in cui esso può rilevare la presenza di eventuali corpi esterni. Si può selezionare il raggio di tali sfere (corrispondente a r specificato nella funzione *pianificazione*), il colore e la loro trasparenza (con il comando *FaceAlpha*). Dunque si ottiene in risultato riportato in figura 27.

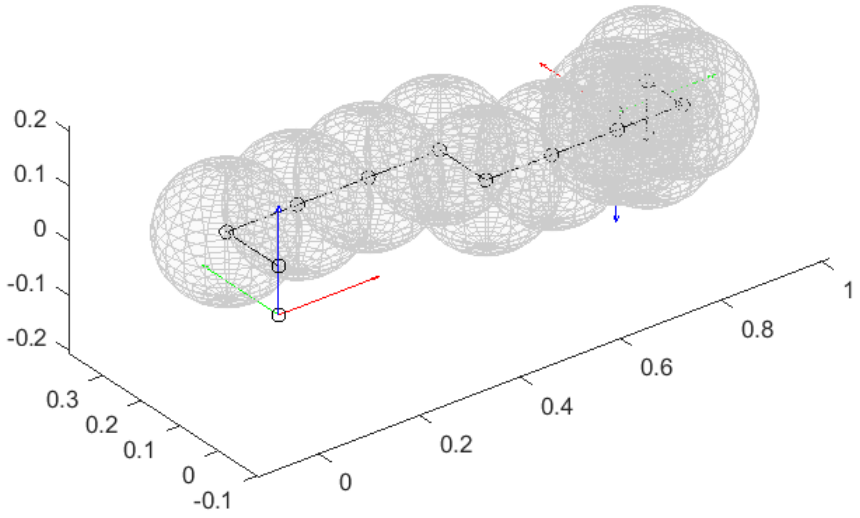


Figura 27: Zone rilevabili sensoristicamente dal robot

L'ultimo passo di questa funzione consiste nel visualizzare il modello CAD del robot UR5, presente nel file *urdf*, nello stesso grafico presente in figura 27. Per fare ciò si utilizza il seguente comando:

```
show(UR5,Q', 'Frames','off');
```

ottenendo così il plot definito del robot, riportato nella figura 28. Con questo si andranno a fare le successive simulazioni per verificare l'effettivo funzionamento dell'algoritmo di *Obstacles avoidance* descritto nel paragrafo 4.3.
Bisogna aggiungere, per correttezza, che il file *urdf* utilizzato per lo svolgimento di questa tesi e mostrato nella figura 28, è stato preso dal seguente blog di MATLAB

https://it.mathworks.com/matlabcentral/answers/511729-load-ur5-robot-with-robotics-system-toolbox

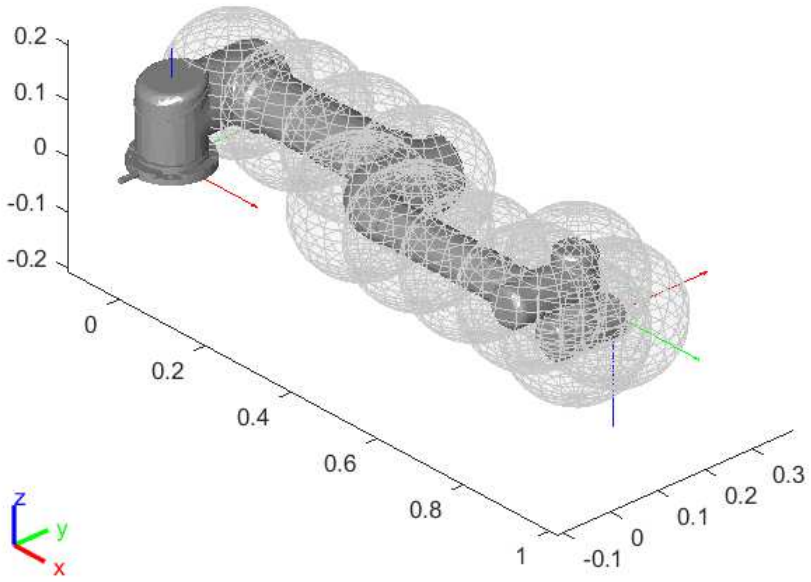


Figura 28: Modello CAD del robot con tutti i punti di controllo

6 Risultati e simulazioni

Gli algoritmi riportati nel capitolo 5 possono essere utilizzati per compiere simulazioni in MATLAB. Modificando opportunamente la funzione *pianificazione* (5.1), si possono simulare diverse traiettorie di lavoro del robot riportato nella figura 28; inoltre, all'interno della simulazione possono essere inseriti anche degli ostacoli, sia fissi che mobili, per verificare la validità della strategia di *obstacle avoidance*, riportata nel capitolo 4.

Nelle immagini presenti nella figura 29 è presente l'esempio del robot che si muove da una posa iniziale a una posa finale, perciò anche l'orientamento dell'end effector nel punto finale è prestabilito. Come era stato previsto, in assenza di ostacoli esterni, il manipolatore segue un moto lineare per raggiungere la posizione finale, in modo da compiere il moto nel minor tempo possibile.

In figura 30 è riportato il caso del moto di un robot dalla posizione iniziale alla posizione finale (è visibile una terna locale in ciascuna delle due posizioni), ma lungo tale spostamento si avrebbe un urto contro un ostacolo (sfera rossa), perciò esso modifica la propria traiettoria

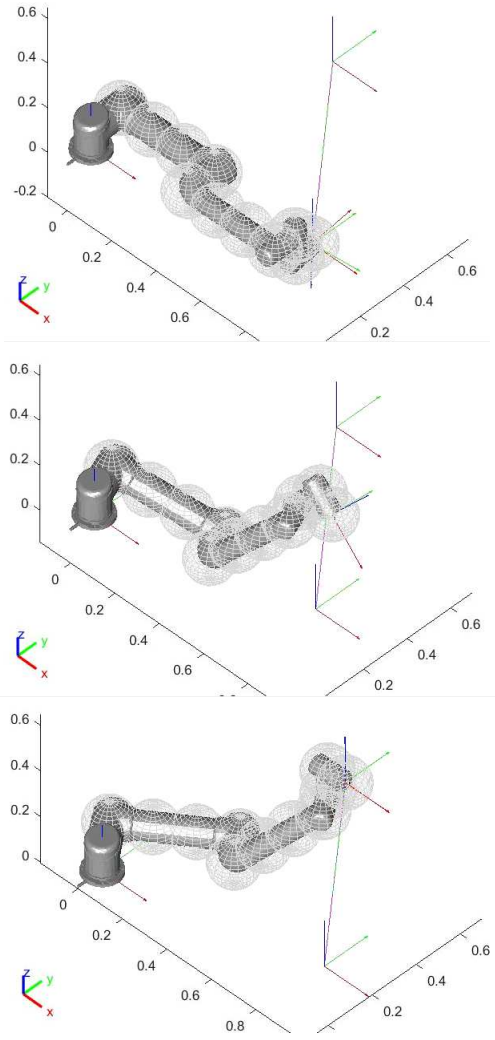
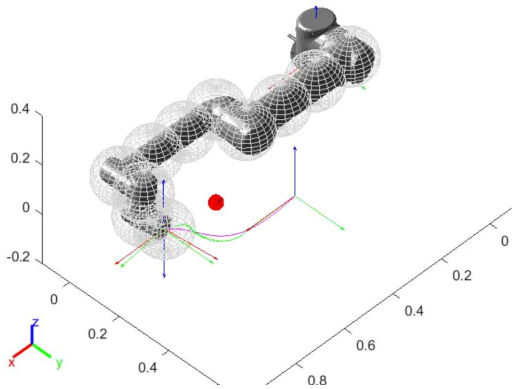
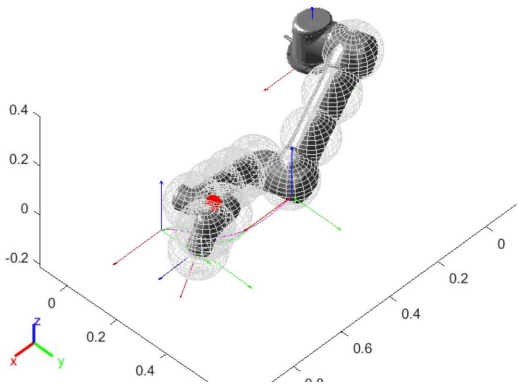


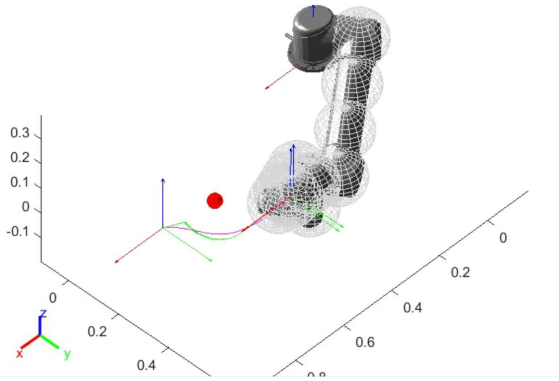
Figura 29: Simulazione moto senza ostacoli



(a) Manipolatore nella posizione iniziale



(b) Manipolatore in una posizione intermedia



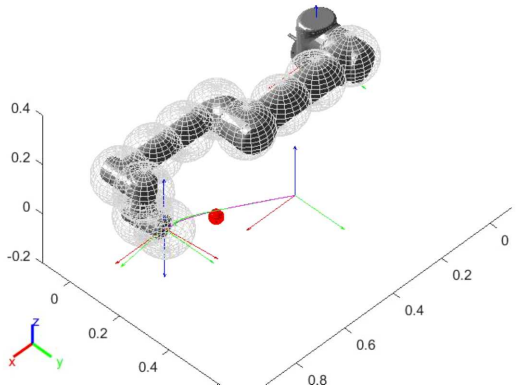
(c) Manipolatore nella posizione finale

Figura 30: Simulazione moto con un ostacolo immobile

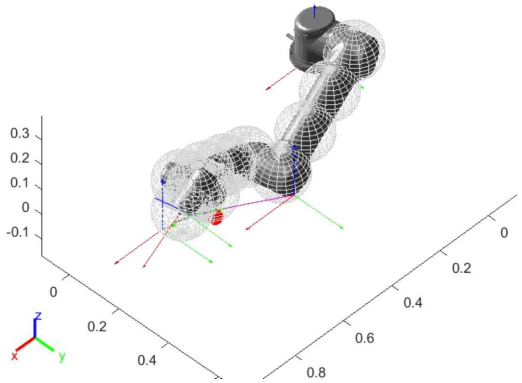
in prossimità dell'oggetto (30b), raggiungendo, infine, la posizione finale.

In figura 31 è riportata una situazione simile, con il robot che incontra un ostacolo immobile durante il proprio moto, l'unica differenza rispetto alla situazione 30 è che la posizione finale che esso deve raggiungere si trova spostato verso l'alto, cioè la coordinata z è maggiore nella situazione 31. In questa situazione, al fine di evitare l'ostacolo, il manipolatore vi passa *sopra*, mentre prima passava sotto, questo perché il robot si calcola e segue la traiettoria più breve per arrivare a destinazione, dunque nella nuova situazione il percorso più breve si trova sopra l'oggetto.

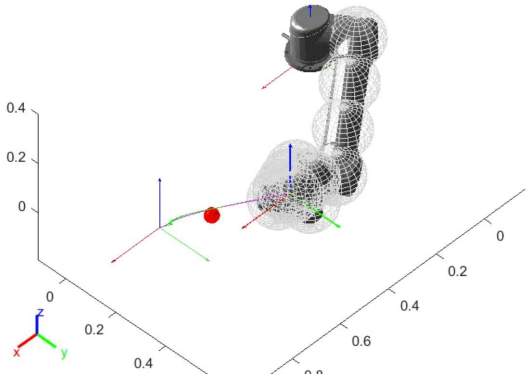
Nelle figure 32 e 33 sono presenti le situazioni dove il robot, nell'esecuzione del suo compito, entrerebbe in collisione con uno e due ostacoli mobili, rispettivamente. In entrambe le situazioni, al fine di evitare l'urto e portare a termine il suo moto, il manipolatore fuoriesce dalla traiettoria lineare che congiunge la posizione iniziale e finale, per poi riprendere a seguire tale traiettoria una volta raggiunta la distanza di sicurezza rispetto all'ostacolo o agli ostacoli (figure 32b e 33b, rispettivamente).



(a) Manipolatore nella posizione iniziale

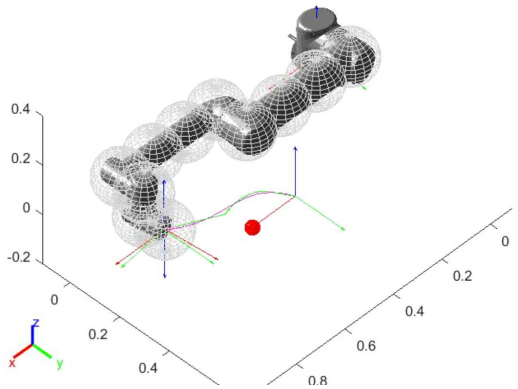


(b) Manipolatore in una posizione intermedia

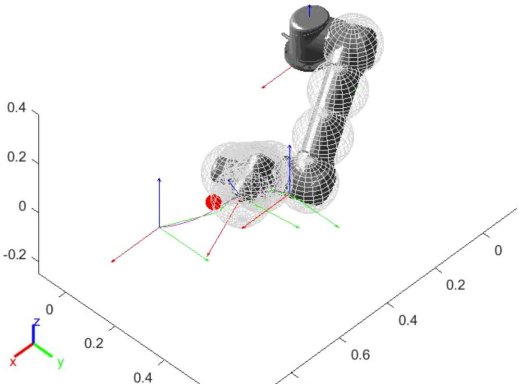


(c) Manipolatore nella posizione finale

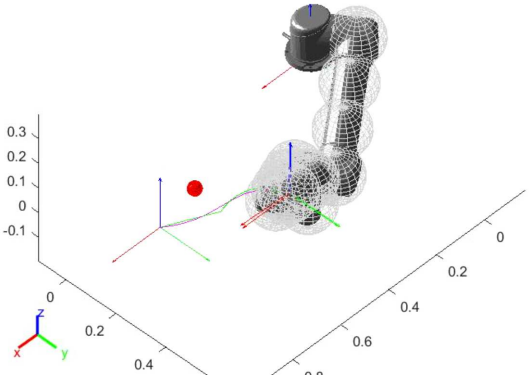
Figura 31: Simulazione moto con un ostacolo immobile



(a) Manipolatore nella posizione iniziale

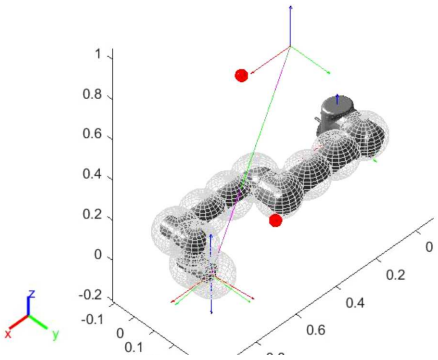


(b) Manipolatore in una posizione intermedia

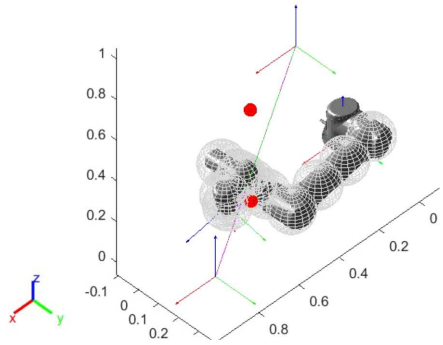


(c) Manipolatore nella posizione finale

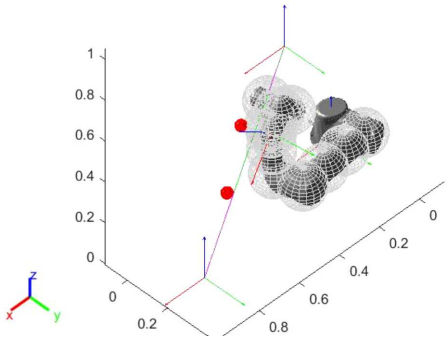
Figura 32: Simulazione moto con un ostacolo mobile



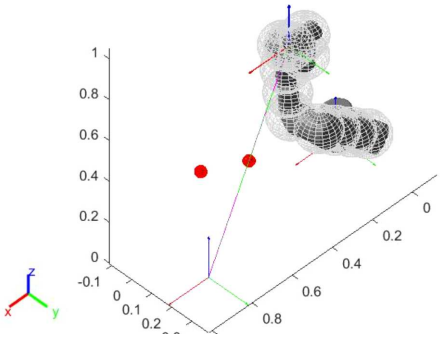
(a) Manipolatore nella posizione iniziale



(b) Manipolatore in una posizione intermedia



(c) Manipolatore in una posizione intermedia



(d) Manipolatore nella posizione finale

Figura 33: Simulazione moto con un due ostacoli mobili

7 Conclusioni

Il campo della robotica è relativamente moderno, ma, per via degli enormi vantaggi che ne derivano nell'ambito della produzione, sia da un punto di vista quantitativo che qualitativo, risulta di grande interesse industriale. Perciò, numerose aziende stanno investendo grandi quantità di denaro per automatizzare il proprio processo produttivo. Dunque, è nell'interesse dei produttori di robot fornire ai propri clienti macchinari efficienti, sicuri e capaci, in un'ottica di Industry 4.0, di collaborare con l'operatore umano.

Il soggetto di questa tesi è proprio il ramo della robotica collaborativa. Nello specifico, l'obiettivo finale è quello di pianificare il moto di un robot collaborativo in modo che questo segua il percorso più breve per raggiungere il punto di destinazione evitando i possibili urti contro eventuali oggetti, statici o mobili, che si potrebbero presentarsi lungo la traiettoria del robot. Per raggiungere questo scopo, è stata utilizzata la teoria basata sui campi potenziali proposta nell'articolo *Real-Time Strategy for Obstacle Avoidance in Redundant Manipolators*. In questa strategia viene proposto che sul dispositivo terminale presente sull'estremità del robot agiscano contemporaneamente due campi potenziali virtuali: il primo tende a posizionare l'end effector sulla traiettoria rettilinea che porta al punto finale da raggiungere (si ricorda che il percorso rettilineo che collega la posizione iniziale e terminale del robot è il più breve tra le infinite possibilità a disposizione del robot); il secondo campo virtuale è repulsivo, ovvero tende ad allontanare il dispositivo terminale da eventuali ostacoli rilevati dalla sensoristica. Il modulo e il verso di queste due forze agenti sul robot sono definite in base alla distanza tra l'ostacolo e il punto del robot più vicino ad esso, e alla distanza tra la posizione attuale dell'end effector e la posizione finale da raggiungere. Queste due grandezze vengono paragonate con tre valori, ovvero, r che è il raggio della area massima di controllo del robot; r_{min} è la distanza di sicurezza, al di sotto della quale avviene l'arresto il robot per non arrecare danno all'oggetto o alla persona che si sono avvicinati a esso; infine, r_m è una distanza intermedia tra le due precedenti. Inoltre, per evitare fenomeni di discontinuità e dare maggiore scorrevolezza al moto, si introducono tre parametri di guadagno, a_h, a_v e a_e, variabili tra 0 e 1, il cui valore si calcola in funzione della distanza tra l'ostacolo e il punto del robot più vicino allo stesso.

Alla fine del procedimento descritto finora, si ricava l'espressione della velocità nello spazio dei giunti, ovvero la velocità da imprimere a ciascuno dei giunti del robot al fine di ottenere la velocità desiderata dell'end effector nello spazio operativo. Ricavata tale velocità, si può calcolare la posizione nello spazio dei giunti all'istante di tempo $t+1$ (si ragiona per intervalli di tempo discreto) come la posizione all'istante t più la velocità che il moltiplica l'intervallo

di tempo discreto tra due istanti di controllo successivi.
Questo procedimento teorico viene poi implementato sotto forma di algoritmi su MATLAB, in modo da riuscire ad eseguire delle simulazioni per constatare la validità di quanto detto sinora.
Per eseguire le simulazioni, è stato deciso di utilizzare un modello CAD del robot collaborativo UR5 della *Universal Robots*. Gli algoritmi scritti su MATLAB consentono di visualizzare il robot, l'area monitorata dai suoi sensori e gli ostacoli esterni. Per mezzo di questi tre elementi è possibile simulare varie situazioni di funzionamento del robot. Sono state eseguite diverse prove con varie condizioni, partendo dal caso semplice del robot che durante il proprio moto rischia la collisione con un ostacolo immobile; in una seconda simulazione, con una situazione più complessa, il robot incontra un ostacolo mobile con cui rischia di urtare; infine, una situazione decisamente più complicata di quelle descritte finora, il robot rischia la collisione con due ostacoli mobili. Come risultato ottenuto in tutte queste simulazioni, appena l'ostacolo entra nella zona di rilevamento del robot, questo ha modificato il proprio percorso per riuscire ad evitare la collisione, per poi inserirsi nuovamente sulla traiettoria prestabilita una volta che l'ostacolo sia fuori dalla zona critica intorno al robot.
Dunque, si può concludere affermando che la strategia di collision avoidance basata su campi potenziali, secondo quanto emerso dalle prove eseguite su software, sia in grado di assicurare il funzionamento dei robot nei processi produttivi che richiedono la stretta collaborazione con l'operatore umano. Il robot è in grado di eseguire il proprio compito senza rappresentare un pericolo per gli oggetti o le persone che lo circondano, in quanto capace di modificare la propria traiettoria per evitare qualsiasi tipo di contatto non previsto con l'esterno.
Tuttavia, bisogna aggiungere per completezza che potrebbero verificarsi delle incongruenze tra quanto ottenuto nelle simulazioni e la realtà, in quanto il volume di controllo rilevabile dai sensori del robot potrebbe non coincidere con quello semplificato presente nel modello virtuale. Come ulteriore elaborazione del procedimento riportato, sarebbe interessante l'applicazione della teoria dei campi potenziali a un robot collaborativo la cui *base*, invece di essere posizionato su un telaio immobile, fosse connesso a un dispositivo mobile, come, ad esempio, un AGV.

Riferimenti bibliografici

[1] Lorenzo Sciavicco, Bruno Siciliano. *Robotica Industriale- Modellistica e controllo di manipolatori*. McGraw-Hill, Seconda Edizione, 2000.

[2] Cecilia Scoccia, Giacomo Palmieri, MatteoClaudio Palpacelli, Massimo Callegari. *Real-Time Strategy for Obstacle Avoidance in Redundant Manipulators*. Università Politecnica delle Marche.

[3] Stefano Chiaverini, Bruno Siciliano, Olav Egeland. *Review of the Damped Least-Squares Inverse KInematics with Experiments on an Robot Manipulators*. IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY, Vol.2, n.2, 1994.

[4] Mohammad Safeea, Pedro Neto, Richard Bearee. *On-line collision avoidance for collaborative robot manipulators by adjusting off-line generated paths:an industrial use case*. LISPEN, Arts et Métiers, University of Coimbra, Department of Mechanical Engigneering.

[5] L. Biagiotti. *Sistemi di controllo - Analisi e pianificazione delle traiettorie*. 2008/2009

[6] Massimo Callegari. *Lezioni di Meccanica delle Macchine Automatiche*. Università Politecnica delle Marche, Dipartimento di Ingegneria Industriale.

[7] Jingguo Wang, Yangmin Li, Xinhua Zhao. *Inverse Kinematics and Control of a 7-DOF Redundant Manipulator Based on the Closed-Loop Algoritm*. International Journal of Advanced Robotics System, Vol.7, No.4, 2010.

[8] Massimo Cefalo, Emanuele Magrini, Giuseppe Oriolo. *Sensor Based Task-Constrained Motion Planning using Model Predictive Control*. Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Universitò di Sapienza, Roma. IFAC (International Federation of Automatic Control) HOsting by Elsevier Ltd.

[9] Leon Zlajpah, Bojan Nemec. *Kinematic Control Algorithms for On-line Obstacle Avoidance for Redundant Manipulators*. Instituite Jozef Stefan, Ljubjana, Slovenia. Intl. Conference on Intelligent Robots and Systems, 2002.

[10] Adrià Colomé, Carne Torras. *Redundant Inverse Kinematics: Experimental Comparative Review and Two Enhancements*. Institut de Robotica i Informatica Industrial (CSIC-UPC), Barcellona, Spagna.

[11] Giacomo Palmieri. *La Robotica Collaborativa negli Impianti Industriali-Principali applicazioni e modalità di interazione con l'uomo*. Università Politecnica delle Marche.

[12] Antony A. Maciejewski, Charles A. Klein. *Obstacle Avoidance for kinematically Redundant Manipulators in Dynamically varying Enviroments*. The International Journal of Robotics Research, 1985;4 ; 109.

[13] Giulio Trigatti, Paolo Boscariol, Lorenzo Scalera, Daniele Pillan, Alessandro Gasparetto. *A new path-constrained palnning strategy for sprat pinting robots-rev.1*. The International Journal of Advanced Manufacturing Technology, 2018.

[14] Lorenzo Scalera. Andrea Giusti. Renato Vidona. Vincenzo Di Cosmo. Dominik T. Matt. Michael Riedl. *APPLICATION OF DUNAMICALLY SCALED SAFTY ZONES BASED ON THE ISO/TS 15066:2016 FOR COLLABORATIVE ROBOTICS*. University of Bozen-Bolzano, Italy. International Jpurnal of Mechanics and COntrol, Vol.21, No.01.

[15] Adrià Colomé, Carne Torras. *Closed-Loop Inverse Kinematics for Redundant Robots: Comparative Assestment and Two Enhancements*. Institut de Robotica i Informatica Industrial (CSIC-UPC), Barcellona, Spagna.