Università degli Studi di Padova

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Triennale in
Ingegneria Informatica

Analisi e applicazioni dei sensori coppia/forza nella robotica collaborativa tramite ROS

Relatore:
Prof. Stefano Ghidoni
Correlatore:
Dott. Matteo Terreran

Laureando:
Simone Peraro

Anno Accademico 2022/2023

Abstract

Con la crescente diffusione della robotica in diversi ambiti, è diventato sempre più importante sviluppare metodi e tecnologie che consentano ai robot di collaborare in modo efficiente e sicuro con gli esseri umani. I sensori di coppia/forza svolgono un ruolo cruciale in questo contesto, fornendo misurazioni accurate delle forze e delle coppie applicate durante le interazioni fisiche. Il presente lavoro di tesi si concentra sull'analisi dei sensori di coppia/forza e illustra i loro utilizzi in diverse applicazioni di robotica collaborativa sviluppate presso il laboratorio di *Intelligent Autonomous Systems* dell'*Università degli Studi di Padova*. L'obiettivo di questa tesi è quello di analizzare il funzionamento dei sensori coppia/forza in contesto collaborativo e comprendere come possono essere utilizzati per rendere più sicura ed efficiente la collaborazione tra robot e esseri umani. Vengono innanzitutto analizzate le diverse tipologie di sensore e vengono presentate le tecnologie utilizzare per sviluppare le applicazioni di robotica collaborativa, quali il sistema ROS, la libreria MoveIt, il braccio robotico collaborativo UR5 e il sensore Robotiq FT-300s. Viene poi presentata una modalità di calibrazione dei sensori coppia/forza e vengono testate le capacità di misurazione del sensore utilizzato. Infine vengono descritte le applicazioni di robotica collaborativa sviluppate individualmente e riunite in un unico *task* collaborativo finale. I risultati ottenuti in questo lavoro di tesi dimostrano come l'utilizzo del sensore nelle applicazioni di robotica collaborativa risulti vantaggioso sia nell'interazione uomo-robot sia nell'interazione tra il robot e l'ambiente circostante.

Indice

Introduzione

L'enorme crescita che il campo della robotica sta subendo in questo periodo spinge i ricercatori allo sviluppo di nuovi metodi che permettano ai robot di collaborare al meglio con l'essere umano. Questo ha portato allo sviluppo di robot detti collaborativi, in quanto il loro scopo di progettazione è quello di interagire fisicamente con le persone nello spazio circostante, sia in ambito industriale che in altre aree applicative. Affinché i robot siano in grado di lavorare in modo più sicuro ed efficiente, è di fondamentale importanza che essi siano in grado di percepire l'ambiente circostante in modo simile a come l'essere umano è in grado di percepirlo, cioè utilizzando i cinque sensi di cui le persone dispongono. Tra questi è di particolare rilevanza il tatto: ogni giorno utilizziamo il contatto e le forze di contatto per interagire con l'ambiente che ci circonda. Ecco allora che le forze di contatto hanno un ruolo cruciale anche nell'interazione dei robot con gli oggetti e le superfici presenti nell'ambiente di lavoro. In questo contesto, i sensori coppia/forza svolgono dunque un ruolo fondamentale, poiché permettono ai robot collaborativi di interagire agli stimoli esterni, migliorandone sia il lavoro che la sicurezza.

Obiettivi

Lo scopo del lavoro presentato è quello di comprendere il funzionamento dei sensori coppia/forza, anzitutto analizzando le tecnologie utilizzate da questi sensori, sia in ambito industriale che in altri contesti. Ulteriore scopo è quello di sperimentare e illustrare le possibili applicazioni del sensore mediante appositi compiti di robotica collaborativa, sviluppati presso il laboratorio di *Intelligent Autonomous Systems* [26] del *dipartimento di Ingegneria dell'Informazione* presso *l'Università degli Studi di Padova*. In primo luogo sono state testate le modalità di collegamento del sensore al computer per la lettura delle misurazioni, con lo sviluppo di un *driver* in grado di ricevere i dati. In questa fase sono state approfondite in particolare la modalità di collegamento diretto via USB e la modalità di collegamento tramite controllore del robot, evidenziando un errore nella lettura dei dati ricevuti dal sensore collegato con la seconda modalità. In seguito, il sensore è stato testato con alcuni esperimenti di calibrazione in modo da verificare il riscontro dei valori misurati con quelli attesi in base alle specifiche dichiarate del sensore. Infine sono state sviluppate alcune applicazioni di robotica collaborativa basate sull'utilizzo del sensore coppia/forza. In particolare, è stato sviluppato un metodo per la ricerca dei bordi interni ed esterni di un

contenitore, in modo da stimare la posizione del centro per il deposito di oggetti. Inoltre è stato implementato un programma in grado di utilizzare le misurazioni del sensore per seguire i movimenti di una persona trasmessi all'estremità del robot. Infine, i vari moduli sviluppati sono stati raccolti in un'unica applicazione robotica per la raccolta e il corretto posizionamento di alcuni oggetti a partire da posizioni non inizialmente note, ma fornite da una persona tramite l'utilizzo del sensore. Questi esperimenti, eseguiti utilizzando il braccio robotico collaborativo UR5 e il sensore Robotiq FT-300s sono stati implementati utilizzando il sistema software ROS e la libreria di *motion planning* MoveIt per il controllo dei movimenti del robot. Tutto il codice sviluppato è stato implementato sotto forma di *nodi* ROS, ovvero piccoli moduli in grado di funzionare parallelamente ciascuno affianco all'altro.

La tesi è suddivisa in diversi capitoli che trattano i seguenti argomenti:

- **Capitolo 1: Analisi dei sensori coppia/forza**: Vengono introdotte le principali caratteristiche e tipologie dei sensori coppia/forza e la loro importanza in diversi ambiti applicativi.
- **Capitolo 2: Introduzione a ROS**: Vengono descritti i concetti principali del sistema ROS utilizzato per interfacciarsi al sensore e al robot collaborativo.
- **Capitolo 3: Setup sperimentale**: Viene descritta la configurazione fisica e programmatica degli strumenti hardware e software utilizzati per svolgere gli esperimenti e raccogliere i dati. Vengono inoltre descritte le modalità di collegamento del sensore a ROS e le differenze constatate tra le due metodologie di lettura dei dati.
- **Capitolo 4: L'ambiente simulato**: Viene descritta la configurazione virtuale dell'ambiente attraverso l'utilizzo del framework MoveIt.
- **Capitolo 5: Applicazioni del sensore**: Viene descritto in che modo è stato utilizzato il sensore nei diversi *task* sviluppati.
- **Conclusioni e studi futuri**: Viene riassunto il lavoro presentato e vengono proposti alcuni spunti per la prosecuzione della ricerca.

Capitolo 1

I sensori coppia/forza

I sensori coppia/forza sono trasduttori in grado di convertire le forze e le torsioni lungo i tre assi spaziali in segnali elettrici di più facile utilizzo nel campi della robotica. In questo capitolo vengono illustrate le diverse tipologie di sensori maggiormente diffuse, i loro utilizzi sia in ambito industriale che in altre aree applicative.

1.1 Tipologie e funzionamento

I sensori coppia/forza permettono di misurare con precisione le forze che agiscono in corrispondenza dell'estremità del braccio robotico (chiamata anche *end effector*), lungo uno o più assi di misurazione. I sensori possono avere fino a sei assi di misurazione, tre per la forza e tre per la torsione, posti perpendicolarmente tra loro. Generalmente l'asse della forza F_z è posto uscente e perpendicolare alla superficie superiore del sensore mentre i restanti assi F_x, F_y sono orientati parallelamente alla superficie. Attorno a questi tre assi vengono misurati i momenti torcenti M_x, M_y, M_z, similmente a quanto mostrato in Figura 1.1

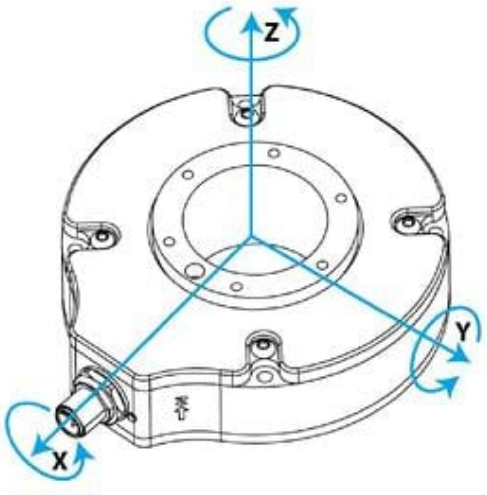


Figura 1.1: Disposizione tipica degli assi di un sensore coppia/forza.

Possiamo suddividere le tecnologie di misurazione in base al principio fisico sfruttato per la misurazione dei carichi lungo gli assi [1]. Una delle modalità di misurazione più comuni sfrutta delle resistenze per misurare la deformazione della struttura interna del sensore. La struttura interna puo' essere costituita da una croce semi-elastica in grado di deformarsi liberamente, fissata ad una struttura più esterna rigida costituita dalle pareti del sensore, come si può vedere in Figura 1.2. Le forze esterne agiscono sulla sezione interna della struttura, e attraverso alcune resistenze poste lungo gli assi della croce, è possibile misurare la deformazione della struttura in base alla variazione della conduttanza della resistenza, che allungandosi o accorciandosi determinerà una diminuzione o un aumento della tensione in uscita. Conoscendo la differenza di tensione iniziale e le proprietà dei materiali, è possibile quindi determinare l'allungamento della resistenza e calcolare la forza che è stata applicata per ottenere la deformazione. Una variazione a questo tipo di struttura prevede l'utilizzo di due superfici parallele schematizzate in Figura 1.3, che vengono fissate tra loro da alcuni supporti elastici e su cui vengono posizionate le resistenze. Nel corso del tempo si sono sviluppate diverse variazioni a questo tipo di strutture, per permettere ai sensori di essere più precisi o di sopportare carichi maggiori senza rotture, anche a costo di ridurre gli assi di misurazione [2].

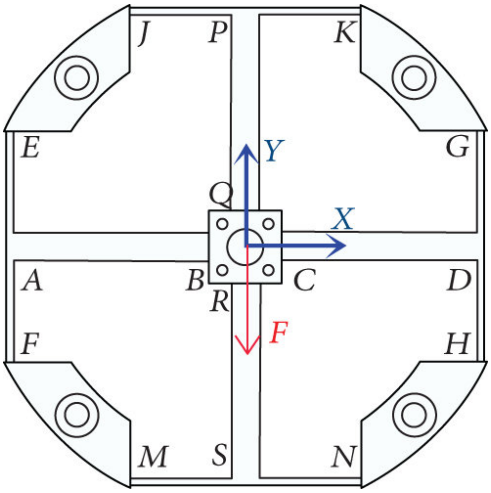


Figura 1.2: Schema della struttura a croce di un sensore coppia/forza.

Tuttavia, i sensori coppia/forza basati sulla tecnologia resistiva, sono molto sensibili sia alla temperatura che ad eventuali campi magnetici presenti nell'ambiente esterno. Anche se costantemente si ricercano metodi per limitare queste problematiche, sono stati studiate anche tipologie differenti di misurazione della deformazione. Un'altra possibile progettazione prevede l'utilizzo di condensatori in grado di misurare la distanza tra le armature. Anche in questo caso, i sensori di questo tipo sono tipicamente costituiti da una struttura più interna flessibile sulla quale vengono applicate le forze, mentre quella esterna rimane fissata, come mostrato in Figura 1.4. Durante le sollecitazioni esterne, i condensatori vengono caricati e scaricati costantemente, e si misura la loro carica residua per determinare la distanza tra le armature e la forza necessaria a deformare il materiale. Tra i vantaggi di questo approccio vi sono le ridotte dimensioni e soprattutto una mag-

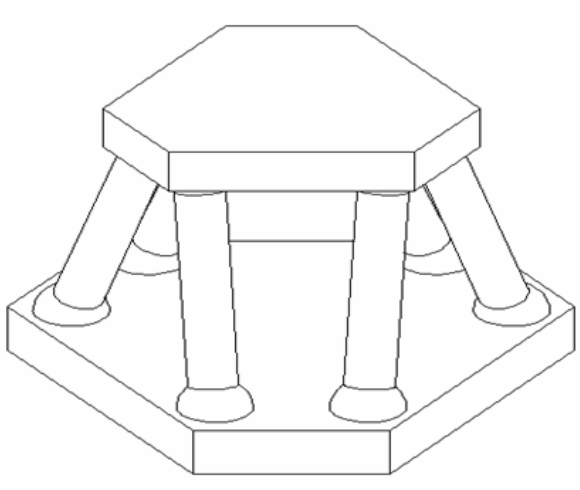


Figura 1.3: Schema della struttura a superfici parallele di un sensore coppia/forza.

gior precisione nelle misurazioni, ma ciò avviene a discapito dei costi di produzione che diventano piuttosto elevati. Si sono pertanto sviluppate diverse variazioni nella struttura interna del sensore, per garantire compattezza e solidità mantenendo bassi i costi di fabbricazione [3], nelle metodologie di funzionamento, ad esempio utilizzando un materiale dielettrico elastico compresso [4], oppure sviluppando diversi design per ridurre i costi di assemblaggio [5].

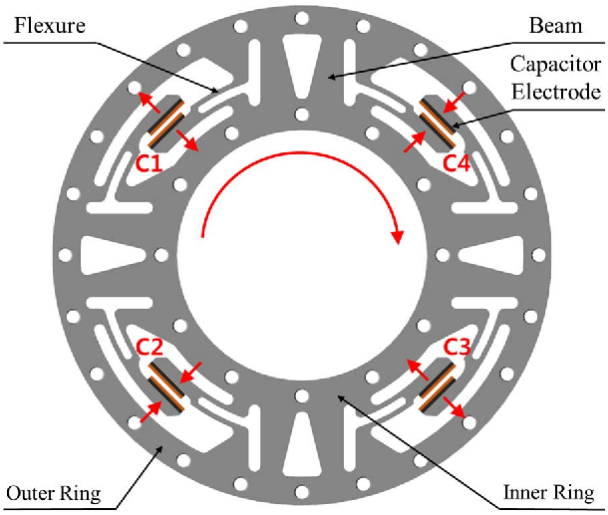


Figura 1.4: Schema della struttura a superfici parallele di un sensore coppia/forza.

I sensori di tipo capacitivo e resistivo sono le tipologie più utilizzate nell'ambito della robotica collaborativa, ma nel corso del tempo si sono sviluppate altre tecnologie per la misura delle forze. Un nuovo approccio sfrutta la proprietà piezoelettrica di alcuni materiali cristallini come il quarzo [6][7]. Durante la compressione, questi materiali generano una differenza di potenziale misurabile, tramite la quale è possibile stimare la forza necessaria a comprimere il materiale. Infine, esistono modelli di sensori ottici [8][9] che prevedono l'utilizzo di diodi LED per la misura della distanza tra le componenti del sensore. I vari LED vengono posizionati sotto ad alcuni piccoli specchi in grado di riflettere la sorgente luminosa, e il cono di luce riflesso verrà misurato fotodiodi che trasmetteranno un im-

pulso elettrico differente in base alla quantità di luce assorbita. Con questo approccio si hanno vantaggi nella semplificazione dei processi produttivi e resistenza ad interferenze elettromagnetiche, ma con una riduzione della sensibilità e affidabilità delle misurazioni.

In conclusione, i diversi approcci per la misurazione possiedono ciascuno vantaggi e svantaggi differenti, che possono meglio adattarsi ad specifici ambiti di applicazione in base a requisiti quali precisione, tempi di risposta e robustezza ambientale, che miglioreranno con il passare del tempo e con la ricerca. Il sensore coppia/forza utilizzato in questa tesi utilizza la tecnologia capacitiva [27], basata quindi su condensatori piuttosto che su resistenze, che consentono al sensore Robotiq FT-300s di aumentare il livello di precisione nelle misurazioni.

1.2 Utilizzi e applicazioni

I sensori coppia/forza analizzati fin'ora sono comunemente usati nella robotica collaborativa. In ambito industriale ad esempio vengono utilizzati durante le fasi di assemblaggio delle componenti, in modo da guidare il robot nell'inserimento delle parti meccaniche, oppure nelle applicazioni di levigatura/smerigliatura in modo da avere controllo preciso sulle forze applicate tramite gli strumenti, soprattutto quando la forma dell'oggetto non è regolare e vi è necessità di seguire le forme del prodotto. Vengono utilizzati anche nelle fasi di test, ad esempio per testare il corretto funzionamento di schermi tattili, oppure in fase di confezionamento durante la ricerca e il sollevamento delle scatole. Più interessanti sono le applicazioni in collaborazione diretta con l'essere umano, che può sfruttare i sensori per muovere il robot e insegnare le posizioni in cui deve muoversi, oppure seguirlo nei movimenti in fase di sollevamento di oggetti pesanti. Inoltre, i sensori possono essere inclusi anche nelle tecniche di controllo di sicurezza dei movimenti, ad esempio per determinare collisioni nei movimenti del robot.

L'utilizzo dei robot collaborativi con sensori di forza è molto importante anche in ambito medico. Alcune operazioni tramite dispositivi robotici permettono al medico di operare remotamente sfruttando la precisione dei movimenti meccanici [10][11]. In questi casi però non è sufficiente l'occhio del medico o di un dispositivo esterno e un ulteriore *feedback* è fornito proprio grazie a dei sensori di forza posti sulle estremità degli strumenti. Possono però anche essere utilizzati nella fase riabilitativa del paziente [12][13], assistendolo nei movimenti e favorendo il recupero della mobilità.

In ambito automobilistico, è possibile utilizzare i sensori per determinare con più accuratezza la traiettoria effettiva del veicolo durante i movimenti curvilinei [14]. Sono anche stati sviluppati metodi che utilizzano sensori di forza posti sulle ruote per analizzare lo stato di degrado del manto stradale [15]. Questi dati sono fondamentali per migliorare il controllo del veicolo in ambienti parzialmente osservabili.

I sensori possono essere anche utilizzati in ambito navale per migliorare il controllo dell'assetto della nave, misurando la spinta idrostatica ricevuta dagli stabilizzatori laterali

usata per calcolare il rollio della nave [16]. Infine, nell'ambito della ricerca sui controlli di volo dei micro veicoli aerei ispirati alla natura, vengono impiegati sensori di forza di ridotte dimensioni per raccogliere dati sulla forza e sulla pressione aerodinamica durante il volo [17][18]. Questi sensori sono progettati per essere leggeri, compatti e altamente sensibili, in modo da poter monitorare con precisione le forze applicate sulle ali o su altre superfici dell'aeromobile.

Possiamo quindi concludere affermando che gli utilizzi dei sensori sono particolarmente utili per il controllo, la sicurezza e l'efficienza nelle operazioni condotte attraverso sistemi automatici, vengono applicati anche in ambiti meno classici e innovativi nelle ricerche che richiedono nuovi approcci.

Capitolo 2

L'ambiente ROS

Il sistema ROS (Robot Operating System)[19][28] è un framework di sviluppo software che permette di applicare i concetti di un sistema operativo nel campo della robotica. Fornisce le librerie per il controllo e l'astrazione dell'hardware, per la gestione dei pacchetti, per lo scambio di messaggi tra processi nonché per loro gestione. ROS è stato parzialmente riscritto a partire dal 2015 [29], ed è ora suddiviso in due rami di sviluppo: ROS 1 e ROS 2.

2.1 Introduzione a ROS

ROS nasce nel 2007 presso la Stanford University, quando i ricercatori Eric Bergen e Keenan Wyrobek incontrarono Scott Hassan, fondatore del laboratorio Willow Garage per la ricerca nel campo della robotica. Insieme iniziarono a collaborare per costruire il robot PR2 come base di test per un software che fosse in grado di astrarre in modo agevole i differenti task che un robot dovrebbe compiere. L'obiettivo era quello di rendere semplice al programmatore l'utilizzo di procedure e algoritmi già consolidati da altri sviluppatori, in modo che potessero essere rapidamente incluse in nuovi progetti senza dover riscrivere tutte le funzioni necessarie a svolgere un determinato compito.

2.1.1 Concetti base di ROS

ROS non è propriamente un sistema operativo, ma concettualizza aspetti simili a quelli di un sistema operativo, permettendo di eseguire e gestire processi che possono accedere a diversi livelli di astrazione dell'hardware o di attuare lo scambio di informazioni tra processi, e fornisce gli strumenti per lo sviluppo e la distribuzione del codice.

ROS è un sistema modulare, in cui ogni processo è astratto con il concetto di *nodo*. Ogni nodo esegue il proprio codice autonomamente, eventualmente scambiando o ricevendo opportuni messaggi su canali di comunicazione chiamati *topics*. Per gestire tutti i nodi, durante l'esecuzione di ROS è sempre presente un nodo centrale che coordina i vari processi, denominato *ROS Master*. Il *ROS Master* coordina tutti gli aspetti di ROS in maniera centralizzata: gestisce la registrazione dei nodi in uno schema chiamato *grafo di*

calcolo (un esempio è mostrato in Figura 2.1), si occupa di collegare tra loro i nodi che necessitano di scambiarsi messaggi e gestisce uno spazio di memoria condiviso in cui i nodi possono caricare variabili utili ad altri, in una entità chiamata *ROS Parameter Server*. I nodi collaborano tra loro per gestire diversi compiti, quali l'attivazione di attuatori o la lettura dei valori dai sensori, e sono identificati univocamente da un nome.

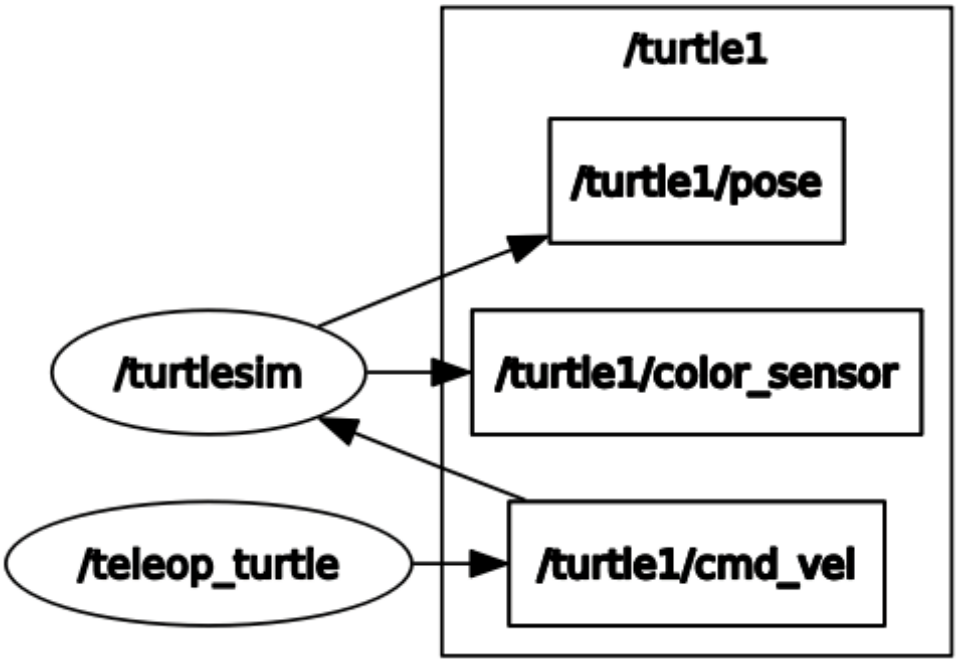


Figura 2.1: Un esempio del grafo di calcolo durante un'esecuzione di ROS.

Possiamo suddividere ROS su due livelli: il piano del grafo di calcolo e il piano del filesystem.

Grafo di calcolo ROS

Come già detto, l'esecuzione dell'ambiente ROS è schematizzabile come un insieme di nodi lungo un grafo, che vengono gestiti dal nodo principale detto *ROS Master*. I vari *nodi* rappresentano i processi eseguibili dal framework. Essi collaborano tra loro per gestire i differenti compiti, quali l'attivazione di attuatori o l'analisi dei dati ottenuti dai sensori. Ogni compito richiesto può essere compiuto da uno o più nodi, ciascuno con differenti sotto compiti da portare a termine. Ogni nodo utilizza direttamente la libreria client di ROS, e viene identificato da un nome univoco all'interno del sistema. Il nodo centrale *ROS Master* deve sempre essere mantenuto in esecuzione per poter permette ai singoli nodi di interagire. Esso ha il compito di registrare i nodi che vengono avviati, permettendo loro di scambiarsi informazioni o di accedere a servizi, un po' come accade nel Web con l'utilizzo dei server DNS. Il *ROS Master* permette ai vari nodi di interagire tra loro attraverso diversi meccanismi di comunicazione e scambio dati. La modalità di scambio principale di messaggi tra i vari *nodi* è quella che prevede l'utilizzo dei *topics*. I *topics* sono i canali di comunicazione inter-processo, che ciascun *nodo* può creare specificandone il nome e il tipo di dati che vengono scambiati. Una volta registrato un *topic* sul nodo *master*,

questo diventa disponibile a tutti i nodi in esecuzione, che possono iscriversi al canale e mettersi in ascolto o pubblicare dati, e grazie a lui i *nodi* sono in grado di mettersi in comunicazione diretta, generalmente usando il protocollo TCP/IP. Ogni *topic* è associato anche ad uno specifico tipo di *message*. I *messages* descrivono la struttura propria dei dati che vengono inviati sui vari *topics*, che possono essere costituiti da un insieme di oggetti semplici, come interi o stringhe, o oggetti più complessi quali vettori o punti. Il vantaggio è che in questo modo ogni nodo che crea e pubblica su un determinato *topic* può specificare la tipologia dei dati che verranno trasmessi in modo personalizzato in base alle proprie esigenze. I diversi *nodi* che si mettono in ascolto sul canale, riceveranno i messaggi contemporaneamente e saranno in grado di ricostruire i dati trasmessi: al programmatore basterà importare il tipo del messaggio per accedere alle informazioni di cui ha bisogno. Il framework *ROS* fornisce già diverse tipologie di messaggi semplici dalla libreria *std_msgs* quali ad esempio *String, Time, Float64*, ma anche messaggi più complessi come *Vector3, WrenchStamped* dalla libreria *geometry_msgs*. In ogni caso è sempre possibile utilizzare tipi già noti per creare messaggi più specifici.

Un altro tipo di comunicazione tra nodi è costituito dal meccanismo dei *services*. I *services* permettono ai nodi di interagire tra loro direttamente con un meccanismo di tipo *client-service*: un *nodo client* invia una richiesta con dei parametri ad un *nodo server* che elabora la richiesta e fornisce la risposta direttamente al *client*. Anche in questo caso, un *service* è associato a due tipologie di dato, che descrivono esattamente sia come deve essere composta la richiesta, che come sarà strutturata la risposta. Infine il *Master* gestisce uno spazio di memoria in cui i singoli *nodi* possono caricare alcune variabili in modo che siano disponibili anche ad altri utilizzatori. Questo è il *ROS Parameter Server*, una sorta di elenco che mappa i nomi delle singole variabili al loro effettivo valore. Qualunque *nodo* può in ogni momento leggere il valore di uno dei parametri disponibili sul *Parameter Server* oppure aggiungere una nuova variabile.

2.1.2 Namespaces

Poiché ogni *nodo, topic, service* all'interno di *ROS* è identificato dal *Master* univocamente tramite il suo nome, è necessario introdurre il concetto di *namespace*. Il namespace permette di ridurre le possibilità di conflitto tra elementi con lo stesso nome, anteponendo al nome dell'elemento il nome del pacchetto di appartenenza, o in generale una stringa univoca ben definita che potrebbe corrispondere al nome di una funzionalità implementata tramite più nodi. Ad esempio quando si crea un *topic* e si specifica il nome, è possibile fare in modo che il nome completo del canale di comunicazione appaia preceduto dal nome del *nodo* che lo ha creato, seguendo opportune regole. Le regole per la definizione dei nomi dei nodi, topic e servizi sono simili ai nomi dei percorsi nel filesystem Linux: specificando semplicemente un *nome*, l'elemento avrà *namespace* predefinito, mentre se al nome viene anteposto un *backslash /nome*, allora il nome diventa globale. In questo modo è possibile organizzare al meglio il grafo dei nodi in modo da evitare errori. In genere i *nodi*

eseguititi direttamente non hanno un *namespace* predefinito pertanto verranno identificati direttamente dal loro nome. Invece i *topics* o *services* prenderanno come *namespace* il nome del *nodo* che li ha creati, creando un identificatore del tipo */nodo/topic*. E' anche possibile raggruppare gli elementi, un po' come i file vengono organizzati in cartelle e sottocartelle. Ad esempio in Figura 2.1 è possibile osservare i due *nodi /turtlesim, /teleop_turtle* che comunicano con i *topic /pose, /color_sensor, /cmd_vel* raggruppati nel *namespace /turtle1*.

Nei casi in cui ci si dovesse trovare a dover eseguire diverse istanze dello stesso *nodo*, magari per controllare due robot contemporaneamente, può essere utile rinominare ciascuna parte del grafo in modo che i *topic* o i *servizi* relativi a ciascun robot non interferiscano tra loro. L'architettura di ROS permette di rinominare direttamente i nomi senza dover necessariamente porre modifiche al codice, ma semplicemente utilizzando l'operatore *:=* da riga di comando.

Nel caso di un *launchfile* è possibile specificare sia il *namespace* del nodo sia il suo nome utilizzando gli attributi *ns="", name=""*.

2.1.3 Filesystem di ROS

Gli elementi che compongono il filesystem di ROS sono simili a quelli che utilizza un normale sistema operativo. Possiamo infatti trovare:

- **Packages**: pacchetti di software che possono contenere uno o più nodi, sotto forma di file eseguibili scritti in *C++* o *Python*. I vari pacchetti possono inoltre contenere anche altri file, come le descrizioni di nuovi tipi di servizi o messaggi, o ulteriori file di configurazione. La compilazione del codice per ROS avviene a livello di pacchetto, che viene quindi distribuito come un'unica entità a sé stante, ma che può strettamente dipendere da altri pacchetti.

- **Package Manifest**: è il manifesto che fornisce all'ambiente ROS informazioni sul pacchetto quali il suo nome e la versione, gli autori, la descrizione, le licenze e le dipendenze da altri pacchetti.

- **Metapackages**: i metapacchetti permettono di raccogliere i pacchetti strettamente dipendenti in un unico pacchetto, in modo da rendere più agevole la distribuzione del codice.

- **Repositories**: le repositories permettono agli sviluppatori di pubblicare e tenere aggiornati i pacchetti tramite sistemi di controllo versione.

- **Message types**: sono file che descrivono come sono costruite le strutture dati dei messaggi che si scambiano i vari nodi all'interno dei topic. Alcuni tipi sono forniti direttamente da ROS, ma ogni pacchetto può definire i propri.

- **Service types**: come per i messaggi, questi file descrivono le richieste/risposte che i processi mettono a disposizione durante la loro esecuzione.

Vi sono inoltre ulteriori tipologie di file associate all'ambiente ROS con diverse funzionalità:

- **Bags**: poiché nella programmazione robotica si ha spesso a che fare con sensori e attuatori che pubblicano costantemente messaggi sullo stato dell'hardware attraverso i topic, si rende utile poter salvare rapidamente i dati trasmessi. Per ovviare questo problema sono stati introdotti i file *bags*, che permettono di registrare e successivamente riprodurre i messaggi pubblicati su un determinato topic.

- **Yaml**: spesso i nodi utilizzano specifici parametri che caricano sul *Parameter Server*. Per scorporare il processo di caricamento dei dati sul server da quello di assegnamento dei valori effettivi, è possibile utilizzare dei file di configurazione *.yaml* in cui è possibile definire nomi e valori delle variabili. Basterà quindi leggere il file e inviare i parametri al *Parameter Server* senza dover modificare alcun codice eseguibile.

- **Launchfile**: Per poter automatizzare l'esecuzione di più nodi contemporaneamente, è possibile creare dei file *.launch* che specificano i percorsi dei file eseguibili, con la possibilità di definire eventuali argomenti da passare ai singoli nodi, o di specificare un namespace specifico per il singolo nodo.

2.2 Interazione con ROS

L'interazione con il sistema ROS può avvenire principalmente in due modi: tramite linea di comando o tramite una *Graphical User Interface*.

2.2.1 Interazione da riga di comando

Interagire con ROS tramite riga di comando è il modo più semplice per gestire l'esecuzione dei nodi, ma è anche utile per monitorare i messaggi sui topic o per richiedere specifici servizi. Dopo aver scaricato i vari pacchetti, è necessario aggiungerli all'ambiente di sistema utilizzando il comando *source*. In seguito si possono eseguire i vari comandi per avviare uno o più nodi ROS:

roscore: Questo comando avvia una nuova istanza del nodo ROS *Master*, che inizializza il grafo computazionale e rimane in esecuzione per aggiungere/rimuovere nodi dall'ambiente ROS. Senza una istanza del nodo *Master* i nodi non saranno in grado di interagire tra loro tramite topic e servizi.

rosrun *package node*: E' il comando che permette di avviare uno specifico nodo all'interno di uno specifico pacchetto. Il pacchetto deve trovarsi all'interno di una

delle aree aggiunte all'ambiente di sistema, altrimenti il pacchetto non viene riconosciuto e l'esecuzione fallisce. I messaggi di log e di output del nodo verranno scritti sul terminale in cui il nodo è stato eseguiti.

roslaunch *package launchfile*: Eseguire molti nodi contemporaneamente da riga di comando è poco scalabile. Per comodità è possibile utilizzare un file *xml .launch* che contiene le istruzioni per avviare una serie di nodi contemporaneamente, permettendo di passare parametri e nomi ad ogni singolo nodo.

rosnode list|info|kill *node*: Consente di ottenere informazioni sui nodi in esecuzione, ed eventualmente arrestarli.

rostopic list|echo|pub|info *topic*: Permette di interagire con i vari topic attualmente attivi sul nodo *Master*. E' possibile ottenere informazioni su uno specifico topic oppure pubblicare/ricevere messaggi.

rosservice list|call|info *service*: Come per i topic, questo comando consente di ottenere informazioni su servizi attivi e di eseguirli.

rosbag record|play: Il comando *rosbag* consente di registrare e riprodurre i messaggi su uno o più topic, in modo che possano essere utilizzati per studio o simulazioni successive.

rospack find *package*: E' utile per mostrare la posizione di un determinato pacchetto ROS all'interno del sistema operativo.

roscd *package*: Sposta la directory di lavoro nella directory del pacchetto di destinazione.

rosed *file*: Consente di avviare un editor a riga di comando per la modifica di un file indicizzato dall'ambiente ROS.

roswtf: Esegue una scansione dei nodi topic e servizi in esecuzione per rilevare problemi o errori.

2.2.2 Interazione tramite RQT

Sebbene saper utilizzare la riga di comando sia fondamentale per comprendere il funzionamento di ROS, può rilevarsi comodo utilizzare una interfaccia grafica per schematizzare visivamente l'ambiente ROS in esecuzione. In particolare, ROS dispone di una interfaccia chiamata *rqt* composta da diversi moduli che permettono di eseguire agevolmente tutte le operazioni che si possono effettuare dalla riga di comando. Uno dei moduli più importanti per il debug dei nodi nel caso di ambienti complessi è *rqt_graph*. Questo modulo consente di visualizzare graficamente il grafo dell'ambiente di ROS, mettendo in evidenza diretta i collegamenti tra i nodi e i diversi topic attivi. In questo modo sarà più semplice riuscire a

capire se due nodi sono effettivamente collegati allo stesso topic o se sono stati commessi errori con i nomi nella fase di implementazione.

Un altro modulo utile mostrato in Figura 2.3 è *rqt_plot*, che mostra l'andamento in tempo reale dei valori ricevuti da un sensore, in modo da avere un feedback visivo immediato rispetto a quanto e cosa sta misurando un determinato sensore. L'interfaccia *rqt* presenta ulteriori numerosi moduli, che possono essere anche aggiunti separatamente, tra i quali:

rqt_topic: per visualizzare un elenco di topic e monitorare i messaggi che vengono trasmessi, similmente a quanto mostrato in Figura 2.2,

rqt_pub: per inviare specifici messaggi attraverso i topic,

rqt_service_caller: per visualizzare e avviare specifici servizi messi a disposizione sul nodo *Master*,

rqt_launch: per eseguire *launchfile* impostando parametri manualmente.

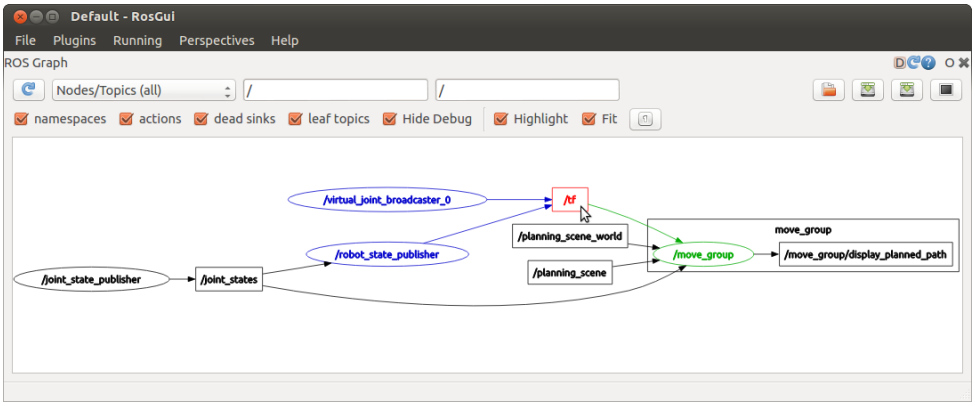


Figura 2.2: Esempio di output *rqt_graph* che mostra il grafico di nodi e topic di una esecuzione ROS.

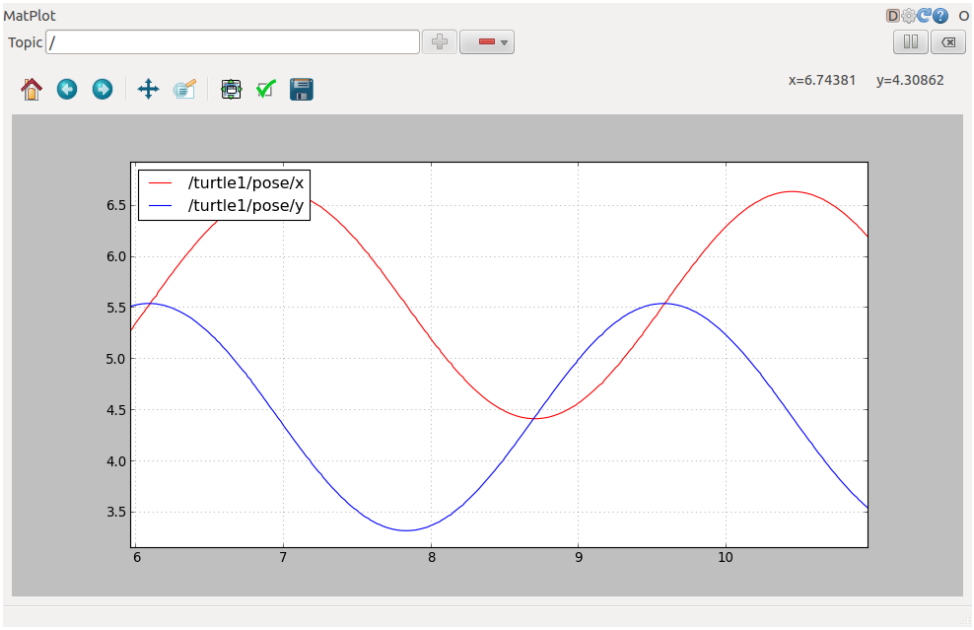


Figura 2.3: Esempio di output *rqt_plot* che mostra il grafico delle posizioni lungo un piano.

Capitolo 3

Setup sperimentale

L'ambiente di lavoro o *workcell* mostrata in Figura 3.1 è costituito dall'insieme *hardware* e *software* che sono stati utilizzati per portare avanti la ricerca nell'utilizzo del sensore. La cella di lavoro è principalmente costituita da: un piano di lavoro, un braccio meccanico collaborativo UR5 della Universal Robot montato su un supporto fisso, un sensore coppia/forza Robotiq FT300-S e il software di planning dei movimenti. E' stato successivamente utilizzato anche un *gripper* stampato in 3D per le applicazioni di presa degli oggetti, comandato tramite una scheda Arduino collegata a ROS.

3.1 Il robot collaborativo Universal Robot

Il braccio collaborativo utilizzato è un braccio collaborativo della Universal Robot, modello UR5 della serie CB3.

3.1.1 Caratteristiche del robot collaborativo

Il braccio è costituito da due segmenti lineari in grado di raggiungere una distanza pari a 850 millimetri dal centro, più una estremità in grado di ruotare e allineare il *Tool Center Point* (*TCP*) all'occorrenza. In totale, il braccio dispone di sei gradi di libertà, e ha una capacità di carico pari a 5 chilogrammi entro 35 centimetri dal centro, che scendono a 3.75 chilogrammi a distanza maggiori come mostrato nel grafico in Figura 3.2. Il braccio può essere montato in configurazioni differenti, anche appeso ad una struttura fissa o ad un carrello mobile.

3.1.2 L'unità di controllo

L'unità di controllo del robot, comunemente chiamata *control box* e posta direttamente sotto il robot, è collegata fisicamente al braccio meccanico tramite un cavo che alimenta i motori lungo i giunti e permette lo scambio di informazioni tra il robot e la *control box*. Come si vede in Figura 3.3, l'unità di controllo dispone di una serie di porte elettriche in grado di alimentare non solo il robot ma anche eventuali dispositivi che possono essere

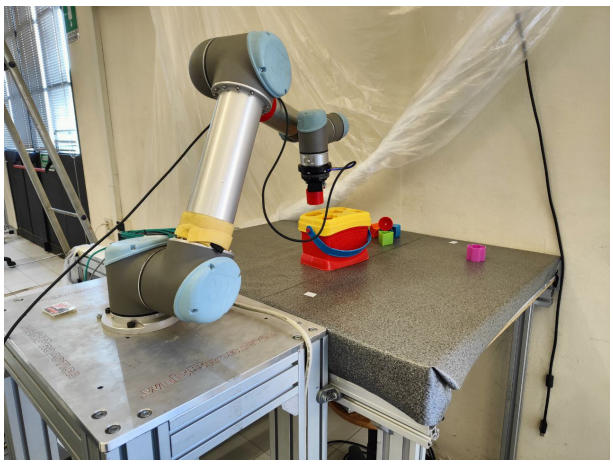


Figura 3.1: Foto del braccio UR5 con il sensore coppia/forza montato.

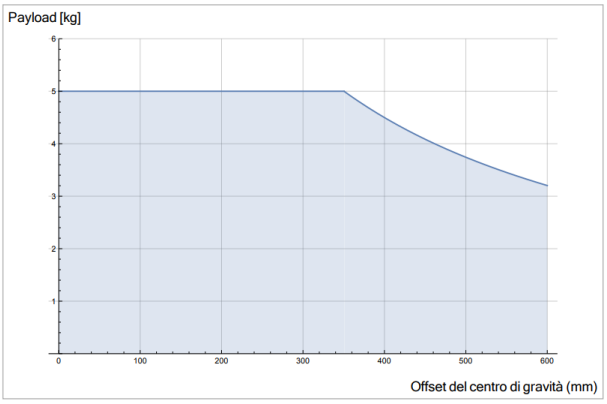


Figura 3.2: Grafico di carico del braccio UR5.

collegati, come ad esempio il sensore coppia/forza utilizzato. La *control box* consente anche di collegare al robot diversi sistemi di controllo/arresto di emergenza del robot, oltre che a consentire il collegamento di cavi per lo scambio dati in input/output sia analogici che digitali. La *control box* è a sua volta alimentata da un cavo di alimentazione ed è collegata al computer su cui è installato ROS tramite un cavo *RJ-45*.

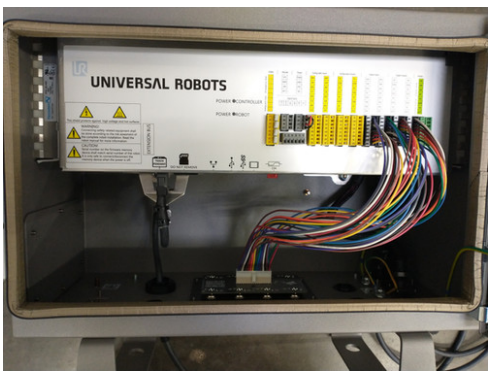


Figura 3.3: *Control box* di un robot Universal Robot.

3.1.3 Il pannello di controllo

La *control box* è collegata ad un display tattile esterno chiamato *teach pendant*. Questo pannello, mostrato in Figura 3.4, permette l'effettiva interazione con il robot attraverso comandi di movimento. Permette all'operatore di visualizzare in tempo reale lo stato del robot e di tutti gli elementi direttamente collegati alla *control box*, e consente la programmazione dei movimenti del robot sia tramite scripts che tramite sequenze di movimenti inseriti direttamente dall'interfaccia. Sull'interfaccia del display viene eseguito

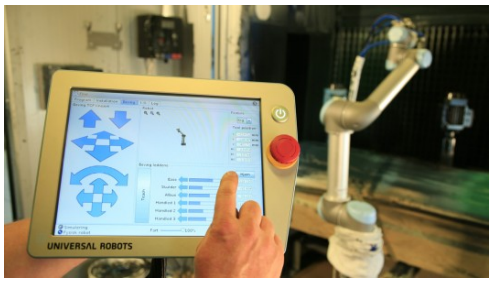


Figura 3.4: Foto del display di controllo *teach pendant*.

un software grafico chiamato *PolyScope Robot User Interface*, mostrato in Figura 3.5, che attraverso varie pagine consente di configurare, avviare e muovere correttamente il braccio collaborativo.

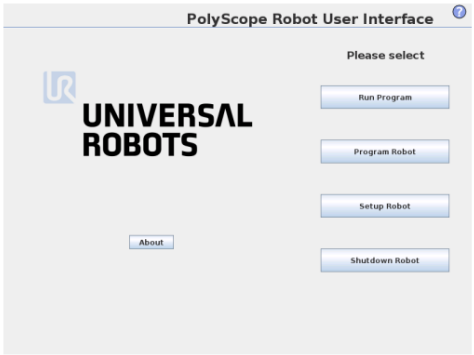


Figura 3.5: Interfaccia *PolyScope* del display di controllo *teach pendant*.

3.1.4 Collegamento con ROS

Affinché il sistema ROS sia in grado di dialogare correttamente la *control box* con l'interfaccia *PolyScope*, è necessario installare il plugin per il controllo esterno del robot *external-control.urcap*[30] sul *teach pendant*. In questo modo è possibile configurare il robot in modo che esponga una porta TCP/IP a cui il driver Universal Robot si potrà collegare.

Il driver è costituito da una serie di nodi che sono in grado di leggere e trasmettere lo stato interno del robot su specifici nodi topic ROS, creando un nodo interfaccia *ur_hardware_interface* che trasmette i comandi direttamente alla *control box* del robot. In particolare, il driver UR dispone di diversi *action-server*[31], cioè topic suddivisi in */goal,*

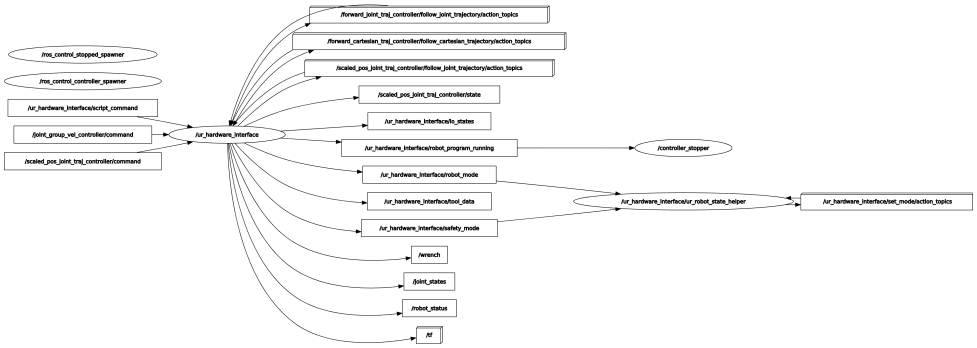


Figura 3.6: Grafo di calcolo ROS con driver UR in esecuzione.

/feedback, /result che consentono di inviare e ricevere feedback sul risultato dell'esecuzione dei comandi di movimento.

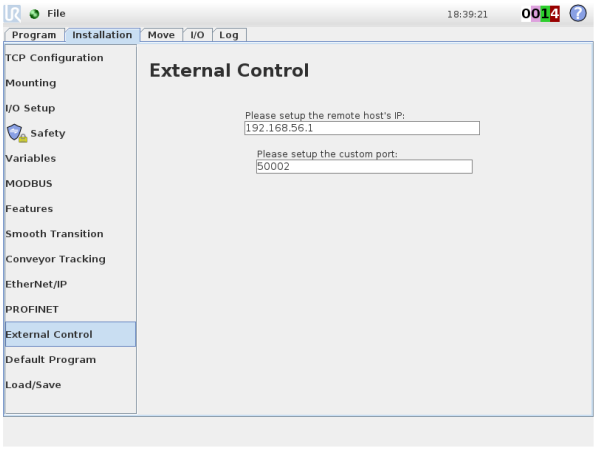


Figura 3.7: Configurazione del plugin external control.

Il driver UR, espone diverse interfacce per il controllo del robot, i *controllers*[32]:

- **joint_state_controller**: controller di sola lettura che trasmette costantemente tutte le posizioni, velocità e correnti sul topic */joint_states*
- **robot_status_controller**: controller di sola lettura per lo stato del robot (arrestato, in movimento *ecc...*)
- **speed_scaling_state_controller**: controller di sola lettura che riporta la velocità di movimento del robot impostata sul *teach pendant*
- **scaled_pos_joint_traj_controller**: controller di movimento predefinito che consente di inviare direttamente al robot la posizione desiderata dei giunti
- **joint_group_vel_controller**: controller di movimento che consente di specificare la velocità di rotazione dei giunti
- **twist_controller**: controller che consente di muovere il *Tool Center Point* secondo velocità e angoli rispetto alla base del robot.

Si noti che mentre i primi controllers di sola lettura sono costantemente in esecuzione, è possibile eseguire un solo controller di movimento per volta, in modo che più controllers non rischino di sovrapporre comandi alle stesse risorse (i giunti del robot).

3.2 Il sensore coppia/forza

Il sensore coppia/forza utilizzato in questo studio è il sensore Robotiq, FT300-S. Questo tipo di sensore è in grado di misurare coppie e forze lungo i tre assi spaziali, per un totale di 6 assi.

3.2.1 Specifiche di funzionamento

Questo specifico sensore è di tipo capacitivo[27] piuttosto che resistivo, pertanto è in grado di misurare la forza e la torsione lungo gli assi tramite la variazione della carica elettrica dovuta alla distanza tra le armature di alcuni condensatori posti all'interno della struttura, che varia durante le sollecitazioni. Per quanto riguarda la forza, il sensore è in grado di effettuare misurazioni in un intervallo di valori che va da -300 Newton a +300 Newton, mentre per la torsione il modulo è limitato a 30 Newton per metro, come mostrato in Figura 3.8. Per le misurazioni di forza, il rumore è pari a 0.1 Newton lungo i tre assi, mentre per le torsioni lungo gli assi x e y il rumore arriva a 0.005 Newton per metro. Infine, lungo l'asse z il rumore è ridotto a 0.003 Newton per metro.

SPECIFICHE	FX	FY	FZ	MX	MY	MZ
Intervallo di misurazione	± 300 N			± 30 Nm		
Capacità di sovraccarico	500%			500%		
Rumore del segnale	0.1 N			0.005 Nm		0.003 Nm
Soglia consigliata per l'individuazione dei contatti	1 N			0.02 Nm		0.01 Nm
Deviazione dell'utensile in condizioni di massimo carico misurabile	0.01 mm			0.17°		0.09°
Sensibilità al rumore esterno	Immune					
Tasso di produzione dei dati (modalità di streaming dei dati)	100 Hz					
Massa	440 g					
Protocollo di comunicazione	Modbus RTU / Data stream (RS-485)					
IP valutazione	IP65					

Figura 3.8: Tabella riepilogo specifiche del sensore Robotiq FT300-S.

Il sensore è in grado di fornire dati con una frequenza di 100 Hertz, ed è in grado di dialogare tramite le modalità *Modbus* o *datastream*.

3.2.2 Collegamento del sensore

Il sensore è collegato elettricamente alla *control box* del braccio collaborativo, che lo alimenta e permette all'interfaccia *PolyScope* di ricevere automaticamente le misurazioni dei valori di coppia/forza. Per utilizzare il sensore in questa modalità, è necessario installare

un plugin *urcap* sul *teach pendant*, chiamato *Copilot* e mostrato in Figura 3.9, che permette la programmazione del robot con l'utilizzo di alcuni nuovi blocchi di programmazione (ad esempio, è possibile programmare il robot per inserire viti o altri oggetti in appositi alloggiamenti). E' tuttavia possibile anche utilizzare il sensore in collegamento diretto con il computer, utilizzando un cavo USB che permette la comunicazione sia in modalità *Modbus* che *datastream*.

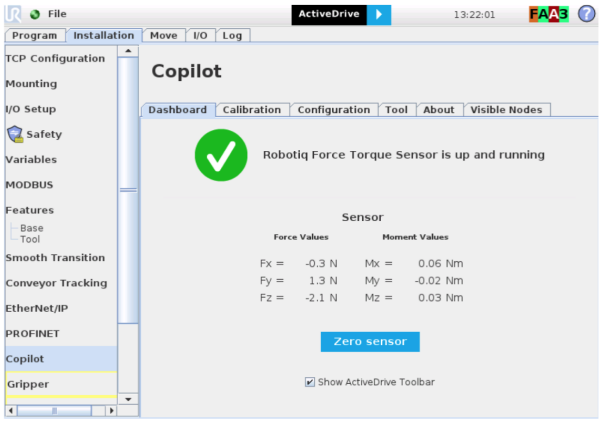


Figura 3.9: Pannello di configurazione del plugin Copilot.

Durante gli esperimenti svolti, è stato possibile testare il sensore in entrambe le modalità di collegamento. In un primo momento, il sensore è stato collegato alla *control box* del robot, e sono state utilizzate alcune funzionalità del plugin *Copilot* direttamente così come fornite dal produttore. In una fase successiva, si è cercato di sfruttare il collegamento TCP/IP tra la *control box* e il computer su cui viene eseguito ROS. Sono stati sviluppati dei nodi ROS scritti in Python e C++ il cui scopo era quello di ottenere i valori dalla *control box* e ritrasmetterli all'interno di un topic in ROS.

Utilizzando questo tipo di collegamento si sono presentati due problemi fondamentali che hanno richiesto particolare attenzione. In un primo momento non è stato chiaro in che modo venissero trasmessi i valori dall'unità di controllo al computer poiché nonostante il manuale del sensore Robotiq[33] specificasse l'utilizzo di valori di byte predefiniti per delimitare l'inizio e la fine dei messaggi, ci si è in realtà accorti che i valori delle forze misurati dal sensore venivano trasmessi tramite stringhe nella forma *(Fx, Fy, Fz, Mx, My, Mz)*. Inoltre, la lunghezza della stringa non ha dimensione fissa poiché in caso di letture negative veniva anteposto un trattino per indicare il segno negativo del valore, ed inoltre il numero di cifre poste prima della virgola variava in base alla grandezza della misurazione, aumentando ancora di più la lunghezza del messaggio. Si è reso necessario quindi sviluppare una procedura di *parsing* della stringa che fosse in grado di assecondare tutte le possibili variazioni ricevibili.

Un secondo problema si è presentato nel momento in cui sono state sfruttate le letture generate da questa modalità di collegamento. Come mostrato in Figura 3.10, ci si è accorti che i valori misurati dal sensore privo di carico soffrivano di un errore di *drift* nel tempo, probabilmente a causa di un costante tentativo del software di compensare

alcuni errori di misurazione. Nonostante la misurazione sia stata ripetuta diverse volte posizionando il sensore in modi diversi, non è stato possibile ottenere una misurazione statica consistente, e pertanto si è deciso di collegare il sensore direttamente al computer per vedere se il fenomeno si verificasse nuovamente.

La seconda modalità di collegamento prevede l'utilizzo del protocollo *Modbus* per la comunicazione con il sensore. In questo caso il sensore è stato collegato direttamente al computer tramite cavo USB e grazie al driver fornito dal produttore Robotiq e aggiornato dall'Università di Amburgo[34]. In questo modo, i dati vengono pubblicati automaticamente su uno specifico topic ROS ed è possibile interagire con il sensore attraverso alcuni servizi ROS esposti dal driver, che permettono la ri-calibrazione del sensore all'occorrenza. Sfruttando questa modalità di collegamento, le letture dei valori sono risultate molto più accurate e stabili rispetto alla precedente modalità, pertanto si è deciso di utilizzare il sensore in questa maniera.

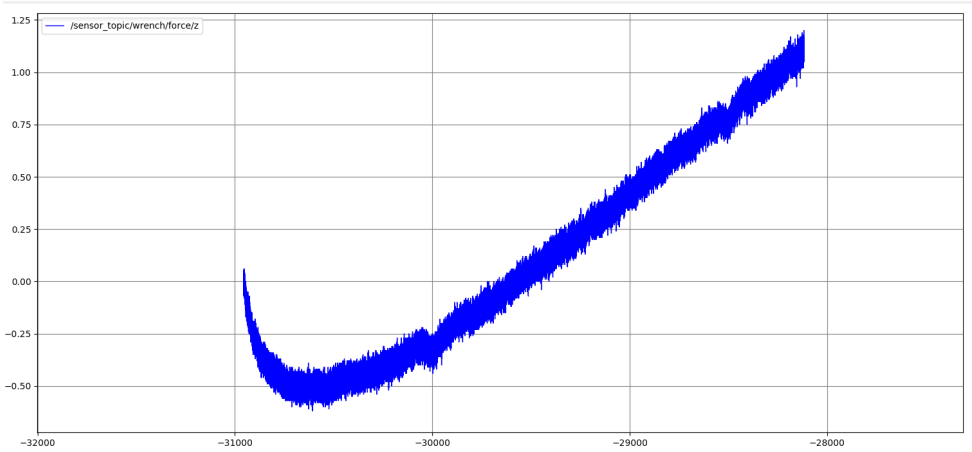


Figura 3.10: Grafico che mostra i valori misurati dal sensore a vuoto lungo l'asse z quando collegato direttamente alla control box. Si nota come al passare del tempo, i valori misurati aumentino considerevolmente nonostante nessuna forza reale sia applicata.

Capitolo 4

L'ambiente simulato

Poiché è spesso difficile immaginare i movimenti che il braccio collaborativo compie per raggiungere determinate posizioni nello spazio, è sempre utile avere anche una rappresentazione virtuale dell'ambiente in cui il robot deve muoversi. In questo modo è possibile sia visualizzare in tempo reale i movimenti del robot che proiettare il percorso che il robot seguirà per spostarsi in una certa posizione prima che vengano effettivamente compiuti. Questo è comodo quando l'ambiente intorno al robot presenta ostacoli che potrebbero compromettere l'esecuzione dei movimenti, poiché è possibile pianificare i movimenti in modo tale da evitare gli ostacoli. Per poter eseguire questo processo di *motion planning* è però necessario utilizzare apposite librerie di movimento in grado di funzionare con ROS. In particolare, in questo lavoro di tesi viene utilizzata la libreria MoveIt[20] che consente sia di visualizzare i movimenti pianificati che di selezionare graficamente la posizione desiderata nell'ambiente. In Figura 4.1 viene mostrato l'ambiente reale rappresentato virtualmente su MoveIt composto da tavolo, muro, braccio collaborativo e sensore di forza.

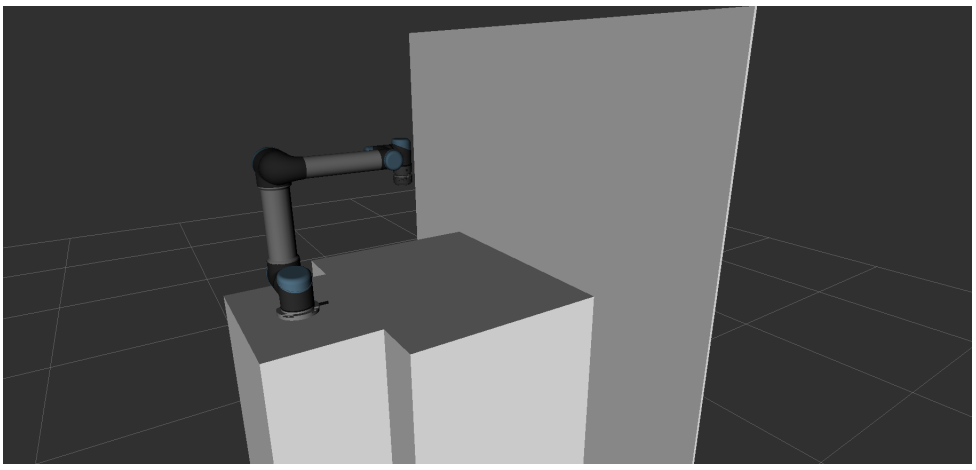


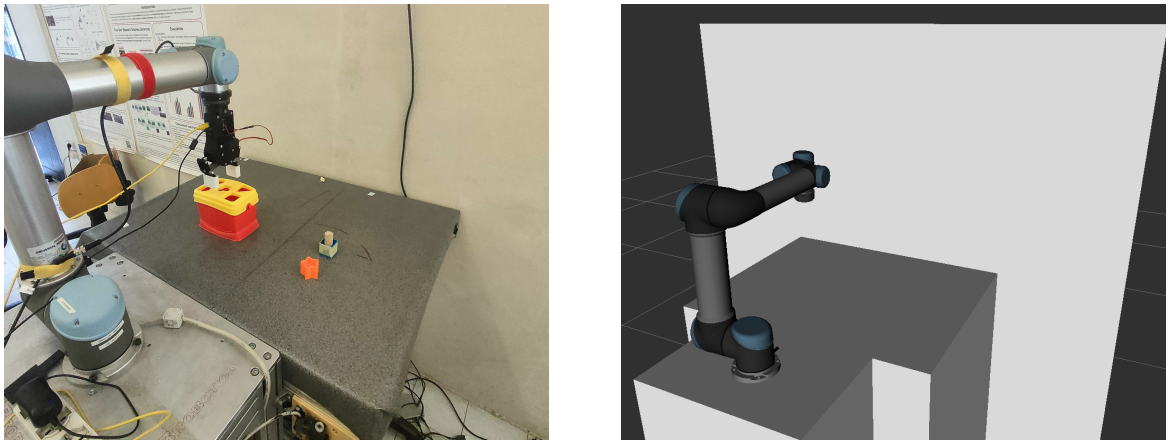
Figura 4.1: Rappresentazione grafica della *workcell* simulata.

4.1 Il framework MoveIt

Per pianificare i movimenti del robot, non è sufficiente inviare al robot una posizione nello spazio, ma è necessario calcolare volta per volta le posizioni (o le velocità) desiderate dei singoli giunti. Tuttavia è possibile utilizzare il framework MoveIt per calcolare agevolmente le varie posizioni dei giunti a partire dalla posizione desiderata nello spazio, e viceversa. Inoltre, è possibile salvare alcune posizioni del robot in modo che siano facilmente impostabili senza dover ogni volta specificare le posizioni desiderate dei giunti. Per visualizzare l'ambiente viene utilizzato il programma RViz[35] che è in grado di leggere la descrizione degli elementi dell'ambiente per mostrarla all'utente. La descrizione viene creata usando un apposito file *urdf* (Unified Robot Description Format)[36] che consiste in un elenco di tag *xml* che descrivono forme e dimensioni dei vari componenti. Per visualizzare gli elementi, il file viene caricato sul *Parameter Server* all'interno della variabile *robot-description*, a cui RViz accede per poter costruire i modelli tridimensionali.

4.1.1 Descrizione geometrica del robot

Affinché MoveIt sia in grado di controllare il robot, è necessario caricare la descrizione fisica e cinematica del robot sul sistema ROS. Nel caso dell'UR5 in oggetto, è stato possibile recuperare direttamente la descrizione geometrica già messa a disposizione dalla repository delle descrizioni dei robot Universal Robot, che viene fornita tramite file *urdf* caricato dinamicamente tramite due file *xacro* (macro *xml*). A questa iniziale descrizione, sono stati però aggiunti anche altri oggetti presenti nell'area di lavoro, ovvero il muro, la base di appoggio del robot, il piano di lavoro del robot e il sensore Robotiq, in modo che l'ambiente visualizzato fosse simile a quello effettivamente presente, come mostrato in Figura 4.2. In Figura 4.3 viene mostrato un frammento del file che contiene la descrizione dell'ambiente, in cui si vede la definizione del piano di supporto del robot e il collegamento statico tra questi.



(a) Ambiente reale (b) Ambiente simulato

Figura 4.2: Confronto tra ambiente reale e simulato in RViz

urdf > ur5_ft.urdf.xacro > robot

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://wiki.ros.org/xacro" name="workcell">
  <!--Load the macro for creating a UR5-->
  <xacro:include filename="$(find ur_description)/urdf/inc/ur5_macro.xacro"/>
  <xacro:include filename="$(find robotiq_ft_sensor)/urdf/robotiq_ft300.urdf.xacro" />

  <!--Instanciate the UR5-->
  <xacro:ur5_robot prefix="" />

  <!--Instanciate the Robotiq sensor-->
  <xacro:robotiq_ft300 parent="tool0" prefix="">
    <origin xyz="0 0 0" rpy="0 0 0"/>
  </xacro:robotiq_ft300>

  <!-- Define materials -->
  <material name="white">
    <color rgba="1 1 1 1"/>
  </material>

  <!-- Define mount link -->
  <link name="mount">
    <visual>
      <origin xyz="0 0 0.45" rpy="0 0 0"/>
      <geometry>
        <box size="0.6 0.4 0.9"/>
      </geometry>
      <material name="white"/>
    </visual>
    <collision>
      <origin xyz="0 0 0.45" rpy="0 0 0"/>
      <geometry>
        <box size="0.6 0.4 0.9"/>
      </geometry>
    </collision>
  </link>

    <!--The robot will be on the top surface of the box in the center-->
  <joint name="mount_to_robot" type="fixed">
    <parent link="mount" />
    <child link = "base_link" />
    <origin xyz="0 0 0.9" rpy="0 0 0" />
  </joint>
```

Figura 4.3: Frammento di codice della *workcell* nel file *urdf*.

4.1.2 Configurazione e lancio di MoveIt

La sola descrizione del robot non è sufficiente all'interazione del robot tramite MoveIt, in quanto è necessario che la posizione dei giunti venga costantemente aggiornata a seguito dei movimenti richiesti. MoveIt pertanto si pone in ascolto sul topic */joint_states* ed interpreta i valori ricevuti in modo da visualizzare correttamente la posizione del robot. Il driver fornito dalla Universal Robot si occupa già di inviare lo stato del robot sul topic specifico, pertanto è necessario assicurarsi che MoveIt sia in grado di accedervi. Per quanto riguarda il sensore invece, poiché il driver UR non è in grado di gestirne la posizione, è necessario utilizzare il nodo *robot_state_publisher* che ne inferisce la posizione a partire dalla disposizione definita nello schema *urdf*.

Per poter inviare i comandi al robot tramite MoveIt, è necessario che la libreria sia in grado di collegarsi correttamente ai topic *action-server*. MoveIt permette di specificare i *namespace* dei topic *action-server* tramite alcuni file di configurazione *.yaml*, come quello mostrato in Figura 4.4, dove vengono inoltre specificati i nomi dei giunti direttamente controllati e che devono corrispondere ai nomi dei giunti specificati nella descrizione del robot. In questo modo MoveIt permette tramite RViz di selezionare la posizione desiderata del robot, trascinando il punto impostato come *end effector* e permettendo di visualizzare la simulazione del movimento in tempo reale, come mostrato in Figura 4.5. Inoltre, quando

la posizione finale viene impostata tramite MoveIt, il framework verifica che non vi siano collisioni tra il robot e l'ambiente circostante nel calcolo della traiettoria richiesta.

config > ! ros_controllers.yaml

```yaml
controller_list:
  - name: "scaled_pos_joint_traj_controller"
    action_ns: follow_joint_trajectory
    type: FollowJointTrajectory
    joints:
      - shoulder_pan_joint
      - shoulder_lift_joint
      - elbow_joint
      - wrist_1_joint
      - wrist_2_joint
      - wrist_3_joint

```

Figura 4.4: Sezione del file *ros_controllers.yaml* che gestisce il collegamento tra MoveIt e i topic *action-server* relativi al controllore *scaled_pos_joint_traj_controller*.

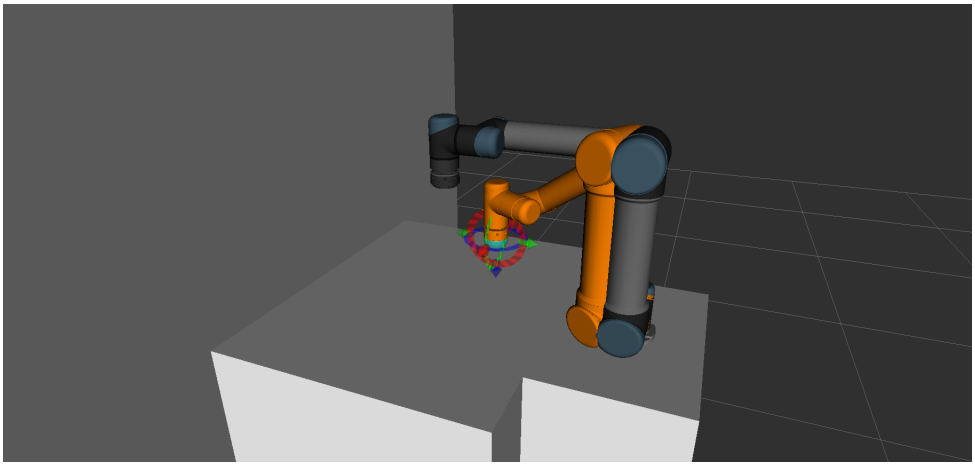


Figura 4.5: Schermata di RViz con posizione attuale del robot (in grigio) e posizione desiderata tramite MoveIt (in arancione).

Affinché MoveIt sia in grado di avviarsi e leggere correttamente la descrizione del robot e dell'ambiente, è necessario creare un pacchetto ROS contenente tutti i file di configurazione *.yaml*, tramite uno strumento chiamato *moveit setup assistant* [37], che si occupa di generare anche un file *srdf* (*Semantic Robot Description File*)[38] in grado di memorizzare anche altre ulteriori informazioni utili quali eventuali posizioni preimpostate per il robot o la presenza di giunti passivi o anche la definizione dell'elemento *end effector*. Alla creazione del pacchetto, vengono creati anche dei file di lancio *.launch* che consentono di caricare i nodi necessari all'avvio di MoveIt e di impostare le configurazioni definite tramite i file *.yaml*, quale ad esempio la configurazione del controller necessario a muovere i vari giunti del robot, mostrata in Figura 4.4.

Capitolo 5

Valutazione delle capacità del sensore

In questo capitolo vengono presentate alcune esperienze progettate per la valutazione delle capacità del sensore coppia/forza Robotiq. In primo luogo viene illustrata una procedura per poter effettuare la calibrazione del sensore, necessaria per effettuare misurazioni corrette delle forze e delle torsioni realmente agenti sul sensore. In seguito, vengono illustrati alcuni esperimenti il cui scopo è quello di valutare precisione del sensore Robotiq. In particolare, sono stati sviluppati alcuni esperimenti per la stima della massa di un oggetto e della viscosità di un fluido, con l'obiettivo di testare le capacità del sensore e di comprendere quali siano i limiti fisici dello strumento.

5.1 La calibrazione del sensore

Prima di poter misurare accuratamente le forze applicate sul sensore, è opportuno calibrare il sensore ogni volta che viene utilizzato. Questa procedura è necessaria poiché il fissaggio del sensore sul braccio robotico e il ri-orientamento del sensore nello spazio durante i movimenti del robot comportano una serie di errori di misurazione dei valori dovuti alla massa non trascurabile del sensore e del *tool* eventualmente utilizzato. Attraverso il processo di calibrazione, è possibile eliminare queste misurazioni fissando un riferimento ai valori misurati. La calibrazione è anche necessaria nel caso in cui bisogna effettuare movimenti che comporterebbero il ri-orientamento del sensore nello spazio, poiché le masse agganciate al sensore verrebbero costantemente misurate. Per poter stimare il vettore forza peso agente sul sensore e l'errore aggiunto sistematicamente ai valori misurati, è necessario posizionare il braccio robotico in modo da allineare la direzione del vettore forza peso con ciascuno degli assi di misurazione del sensore, in modo da riuscire a calcolare il modulo del vettore per poterlo compensare durante i movimenti del robot [21][22]. Una volta che il braccio robotico è stato posizionato in tali posizioni, la calibrazione è eseguita stimando il valore della forza applicata ed il centro di massa; una volta stimati tali valori, è possibile calcolare i valori di bias della forza peso da utilizzare per compensare le mi-

surazioni lungo ciascun asse del sensore. Nelle sottosezioni seguenti vengono presentati i dettagli per la stima dell'errore di misurazione lungo ogni asse del sensore e per la stima della massa e della posizione del centro di massa.

5.1.1 Stima dell'errore di misurazione per la forza e la torsione

Prima di poter stimare con accuratezza il valore del modulo del vettore forza peso, è necessario stimare l'errore di misurazione misurato su ciascun asse, posizionando il sensore in modo che la direzione del vettore forza peso sia parallela ad uno dei tre assi del sensore. Per esempio, posizionando il sensore con l'asse z orizzontale, è possibile orientare l'asse y in alto o in modo da allinearlo al vettore forza peso $\vec{P} = m\vec{g}$, come mostrato in Figura 5.1. In questo modo, i due assi x, z risultano scarichi da qualsiasi forza, e i valori misurati, se non nulli, sono dovuti all'errore di misurazione del sensore. Possiamo prendere nota devi valori misurati lungo x e z e utilizzarli successivamente per calcolare il valore medio dell'errore lungo questi assi. Orientando nuovamente il sensore nello spazio, ad esempio allineando l'asse x con il vettore forza peso, è possibile prendere nota degli errori di misurazione lungo gli assi y e z. Se ripetiamo la procedura allineando anche l'asse z al vettore forza peso, siamo in grado di calcolare l'errore di misurazione medio lungo ciascun asse, che chiameremo rispettivamente $\tilde{f}_x, \tilde{f}_y, \tilde{f}_z$. Tramite questi valori è possibile calcolare l'errore complessivo di misurazione del sensore, dovuto alla somma degli errori lungo i tre assi:

$$\tilde{f} = \sqrt{\tilde{f}_x{}^2 + \tilde{f}_y{}^2 + \tilde{f}_z{}^2}$$

Possiamo ripetere la misurazione per misurare l'errore relativo ai momenti torcenti. In questo caso però è necessario riflettere sul fatto che solo l'asse parallelo al vettore forza peso non subisce torsioni dovute alla massa attaccata al sensore, mentre gli altri due assi misureranno delle torsioni dovute alla forza peso agente sul sensore. Pertanto è necessario misurare di volta in volta solo il momento relativo all'asse parallelo al vettore forza peso, ad esempio il momento M_y in Figura 5.1. Anche in questo caso, dopo aver misurato tutti gli errori relativi ai momenti torcenti di ogni asse a vuoto, è possibile calcolare l'errore complessivo di misurazione del momento torcente:

$$\tilde{t} = \sqrt{\tilde{t_x}{}^2 + \tilde{t_y}{}^2 + \tilde{t_z}{}^2}$$

Ora che i vettori relativi agli errori di misurazione della forza e della torsione sono stati calcolati, è possibile sottrarli sistematicamente ai valori misurati dal sensore, eliminando quindi le forze residue misurate dal sensore.

5.1.2 Stima della massa e del centro di massa

Una volta calcolato il *bias* sistematico per i valori di torsione e forza misurati dal sensore, possiamo procedere con il calcolo per la stima delle forze e torsioni dovute al peso del

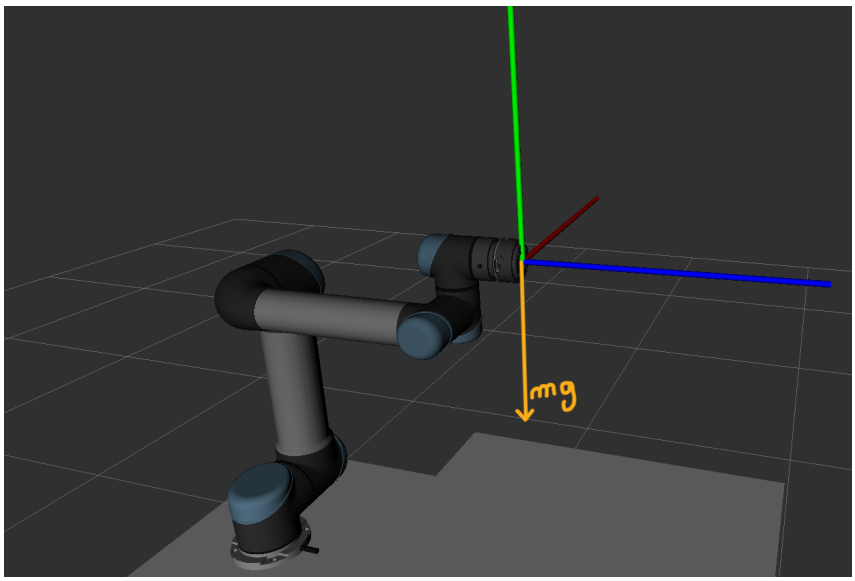


Figura 5.1: Rappresentazione del vettore forza peso agente sul sensore.

sensore e dell'eventuale *tool* agganciato ad esso. In questo caso sarà necessario misurare la forza agente lungo l'asse posto parallelo al vettore forza peso, mentre contemporaneamente è possibile misurare il momento torcente relativo agli altri due assi perpendicolari.

A questo punto possiamo procedere con il calcolo per la stima della massa da utilizzare per compensare il vettore forza peso. Lungo ciascun asse verrà misurato:

$$\begin{aligned} F_x &= -m_1 g + \tilde{f}_x \quad \rightarrow \quad m_1 g = -F_x + \tilde{f}_x \\ F_y &= -m_2 g + \tilde{f}_y \quad \rightarrow \quad m_2 g = -F_y + \tilde{f}_y \\ F_z &= -m_3 g + \tilde{f}_z \quad \rightarrow \quad m_3 g = -F_z + \tilde{f}_z \end{aligned}$$

e possiamo stimare la massa calcolandone una media tra le misure degli assi:

$$\Rightarrow \quad \hat{m} = \frac{(m_1 + m_2 + m_3)g}{3g}$$

Possiamo ora usare la massa stimata $\hat{m}$ per calcolare il centro di massa del sistema composto da *tool* e sensore. Se supponiamo che il sistema sia simmetrico rispetto all'asse z, ci basterà calcolare la distanza del centro di massa lungo l'asse z sfruttando le misurazioni ottenute rispetto agli assi x, y:

$$\begin{aligned} M_x &= \hat{m} g \times \vec{r} + \tilde{t}_x \\ M_y &= \hat{m} g \times \vec{r} + \tilde{t}_y \\ \Rightarrow \quad \hat{r}_{cm,z} &= \frac{(M_x - \tilde{t}_x) + (M_y - \tilde{t}_y)}{2\hat{m}g} \end{aligned}$$

Sfruttando quindi la massa $\hat{m}$ e la distanza del centro di massa $\hat{r}_{cm,z}$ lungo l'asse z, è possibile orientare il braccio con il sensore nello spazio e ricostruire momento per momento

il vettore forza peso agente sul sensore per sottrarlo ai valori misurati in modo da ottenere il valore reale delle forze agenti sul sensore.

5.2 Esperimenti per la valutazione della precisione del sensore

Durante l'utilizzo del sensore coppia/forza Robotiq in modalità di collegamento diretto via USB, è possibile utilizzare la funzionalità di azzeramento già resa disponibile del sensore, che consente in ogni momento di azzerare i valori misurati dal sensore lungo ogni asse. Questa procedura è raccomandata in tutte le applicazioni che richiedono la ricerca di un *contatto* in un momento specifico dell'applicazione, quando l'orientamento del sensore sul braccio robotico è noto. In questo modo è possibile eliminare gli errori di misurazione e l'influenza del vettore forza peso agente sul sensore, ed è possibile ricercare accuratamente le forze definite come soglia nell'applicazione robotica, a patto però di non orientare diversamente il sensore nello spazio, cosa che richiederebbe invece una calibrazione completa per la stima delle massa e del centro di massa agenti sul sensore.

Per testare sperimentalmente questa funzionalità e le capacità del sensore, è stato sviluppato un esperimento per la valutazione dell'azzeramento del sensore in modo dinamico [23], utilizzando un filo e un oggetto di massa nota. In seguito, per valutare la precisione delle misurazioni del sensore è stato sviluppato un esperimento che prevede l'utilizzo del sensore per stimare la viscosità di un fluido misurando i valori di torsione ottenuti ruotando un cilindro all'interno del fluido.

5.2.1 Esperimento del taglio del filo e stima della massa di un oggetto

L'esperimento prevede l'utilizzo di un filo e di una massa nota abbastanza grande da essere misurabile dal sensore. Dopo aver agganciato la massa al sensore, ad esempio come mostrato in Figura 5.2, ed aver calibrato la misurazione, il filo viene tagliato in modo da sollecitare negativamente il sensore in maniera impulsiva, e ci si aspetta che il valore a riposo misurato dal sensore dopo il taglio del filo, dovrebbe coincidere con il valore della massa dell'oggetto agganciato.

Analizzando i dati ottenuti in seguito alla ripetizione dell'esperimento lungo i differenti assi del sensore, è possibile stimare la massa dell'oggetto a partire dalla variazione istantanea della forza misurata lungo ciascun asse, come mostrato in Figura 5.3. Considerando la media della forza percepita negli istanti precedenti e successivi al taglio del filo, nelle tre misurazioni lungo i tre assi è stata misurata una massa pari a $167, 156, 148$ grammi, da cui è possibile stimare una massa pari a 157 grammi a fronte della massa reale pari a 155 grammi, e quindi concludendo che il sensore sia in grado di rilevare correttamente le forze ad esso applicate con discreta precisione.

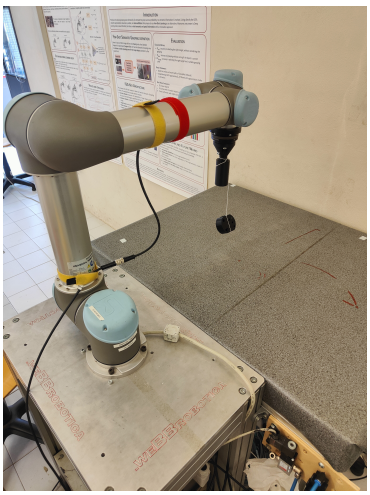


Figura 5.2: Setup sperimentale per la misurazione del peso lungo l'asse z.

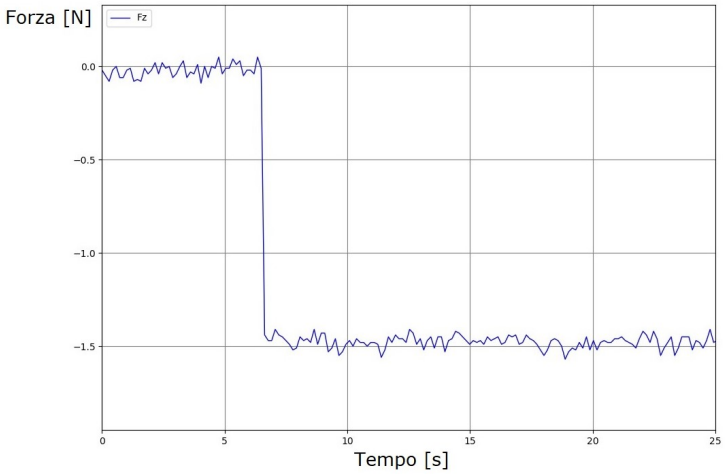


Figura 5.3: Grafico misurazione del peso lungo l'asse z.

5.2.2 Esperimento per la stima della viscosità di un fluido

Un ulteriore esperimento che è stato sviluppato con l'obiettivo di testare la precisione delle misurazioni del sensore prevede l'utilizzo dei valori di torsione misurati per la stima della viscosità di un fluido. In particolare, è stato montato un cilindro di raggio r pari a $2cm$ sull'*end effector* del robot, che è stato fatto ruotare a velocità costante all'interno di un vasetto di burro di arachidi, costruendo un viscosimetro [24] e misurando i valori di torsione lungo l'asse z. Immergendo il cilindro all'interno del contenitore del burro di arachidi come in Figura 5.4, è stato possibile misurare la resistenza percepita dal sensore in modo da poter risalire alla viscosità η della sostanza, ottenuta considerando la velocità angolare $\vec{\omega}$, il raggio del cilindro immerso nel burro di arachidi r, il raggio R del vasetto contenente il burro di arachidi e l'altezza h del cilindro immerso, come schematizzato in Figura 5.5. Per convertire la torsione misurata in viscosità è stata ricavata la seguente formula:

$$\eta = \frac{\vec{M}}{2\pi\vec{\omega}hr^3}(R - r)$$

dove l'altezza $h = 0.075m$, $r = 0.02m$, $R = 0.035m$ e $\vec{\omega} = 0.8rad/s$

Figura 5.4: Setup dell'esperimento per il calcolo della viscosità del burro di arachidi.

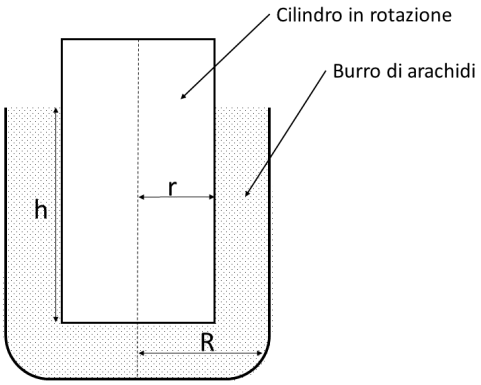


Figura 5.5: Schema del viscosimetro costruito con il cilindro in rotazione all'interno del vasetto di burro di arachidi.

Nel corso delle varie misurazioni effettuate, è stato molto difficile arrivare a costruire il procedimento esatto in grado di riprodurre esattamente le condizioni di partenza dell'esperimento, in quanto l'elevata viscosità del burro rendeva difficile ripetere la misurazione senza aver prima atteso un certo tempo necessario al materiale per assestarsi nuovamente. Nonostante comunque fosse chiaro che il burro di arachidi diventasse sempre meno viscoso con il passare del tempo, a causa della temperatura dell'ambiente e del lavoro assorbito durante le varie rotazioni, è stato possibile ripetere l'esperimento diverse volte, in cui sono stati ottenuti i risultati riportati nella Tabella 5.1.

Nonostante le difficoltà di misurazione, i risultati ottenuti sono in linea con le aspettative sulla viscosità del burro di arachidi, che si attesta in una fascia tra i 150.000 e 250.000 $mPa \cdot s$.

In conclusione possiamo dire che i due esperimenti proposti hanno ottenuto risultati coerenti con le capacità dichiarate del sensore, che è stato in grado sia di misurare il

N° misurazione	Viscosità misurata $[mPa \cdot s]$
1	185.000
2	165.000
3	149.000
4	145.000
5	150.000
6	143.000

Tabella 5.1: Stima della viscosità del burro di arachidi nelle varie misurazioni.

peso dell'oggetto appeso tramite filo, sia la viscosità del burro di arachidi nonostante la misurazione di quest'ultima risulti molto più complessa e difficile da calcolare in questa modalità. Inoltre nell'esperimento della stima del peso di un oggetto è stato possibile verificare il corretto funzionamento della capacità del sensore di azzerarsi correttamente una volta che il peso è stato agganciato al sensore.

Capitolo 6

Usi e applicazioni del sensore coppia/forza

In questo capitolo vengono presentati esempi di applicazioni d'uso del sensore coppia/forza Robotiq che sono stati sviluppati durante la fase sperimentale di questo lavoro di tesi. In particolare, le applicazioni proposte si concentrano sull'uso del sensore coppia/forza sia come strumento per il controllo diretto del robot da parte dell'operatore, sia come sensore in grado di fornire *feedback* al robot collaborativo per il raggiungimento di uno specifico *goal*. Nel corso di questo capitolo vengono prima illustrati due applicazioni distinte: la prima descrive come è stato possibile utilizzare i valori misurati dal sensore per muovere il robot collaborativo assecondando i movimenti dell'operatore, mentre la seconda applicazione descrive come è possibile utilizzare un sensore per posizionare correttamente gli oggetti all'interno di un contenitore. Infine, in questo capitolo viene presentato un esempio completo di robotica collaborativa che utilizza entrambi i *task* descritti in precedenza per realizzare un'applicazione di *pick and place* collaborativa. Il software è stato sviluppato e suddiviso in diversi nodi ROS, sia in linguaggio Python che C++ [1], e la sua modularità ha permesso di poter riunire le differenti parti in un'unica esperienza conclusiva in cui i vari nodi ROS collaborano per portare a termine un unico compito.

6.1 Applicazione per il controllo manuale del robot

In diverse occasioni di sperimentazione con il braccio meccanico, ci siamo ritrovati a dover spostare manualmente il robot in alcune posizioni più agevoli. Questo accade spesso quando non è possibile conoscere a priori la posizione spaziale desiderata del robot, e muoverlo attraverso i meccanismi automatici forniti dal controllore del robot o dalla libreria MoveIt richiede un certo tempo a causa dei diversi tentativi da effettuare per raggiungere la posizione finale. Ecco allora che l'utilizzo del sensore per il movimento del robot risulta una modalità alternativa vantaggiosa: attraverso il feedback di forza misurato dal sensore, l'operatore può imprimere una forza sull'end-effector del robot (ad esempio tirarlo a se)

[1] Tutto il software sviluppato è stato pubblicato all'interno della repository GitHub [39]

che è convertita in un comando di velocità per muovere il robot seguendo l'operatore. Le misurazioni ottenute dal sensore vengono ricevute dal nodo ROS e trasformate in un vettore velocità utilizzato come input per il controllo dei movimenti del robot. In tal modo, l'operatore può muovere direttamente il braccio collaborativo nelle posizioni desiderate, agendo con più o meno forza a seconda della distanza dalla posizione desiderata, come si vede in Figura 6.1.

Questo tipo di controllo è reso possibile utilizzando uno tra i controllori messi a disposizione da Universal Robots, il controllore *twist_controller*, che tramite un apposito *topic* in ROS, riceve in ingresso un vettore velocità che viene utilizzato per muovere l'*end effector* del robot. Inoltre, per rendere più robusto il controllo tramite forza, i valori letti non vengono passati direttamente al controllore, ma vengono applicate alcune operazioni per rendere il controllo più stabile. Ad esempio utilizzando il valore medio dei pacchetti piuttosto che direttamente gli ultimi valori misurati, è stato possibile rendere il controllo tramite forza molto più stabile e graduale. Nel corso dei diversi test effettuati, è stato riscontrato che utilizzare i valori di forza misurati negli ultimi 20 pacchetti ricevuti, sebbene comporti un lievissimo ritardo, consente di eliminare le vibrazioni del robot dovute agli eventuali rapidi cambiamenti nei valori di forza misurati. In seguito il valore medio della forza misurata viene diviso per un fattore α, che consente di rallentare i movimenti del robot in modo da renderli più precisi. Nel corso dei vari test effettuati, è stato possibile determinare sperimentalmente il valore del parametro α, ed è stato riscontrato che con α pari a 175 il controllo risulta abbastanza preciso senza che l'operatore debba imprimere forze troppo grandi per muovere il braccio. Infine, per eliminare movimenti non voluti, è stata determinata sperimentalmente una soglia di ascolto per i valori di foza, posta pari a $5N$. Durante le varie fasi di sviluppo di questo lavoro di tesi, si è riusciti a utilizzare frequentemente questo tipo di controllo in diversi momenti di interazione con il robot collaborativo, a rafforzare il fatto che questo tipo di controllo consente un'interfaccia con il robot molto più diretta e intuitiva rispetto al posizionamento tramite la libreria MoveIt o i comandi integrati del robot.

Questo tipo di applicazione del sensore è stata utilizzata anche nell'ambito del progetto *DrapeBot* [25], in cui un operatore e un robot collaborano per il posizionamento di alcuni drappi in fibra di carbonio lungo una superficie. In questo caso, il robot era dotato di alcune ventose in grado di sollevare un lembo della fibra di carbonio, mentre l'altro lembo era in mano all'operatore; durante il trasporto di tale materiale, i movimenti dell'operatore erano direttamente trasmessi al robot tramite la lettura delle forze e momenti torcenti applicate al robot, che erano poi convertiti in comandi di velocità per permettere al robot di seguire l'operatore, come mostrato ad esempio in Figura 6.2. A differenza dell'applicazione precedente, è stato necessario includere anche le misurazioni della torsione lungo l'asse z, in modo che l'operatore fosse in grado di orientare liberamente il drappo di fibra di carbonio. La lettura dei valori di torsione e la conversione in velocità angolare utilizzata come input per la rotazione dell'*end effector* avviene in modo simile a quanto

già descritto per i valori di forza nel paragrafo precedente: viene calcolata le torsione media utilizzando gli ultimi 20 pacchetti ricevuti e viene divisa per un fattore β determinato sperimentalmente pari a 20, mentre la soglia di rilevamento è stata impostata a $0.1N$.

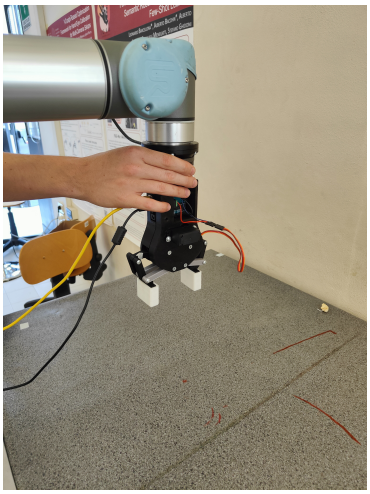


Figura 6.1: Esempio di controllo del movimento del braccio meccanico tramite l'utilizzo del sensore coppia/forza.

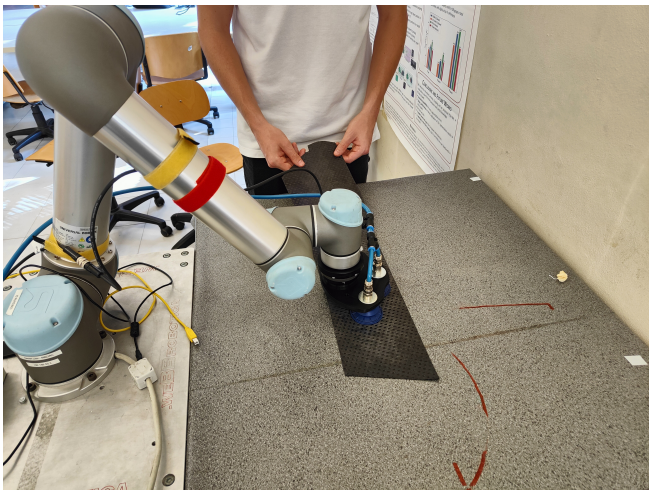


Figura 6.2: Esempio di utilizzo del sensore coppia/forza per il movimento delle fibre di carbonio nell'ambito del progetto DrapeBot.

6.2 Utilizzo del sensore in applicazioni *pick and place*

Uno tra i compiti più comuni che possono essere svolti dai robot collaborativi industriali riguarda l'utilizzo dei bracci meccanici per il posizionamento di vari elementi meccanici in specifiche posizioni. Questo tipo di problema appare essere semplice se si dà per scontato di conoscere la posizione esatta in cui si desidera posizionare l'oggetto, ma nella realtà questo dato non è sempre fornito o in generale non è abbastanza preciso. Inoltre, anche se fossimo a conoscenza con assoluta precisione del punto desiderato, diversi fattori

potrebbero impedire l'allineamento corretto dell'oggetto con la posizione finale desiderata. Ad esempio nel corso degli esperimenti effettuati è accaduto spesso che la pinza utilizzata per afferrare gli oggetti non riuscisse a centrare correttamente l'oggetto, causando quindi un disallineamento tra questo e la posizione finale desiderata. Ecco allora che è possibile effettuare un paragone con le sensazioni fornite dal corpo umano nel momento in cui lo stesso compito deve essere portato a termine da una persona, che quindi utilizzerà la vista e il tatto per correggere gradualmente la posizione dell'oggetto fino ad allinearlo correttamente. In modo simile dunque, si può utilizzare un sensore coppia/forza per cercare di colmare la lacuna dovuta dalla mancanza di informazioni sensoriali in grado di fornire un *feedback* sul posizionamento dell'oggetto. Nei diversi *task* che sono stati sviluppati durante la fase sperimentale di questa tesi, l'obiettivo perseguito è stato quello di riuscire a inserire al meglio alcune forme geometriche all'interno dei rispettivi alloggi caratterizzati dalla stessa forma, partendo da una posizione non esattamente allineata alla posizione finale desiderata. Un esempio delle formine e del contenitore con i vari alloggi sagomati è mostrato in Figura 6.3. Per cercare di raggiungere questo obiettivo, sono stati sviluppati due approcci differenti: uno basato sulla ricerca dei bordi del contenitore ed uno basato su una ricerca con movimento a spirale.

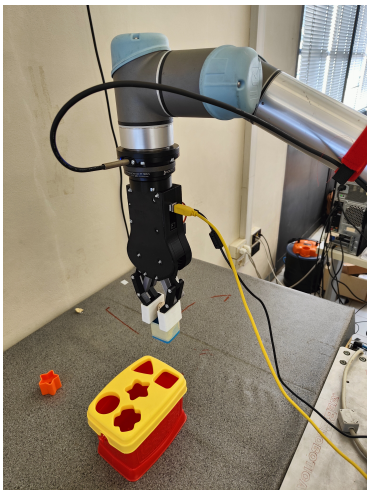


Figura 6.3: Setup di utilizzo del sensore coppia/forza per l'inserimento delle forme all'interno della scatola.

6.2.1 Posizionamento dell'oggetto tramite ricerca dei bordi del contenitore

Un primo approccio prevede la ricerca di una specifica area o contenitore in grado di contenere l'oggetto, assumendo che verrà poi piazzato in posizione centrale. In questo caso, il robot parte da una posizione casuale all'interno del contenitore e comincia a muoversi lungo le quattro direzioni per cercare le superfici perimetrali del contenitore in modo da determinarne le dimensioni e calcolare la posizione del centro della scatola. Come viene schematizzato in Figura 6.4, il braccio parte da una posizione generica A e si muove verso

l'alto fino a rilevare un contatto con il primo bordo nel punto B. Una volta determinata la posizione del bordo superiore, il braccio si muove in direzione opposta, alla ricerca del bordo inferiore, fino al contatto con questo nel punto C. Ora che i bordi superiori e inferiori sono noti, il robot può raggiungere il punto medio D tra i bordi superiore e inferiore. Da qui può riprendere a cercare la posizione dei bordi laterali, esaminando prima il bordo destro nel punto di contatto E, e successivamente il bordo sinistro nel punto di contatto F. A questo punto il robot è in grado di stimare e raggiungere la posizione centrale G della scatola, dove sarà possibile rilasciare l'oggetto. Per rilevare il contatto del braccio collaborativo con i bordi del contenitore, vengono costantemente monitorati i valori di forza misurati dal sensore, e nel momento in cui i valori superano un determinato valore di soglia, il robot arresta il movimento e salva la posizione attuale. Riepilogando, se il modulo della forza $|measured\_force| > threshold$ lungo uno degli assi x o y, allora vi è un contatto con un bordo, e il braccio robotico deve arrestare il movimento e salvare la posizione.

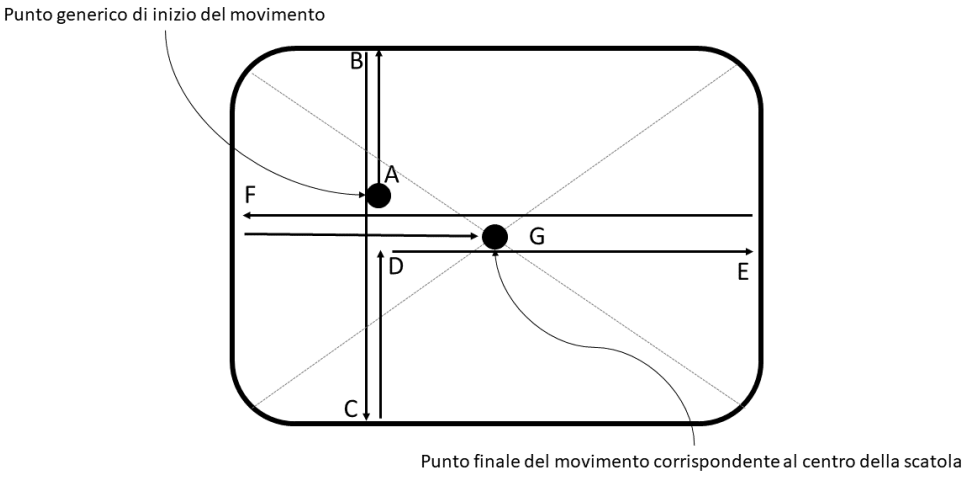


Figura 6.4: Setup di utilizzo del sensore coppia/forza per l'inserimento delle forme all'interno della scatola.

Per poter rendere affidabile il rilevamento del contatto con il bordo del contenitore, sono stati effettuati diversi test utilizzando differenti soglie di riferimento e cercando di ridurre il rumore delle misurazioni della forza mediando i valori su un certo numero di pacchetti ricevuti. Durante i vari test effettuati è stato possibile determinare sperimentalmente il valore di soglia ottimale per la forza di rilevamento del contatto, posto pari a $8N$, mentre è stata utilizzata una coda degli ultimi 15 pacchetti per calcolare la media della forza misurata.

Il problema riscontrato più spesso riguarda il blocco del braccio robotico a causa dell'intervento dei criteri di protezione dell'integrità del robot implementati nel braccio stesso, che percependo uno sforzo imprevisto lungo i vari giunti arresta l'esecuzione del

movimento e richiede l'intervento dell'operatore. In generale, è stato constatato che l'utilizzo di un valore di soglia leggermente più alto e la riduzione del numero di pacchetti su cui mediare la misurazione dei valori, risulta essere la soluzione più efficace per il rilevamento del contatto.

Una variante che aggiunge un minimo livello di complessità a questo tipo di esperienza richiede di utilizzare i bordi *esterni* del contenitore, piuttosto che quelli interni, in modo da poter considerare anche contenitori coperti da una superficie superiore. Nonostante l'idea principale rimanga quella della ricerca dei bordi lungo le quattro direzioni, si aggiunge la necessità di dover muovere il braccio anche verticalmente, in modo da poter raggiungere spigoli opposti senza entrare in collisione con il contenitore durante la sequenza di movimenti. In questa variante è stato necessario fornire al robot un limite inferiore della dimensione della scatola, per evitare che il braccio potesse tentare di cercare un bordo prima che si fosse spostato completamente al di fuori dell'area coperta dal contenitore. Per calcolare l'altezza del contenitore ed evitare che il braccio entri in contatto con la scatola, il sensore coppia/forza viene utilizzato anche per cercare la superficie superiore del contenitore. Il robot viene inizialmente posizionato sopra al contenitore e comincia a scendere verticalmente aspettando il contatto con la superficie superiore della scatola. Una volta che il sensore misura un picco di forza superiore al valore soglia pari a $8N$, il robot si blocca e registra l'altezza in cui si trova come altezza della scatola. A questa altezza viene poi aggiunto un *offset* arbitrario di qualche centimetro per garantire che il braccio non entri in collisione con la scatola. Anche in questo caso, il contatto con i bordi del contenitore ha permesso al braccio meccanico di poter calcolare la posizione centrale del contenitore, utilizzata poi per il posizionamento effettivo dell'oggetto.

6.2.2 Posizionamento dell'oggetto tramite ricerca lungo una superficie

Un secondo approccio per il corretto posizionamento degli oggetti richiede l'utilizzo del sensore in modo più continuativo lungo una superficie orizzontale. Effettuando una ricerca lungo la superficie, è possibile identificare il punto preciso in cui posizionare un oggetto nel caso in cui questo debba essere infilato in una posizione specifica. Questo approccio è risultato essere molto più complesso di quanto inizialmente supposto, a causa delle numerose condizioni che spesso si sono verificate durante i vari tentativi. Innanzitutto è necessario definire come rilevare il successo dell'inserimento, che nell'esperienza sviluppata viene definito quando l'oggetto raggiunge una posizione inferiore di $2cm$ rispetto alla posizione di partenza. In seguito, è necessario stabilire una strategia per l'esplorazione della superficie di riferimento, che può essere una sequenza predeterminata di posizioni o una graduale ricerca lungo il piano. Poiché non è nota con precisione la direzione specifica in cui è necessario effettuare la ricerca, è stato scelto di utilizzare un movimento a spirale archimedea per cercare di esplorare la maggior area possibile, come nell'esempio mostrato in Figura 6.5.

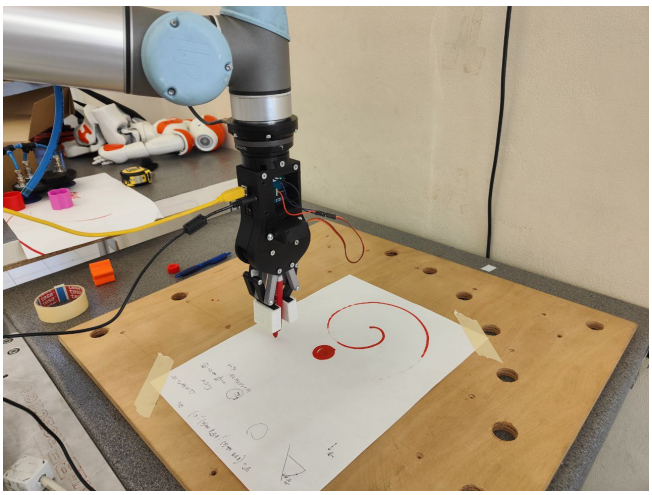


Figura 6.5: Test dell'esplorazione della superficie utilizzando un movimento a spirale.

Stabilite le condizioni di movimento del robot, è necessario effettuare il controllo dei movimenti cercando di mantenere costante il contatto tra il braccio robotico e la superficie di ricerca, cosa che si è rilevata molto difficile in quanto il piano di ricerca non era rigido e perfettamente piatto ma piuttosto flessibile e leggermente curvo. Questo ha richiesto di modificare lievemente la strategia di movimento in modo da poter costantemente abbassare il piano di ricerca quando i valori della forza cominciavano a diminuire. In pratica però, durante le varie esperienze che sono state effettuate non è stato sempre possibile portare a termine con successo l'inserimento, in quanto spesso accadeva che l'oggetto da posizionare si incastrasse o ri-orientasse a causa della forza necessaria a mantenere il contatto, oppure che la difformità della superficie impedisse al robot di proseguire nel movimento.

In conclusione possiamo dire che i due diversi approcci per il posizionamento degli oggetti si possono applicare in casistiche differenti e possono essere utilizzati in modo sia indipendente che non, e premettono di sfruttare il feedback fornito dal sensore per raccogliere informazioni sull'ambiente e utilizzarle per cercare di raggiungere l'obiettivo preposto.

6.3 Applicazione *pick and place* collaborativa

Sfruttando lo studio effettuato nei due *task* precedentemente illustrati, è stato possibile combinare il software sviluppato in ROS per costruire un esempio pratico di uso del sensore in diverse modalità. Il problema progettato consiste nel riuscire a infilare correttamente un solido a forma di stella all'interno di una scatola dotata di fessure con forme differenti. Il braccio collaborativo non conosce inizialmente la posizione della scatola e dell'oggetto, ma vengono forniti in modo approssimato dall'operatore che muove il robot utilizzando il controllo manuale descritto in sezione 6.1. In questa applicazione è stata utilizzata una pinza meccanica o *gripper* in grado di afferrare saldamente l'oggetto e di rilasciarlo nella posizione opportuna. L'esperienza consiste di due fasi principali.

In una prima fase, detta di *addestramento*, l'operatore afferra la pinza agganciata al robot e tramite il controllo basato sulle forze descritto in sezione 6.1 è in grado di posizionare il braccio in posizione verticale sopra all'oggetto. A questo punto, l'operatore conferma al robot la prima posizione effettuando una torsione lungo l'asse z del sensore, cercando di ruotare il *gripper*. Tale movimento di torsione è stato codificato come una gesture collaborativa per il salvataggio delle posizioni del robot; dopo l'effettivo salvataggio della posizione è previsto un feedback verso l'utente in cui il gripper muove leggermente le pinze in modo da fornire indicazione all'operatore del corretto salvataggio della posizione. In seguito l'operatore posiziona il braccio approssimativamente sopra alla scatola dove andrà posizionato l'oggetto e ripete la torsione per confermare la posizione. Infine, l'operatore muove il robot in un area vuota e fornisce una torsione per indicare al robot di iniziare la seconda fase dell'esperienza.

A questo punto, a partire dall'ultima posizione salvata dall'operatore, il braccio meccanico comincia a muoversi autonomamente e come prima cosa, utilizza il sensore per cercare un contatto verticale con il piano di lavoro, per prendere un riferimento iniziale. In seguito, il robot si muove nella prima posizione specificata dall'operatore e cerca di determinare l'altezza dell'oggetto da afferrare toccandone la superficie superiore. Dopo aver calcolato l'altezza dell'oggetto sfruttando la differenza tra il riferimento di altezza del piano di lavoro e il riferimento di altezza dell'oggetto, il robot è in grado di afferrare saldamente l'oggetto e si muove alla seconda posizione specificata dall'operatore, dove è situata la scatola che dovrà contenere l'oggetto. Inizia ora il processo di ricerca delle dimensioni della scatola, che avviene prima toccando la superficie superiore del contenitore e in seguito tutti i bordi esterni presenti lungo il suo perimetro, con la procedura descritta nella sottosezione 6.2.1. A questo punto il braccio collaborativo è in grado di riposizionarsi approssimativamente al centro della scatola, dove può iniziare il processo di esplorazione della superficie in un movimento a spirale fino al corretto allineamento dell'oggetto con la sua corrispondente forma.

In conclusione, l'applicazione di *pick and place* collaborativo riassume i diversi utilizzi del sensore coppia/forza che sono stati studiati nel corso di questo lavoro di tesi, applicandoli ad un caso d'uso reale che prevede l'interazione tra una persona e il braccio collaborativo. L'applicazione sviluppata fornisce una interfaccia di controllo alternativa e più intuitiva rispetto all'uso del *teach pendant* fornito da Universal Robot.

Conclusioni e studi futuri

In conclusione, questa tesi ha esplorato una parte delle possibili applicazioni che possono trarre vantaggio dall'utilizzo di un sensore coppia/forza tramite il sistema ROS. Sono state presentate le diverse tipologie di sensori coppia/forza esistenti, evidenziandone le differenze nei diversi approcci alla misurazione e analizzando i possibili vantaggi o svantaggi relativi alle diverse tecniche di costruzione. Dopo una breve panoramica sull'utilizzo di ROS e la libreria Moveit per il controllo del robot collaborativo UR5, sono state analizzate le diverse tipologie di collegamento del sensore con il computer per la lettura dei dati, ed è stato verificato come la modalità di collegamento attraverso il controllore del robot risulti essere sconveniente sia per il formato dei dati ricevuti che per l'effettiva misurazione poco accurata fornita dal sensore. Sono stati poi presentati gli studi sugli utilizzi del sensore coppia/forza Robotiq tramite specifici *task* in grado di evidenziare come l'utilizzo di questo tipo di sensori sia vantaggioso e meritevole di ulteriore ricerca, soprattutto nel'ambito della robotica collaborativa, dove l'obiettivo non è quello di rimpiazzare completamente il fattore umano, ma quello di migliorare il più possibile le capacità di collaborazione in un ambiente condiviso. Sono state illustrate alcune tecniche per la calibrazione del sensore per consentirne l'utilizzo in modo preciso e affidabile, e sono stati testati i limiti delle capacità di funzionamento del sensore tramite esperimenti mirati a misurare forze e torsioni specifiche. E' stato presentato un metodo alternativo più intuitivo per il controllo collaborativo del robot da parte di un operatore, ed è stato applicato per simulare un *task* di drappeggio collaborativo uomo-robot. Sono stati presentati alcuni metodi per l'utilizzo del sensore in ambito industriale per l'inserimento di oggetti in specifici alloggiamenti utilizzando sia l'esplorazione nello spazio per la ricerca dei limiti dei contenitori, sia l'esplorazione di una superficie per ricercare il corretto allineamento dell'oggetto con una fessura specifica. Infine, è stato presentato un esempio completo di applicazione del sensore che riassume gli studi effettuati, dove il sensore viene prima utilizzato per fornire al robot alcune posizioni approssimative, e in seguito come strumento di feedback per il corretto posizionamento degli oggetti.

Il lavoro presentato fornisce un buono spunto per lo sviluppo di ulteriori applicazioni di robotica collaborativa, ad esempio sviluppando un algoritmo in grado di compensare costantemente il vettore forza-peso in base all'orientamento dell'*end effector* nello spazio, oppure analizzando e sviluppando migliori tecniche per l'esplorazione delle superfici anche non piane al fine di migliorare il posizionamento degli oggetti, magari includendo la possi-

bilità di effettuare rotazioni, ma può anche essere interessante studiare come sia possibile utilizzare il sensore per fornire una nuova interfaccia di comunicazione uomo-macchina. Sarebbe possibile implementare nuove tipologie di *gesture* che possono essere sfruttate per inviare comandi al robot collaborativo sfruttando il sensore coppia/forza. Ad esempio l'operatore potrebbe utilizzare un tocco laterale lungo il *tool* del robot, oppure potrebbe tirare il *gripper* verso il basso, mentre per confermare la ricezione del comando, il robot potrebbe effettuare un breve movimento verso l'alto e verso il basso oppure effettuare una piccola rotazione del *tool*.

Bibliografia

[1] M. Y. Cao, S. Laws e F. R. y. Baena, «Six-Axis Force/Torque Sensors for Robotics Applications: A Review», *IEEE Sensors Journal*, vol. 21, n. 24, pp. 27 238–27 251, 2021. DOI: 10.1109/JSEN.2021.3123638 (cit. a p. 10).

[2] J. O. Templeman, B. B. Sheil e T. Sun, «Multi-axis force sensors: A state-of-the-art review», *Sensors and Actuators A: Physical*, vol. 304, p. 111 772, 2020, ISSN: 0924-4247. DOI: https://doi.org/10.1016/j.sna.2019.111772. indirizzo: https://www.sciencedirect.com/science/article/pii/S0924424719308842 (cit. a p. 10).

[3] U. Kim, C.-Y. Maeng, G. Bak e Y.-J. Kim, «High-Stiffness Torque Sensor With a Strain Amplification Mechanism for Cooperative Industrial Manipulators», *IEEE Transactions on Industrial Electronics*, vol. 69, n. 3, pp. 3131–3141, 2022. DOI: 10.1109/TIE.2021.3068656 (cit. a p. 11).

[4] D. Kim, C. Lee, B. Kim et al., «Six-axis Capacitive Force/Torque Sensor Based on Dielectric Elastomer», vol. 8687, apr. 2013, 86872J. DOI: 10.1117/12.2009970 (cit. a p. 11).

[5] D.-H. Lee, U. Kim, H. Jung e H. R. Choi, «A Capacitive-Type Novel Six-Axis Force/Torque Sensor for Robotic Applications», *IEEE Sensors Journal*, vol. 16, n. 8, pp. 2290–2299, 2016. DOI: 10.1109/JSEN.2015.2504267 (cit. a p. 11).

[6] J. Liu, M. Li, L. Qin e J. Liu, «Active Design Method for the Static Characteristics of a Piezoelectric Six-Axis Force/Torque Sensor», *Sensors*, vol. 14, n. 1, pp. 659–671, 2014, ISSN: 1424-8220. DOI: 10.3390/s140100659. indirizzo: https://www.mdpi.com/1424-8220/14/1/659 (cit. a p. 11).

[7] Y.-J. Li, B.-Y. Sun, J. Zhang, M. Qian e Z.-Y. Jia, «A novel parallel piezoelectric six-axis heavy force/torque sensor», *Measurement*, vol. 42, n. 5, pp. 730–736, 2009, ISSN: 0263-2241. DOI: https://doi.org/10.1016/j.measurement.2008.12.005. indirizzo: https://www.sciencedirect.com/science/article/pii/S0263224108002194 (cit. a p. 11).

[8] G. Palli, L. Moriello, U. Scarcia e C. Melchiorri, «Development of an optoelectronic 6-axis force/torque sensor for robotic applications», *Sensors and Actuators A: Physical*, vol. 220, pp. 333–346, 2014, ISSN: 0924-4247. DOI: https://doi.org/10.1016/j.sna.2014.09.023. indirizzo: https://www.sciencedirect.com/science/article/pii/S092442471400418X (cit. a p. 11).

[9] Y. Noh, J. Bimbo, S. Sareh et al., «Multi-Axis Force/Torque Sensor Based on Simply-Supported Beam and Optoelectronics», *Sensors*, vol. 16, n. 11, 2016, ISSN: 1424-8220. indirizzo: https://www.mdpi.com/1424-8220/16/11/1936 (cit. a p. 11).

[10] A. Biason, G. Boschetti, A. Gasparetto, M. Giovagnoni e V. Zanotto, «A Force-Torque Sensor for the Applications in Medical Robotics», in *AMST'05 Advanced Manufacturing Systems and Technology*, E. Kuljanic, cur., Vienna: Springer Vienna, 2005, pp. 543–550, ISBN: 978-3-211-38053-6 (cit. a p. 12).

[11] G. G. Muscolo e P. Fiorini, «Force-Torque Sensors for Minimally Invasive Surgery Robotic Tools: an overview», *IEEE Transactions on Medical Robotics and Bionics*, pp. 1–1, 2023. DOI: 10.1109/TMRB.2023.3261102 (cit. a p. 12).

[12] H. Su, I. I. Iordachita, J. Tokuda et al., «Fiber-Optic Force Sensors for MRI-Guided Interventions and Rehabilitation: A Review», *IEEE Sensors Journal*, vol. 17, n. 7, pp. 1952–1963, 2017. DOI: 10.1109/JSEN.2017.2654489 (cit. a p. 12).

[13] J. A. Díez, A. Blanco, J. M. Catalán, F. J. Badesa, L. D. Lledó e N. García-Aracil, «Hand exoskeleton for rehabilitation therapies with integrated optical force sensor», *Advances in Mechanical Engineering*, vol. 10, n. 2, p. 1 687 814 017 753 881, 2018. DOI: 10.1177/1687814017753881. eprint: https://doi.org/10.1177/1687814017753881. indirizzo: https://doi.org/10.1177/1687814017753881 (cit. a p. 12).

[14] K. Nam, S. Oh, H. Fujimoto e Y. Hori, «Estimation of Sideslip and Roll Angles of Electric Vehicles Using Lateral Tire Force Sensors Through RLS and Kalman Filter Approaches», *IEEE Transactions on Industrial Electronics*, vol. 60, n. 3, pp. 988–1000, 2013. DOI: 10.1109/TIE.2012.2188874 (cit. a p. 12).

[15] H. Yan, W. Zhang e D. Wang, «Wheel Force Sensor-Based Techniques for Wear Detection and Analysis of a Special Road», *Sensors*, vol. 18, n. 8, 2018, ISSN: 1424-8220. DOI: 10.3390/s18082493. indirizzo: https://www.mdpi.com/1424-8220/18/8/2493 (cit. a p. 12).

[16] M. Sun e T. Luan, «A Novel Control System of Ship Fin Stabilizer Using Force Sensor to Measure Dynamic Lift», *IEEE Access*, vol. 6, pp. 60 513–60 531, 2018. DOI: 10.1109/ACCESS.2018.2875243 (cit. a p. 13).

[17] W. Zhang, V. T. Truong, K. B. Lua et al., «Design and characterization of a silicon piezoresistive three-axial force sensor for micro-flapping wing MAV applications», in *International Conference on Experimental Mechanics 2014*, C. Quan, K. Qian, A. Asundi e F. S. Chau, cur., International Society for Optics e Photonics, vol. 9302, SPIE, 2015, 93023E. DOI: 10.1117/12.2081146. indirizzo: https://doi.org/10.1117/12.2081146 (cit. a p. 13).

[18] X. Deng, L. Schenato e S. Sastry, «Flapping flight for biomimetic robotic insects: part II-flight control design», *IEEE Transactions on Robotics*, vol. 22, n. 4, pp. 789–803, 2006. DOI: 10.1109/TRO.2006.875483 (cit. a p. 13).

[19] M. Quigley, K. Conley, B. P. Gerkey et al., «ROS: an open-source Robot Operating System», in *ICRA Workshop on Open Source Software*, 2009. indirizzo: https://www.bibsonomy.org/bibtex/2281f400bf541a0022e41ace75d9156ea/markusjordan88 (cit. a p. 15).

[20] D. Coleman, I. Sucan, S. Chitta e N. Correll, *Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study*, 2014. arXiv: 1404.3785 [cs.RO] (cit. a p. 31).

[21] F. B. Carlson, *On the Calibration of Force/Torque Sensors in Robotics*, 2019. arXiv: 1904.06158 [cs.RO] (cit. a p. 35).

[22] S. G. Vougioukas, «Bias Estimation and Gravity Compensation For Force-Torque Sensors», 2001 (cit. a p. 35).

[23] L. Fu e A. Song, «Dynamic Characteristics Analysis of the Six-Axis Force/Torque Sensor», *Journal of Sensors*, vol. 2018, pp. 1–11, dic. 2018. DOI: 10.1155/2018/6216979 (cit. a p. 38).

[24] M. M. Rahman e M. Halder, «Design, Construction and Performance Test of a Rotational Digital Viscometer.», dic. 2014 (cit. a p. 39).

[25] S. Ghidoni, M. Terreran, D. Evangelista et al., «A Smart Workcell for Human-Robot Cooperative Assembly of Carbon Fiber Parts», 2021. DOI: 10.5281/ZENODO.6367920 (cit. a p. 44).

Sitografia

[26] IAS-Lab. «IAS-Lab». (2023), indirizzo: http://robotics.dei.unipd.it/ (cit. a p. 7).

[27] bcastets. «What is the working principle of Robotiq ft300s force torque sensor?» (2022), indirizzo: https://dof.robotiq.com/discussion/2627/what-is-the-working-principle-of-robotiq-ft300s-force-torque-sensor (cit. alle pp. 12, 27).

[28] ROS.org, Open Robotics. «What is ROS?» (Ago. 2018), indirizzo: http://wiki.ros.org/ROS/Introduction (cit. a p. 15).

[29] B. Gerkey e ROS.org, Open Robotics. «Why ROS 2?» (Giu. 2018), indirizzo: http://design.ros2.org/articles/why_ros2.html (cit. a p. 15).

[30] Universal Robots. «External Control URCap». (2022), indirizzo: https://github.com/UniversalRobots/Universal_Robots_ExternalControl_URCap (cit. a p. 25).

[31] M.-E. Eitan, P. Vijay e A. Mikael. «Actionlib». (2022), indirizzo: http://wiki.ros.org/actionlib (cit. a p. 25).

[32] Universal Robots. «ROS-Controllers available for the ur_robot_driver». (2021), indirizzo: https://github.com/UniversalRobots/Universal_Robots_ROS_Driver/blob/master/ur_robot_driver/doc/controllers.md (cit. a p. 26).

[33] Robotiq, *Robotiq Copilot Instruction Manual*, Sezione 6.1.3 *Data stream*, mar. 2021. indirizzo: https://assets.robotiq.com/website-assets/support_documents/document/Copilot_Instruction_20Manual_20210301.pdf (cit. a p. 28).

[34] TAMS-Group. «Robotiq package that maintained by community». (2023), indirizzo: https://github.com/TAMS-Group/robotiq (cit. a p. 29).

[35] Dave Hershberger, David Gossow, Josh Faust, William Woodall. «RViz - ROS Wiki». (2023), indirizzo: http://wiki.ros.org/rviz (cit. a p. 32).

[36] S. Ioan e J. Kay. «Urdf». (2023), indirizzo: http://wiki.ros.org/urdf (cit. a p. 32).

[37] MoveIt. «MoveIt Setup Assistant». (2023), indirizzo: https://github.com/ros-planning/moveit_tutorials/blob/master/doc/setup_assistant/setup_assistant_tutorial.rst (cit. a p. 34).

[38] S. Ioan. «Srdf». (2022), indirizzo: http://wiki.ros.org/srdf (cit. a p. 34).

[39] A. Stocco e S. Peraro. «andreastocco01/ur5_ft_tasks». (2023), indirizzo: https://github.com/andreastocco01/ur5_ft_tasks (cit. a p. 43).