Università degli Studi di Padova

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Triennale in
Ingegneria Informatica

Analisi dei sensori coppia-forza per lo sviluppo di applicazioni industriali in ROS

Relatore:
Prof. Stefano Ghidoni
Correlatore:
Matteo Terreran, PhD

Laureando:
Andrea Stocco
2009353

Anno Accademico 2022/2023

Abstract

I sensori coppia-forza sono componenti fondamentali nei sistemi robotici in quanto forniscono dati sulle forze e i momenti esterni applicati al robot. Questi dati possono essere utilizzati per applicazioni a supporto della collaborazione uomo-robot e per l'automatizzazione di attivitá che richiedono elevata precisione. ROS (Robot Operating System) é un framework che fornisce una vasta gamma di librerie e strumenti software per lo sviluppo di applicazioni robotiche. In questa tesi verranno mostrate delle possibili applicazioni ROS in ambito industriale per i sensori coppia-forza, dopo averne validato l'accuratezza attraverso alcuni esperimenti.

Indice

Introduzione **1**

1 ROS **3**

1.1 Workspace 3

1.2 Nodo 3

1.3 Topic 4

1.4 Service 5

1.5 Topic e Service a confronto 6

2 Ambiente di lavoro **7**

2.1 UR5 7

2.2 FT300-S 9

2.2.1 Collegamento via USB tra sensore e control box e via ethernet tra control box e computer 10

2.2.2 Collegamento diretto via USB tra sensore e computer . . . 10

Bibliografia **11**

Introduzione

I robot manipolatori hanno rivoluzionato l'automazione industriale, permettendo lo svolgimento di operazioni complesse in modo rapido, preciso e sicuro. Un ruolo chiave nel controllo di questi robot viene assunto dai sensori coppia-forza, che permettono di misurare e regolare la forza esercitata dal robot durante lo svolgimento delle proprie attivitá. La loro versatilitá li rende strumenti preziosi in molti campi tra cui: la **robotica industriale** e la **medicina**. Essi infatti consentono ai robot di controllare la forza esercitata durante le operazioni di assemblaggio, levigatura, saldatura e manipolazione degli oggetti. Vengono inoltre utilizzati nella riabilitazione, per valutare la forza muscolare e i progressi del paziente. Uno dei modelli di robot manipolatori piú utilizzato é l'**UR5** di **Universal Robot**, per via della sua flessibilitá ed efficienza. Tuttavia, a differenza di altri, l'UR5 non é dotato di sensori coppia-forza. É stato, dunque, necessario installarne uno manualmente. A tale scopo si é scelto di utilizzare il sensore **FT 300-S** di **Robotiq**. In questa tesi verrá presentata l'implementazione di un sistema di controllo della forza per l'UR5 utilizzando i dati forniti dall'FT 300-S e il framework di sviluppo **ROS (Robot Operating System)**. ROS é ampiamente utilizzato dalla comunitá informatica perché fornisce strumenti e librerie per il controllo e la comunicazione tra le componenti di un sistema robotico. Verranno effettuati prima, alcuni test per valutare le prestazioni del sensore in termini di reattivitá e precisione, e successivamente presentate delle possibili applicazioni volte a dimostrare l'efficacia di tali sensori per lo svolgimento di attivitá industriali.

Capitolo 1

ROS

ROS (Robot Operating System) é un framework open-source disponibile in Python e C++ per lo sviluppo di applicazioni robotiche. La versione raccomandata e utilizzata é **ROS Noetic** per **Ubuntu Focal 20.04**. Il codice presentato nel corso di questa tesi é stato scritto in C++. Questo capitolo ci fornirá una panoramica esaustiva del Robot Operating System e dell'ambiente di esecuzione associato [2].

1.1 Workspace

Per poter eseguire codice ROS serve un ambiente che permetta l'organizzazione e l'utilizzo di tutti i pacchetti necessari. Al riguardo, **catkin** é il sistema di compilazione ufficiale di ROS che consente la creazione di un workspace per organizzare e gestire le applicazioni. I termini 'pacchetto' e 'applicazione' sono interscambiabili e possono essere utilizzati in modo equivalente (una definizione piú rigorosa verrá data successivamente nel corso di questo capitolo). Una volta creato il workspace catkin [3], il codice sorgente contenuto all'interno dei pacchetti potrá essere compilato ed eseguito.

1.2 Nodo

Un **pacchetto** ROS contiene al suo interno codici sorgenti, librerie e dati di configurazione. Per creare un pacchetto é sufficiente eseguire il comando `catkin_create_pkg <nome_pacchetto>` all'interno della cartella `src/` situata nel workspace catkin. Un **nodo** é un eseguibile che sfrutta ROS per comunicare con altri nodi. Quando viene lanciato il comando `catkin_make`, ogni file sorgente

in ogni pacchetto viene compilato dando origine ad un nodo. Impropriamente, si potrebbe quindi dire che un pacchetto é un insieme di nodi riguardanti la stessa applicazione. Per poter eseguire un nodo é sufficiente eseguire il comando `rosrun <nome_pacchetto> <nome_nodo>`. Tuttavia si incorrerá in un errore se prima non viene fatto partire un ROS Master. Lo scopo principale di un **ROS Master** é quello di consentire ai singoli nodi di localizzarsi a vicenda. Una volta fatto questo, essi potranno comunicare tra loro attraverso topic o service. Per far partire un ROS Master sará sufficiente eseguire il comando `roscore` in un altro terminale.

1.3 Topic

I **topic** sono dei canali di comunicazione unidirezionali che consentono lo scambio di informazioni tra nodi sottoforma di messaggi. Un nodo che pubblica messaggi su un topic viene chiamato **publisher**, mentre un nodo che legge i messaggi da un topic viene chiamato **subscriber**.

Figura 1.1: Schema di comunicazione

Per pubblicare un messaggio su un topic bisogna utilizzare l'apposita funzione `publish()` passandole come parametro il messaggio che vogliamo pubblicare. A discapito del nome, questa funzione non pubblica effettivamente il messaggio, ma lo mette in una coda d'attesa (la cui dimensione viene specificata quando viene istanziato il publisher). Un thread separato si occupa di inviare effettivamente il messaggio al topic per renderlo visibile a tutti i nodi subscriber connessi. Se il numero di messaggi in coda supera la sua dimensione, i messaggi piú vecchi verranno cancellati per fare spazio a quelli piú recenti. Quando arriva un nuovo messaggio al subscriber, esso viene salvato in una coda d'attesa (stesso funzionamento di quella del publisher) fino a quando ROS non dá la possibilitá al nodo di eseguire la funzione di callback. Tale funzione é definita dall'utente e si occupa

di processare il messaggio ricevuto. ROS eseguirá una callback solo quando gli daremo il permesso di farlo. Ci sono due modi per fare ció:

- `ros::spinOnce()` chiede a ROS di eseguire tutte le callback in sospeso e poi ci restituisce il controllo

- `ros::spin()` chiede a ROS di attendere e di eseguire tutte le callback in sospeso fino a quando il nodo non viene spento. É equivalente a:

```
while (ros::ok()) {
    ros::spinOnce();
}
```

  `ros::ok()` ritorna 0 quando:

  - il nodo viene spento attraverso un SIGINT (Ctrl-C) oppure dalla chiamata di `ros::shutdown()` in un altro punto del codice
  - un altro nodo con lo stesso nome viene eseguito

In altre parole `ros::spin()` vincola il nodo a rimanere sempre e solo in attesa di leggere nuovi messaggi. Se il nodo non deve solamente eseguire le callback, allora un loop con `ros::spinOnce()` é la scelta corretta.

1.4 Service

I **service** sono un altro modo in cui i nodi possono comunicare tra loro. A differenza dei topic, che consentono la comunicazione asincrona, i service instaurano una comunicazione di tipo 'client-server'. Il nodo **client** invia una richiesta al nodo service e attende la sua risposta prima di andare avanti con l'esecuzione. Per implementare un service in ROS é necessario per prima cosa definire la struttura dei messaggi di richiesta e risposta [6]. Successivamente il client potrá creare una richiesta nel formato specificato e inviarla al nodo service come parametro della funzione `call()`. Il nodo service elaborerá quindi la richiesta fornendo una risposta al client.

1.5 Topic e Service a confronto

I topic e i service sono due modalitá di comunicazione molto diverse a livello concettuale. Con un topic si instaura una comunicazione asincrona in cui tutti i subscriber connessi attendono la pubblicazione di un messaggio da parte del publisher, mentre con un service é il 'server' che rimane in attesa di richieste da parte dei client connessi. Questo rende i topic piú adatti per la trasmissione di flussi di dati continui (come quelli provenienti dai sensori) e i service piú indicati nel caso di servizi puntuali, come la richiesta di un calcolo o di un'altra specifica azione ad un altro nodo [5]. Nel corso di questa tesi verranno utilizzate entrambe le modalitá di comunicazione, con prevalenza di quella topic.

Capitolo 2

Ambiente di lavoro

Questo capitolo offrirá una panoramica sul setup dell'ambiente di lavoro. Si parlerá diffusamente delle caratteristiche e delle specifiche tecniche del robot e del sensore. Verranno, inoltre, mostrati i passaggi fondamentali che hanno consentito il controllo del sistema attraverso ROS.

2.1 UR5

La versione attuale dell'UR5 é quella appartenente alla **e-series**, rilasciata nel 2018. Tuttavia, in questa tesi, viene utilizzata la versione precedente della famiglia **CB3**, commercializzata a partire dal 2008.

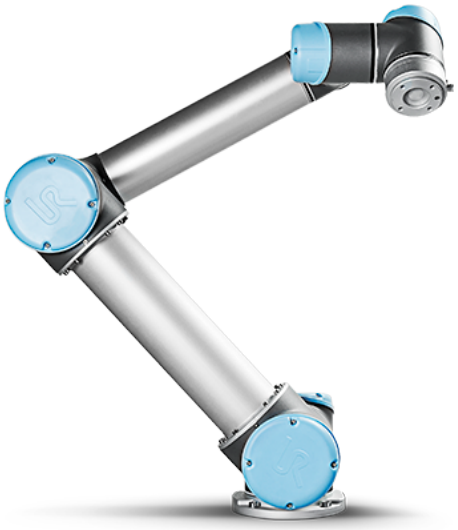


Figura 2.1: UR5/CB3

All'estremitá del robot é possibile installare un **end effector**, ossia un dispositivo concepito per la manipolazione degli oggetti che fornisce l'unica possibile interazione con l'ambiente esterno. Il carico massimo sopportabile dall'UR5 dipende dall'offset del centro di gravitá, ossia la distanza tra l'estremitá del braccio robotico (punto di applicazione dell'end effector) e il centro di gravitá dell'UR5.

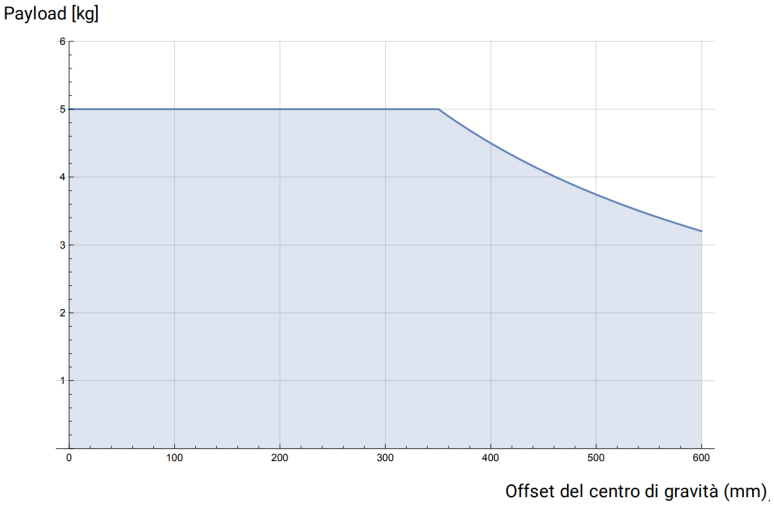
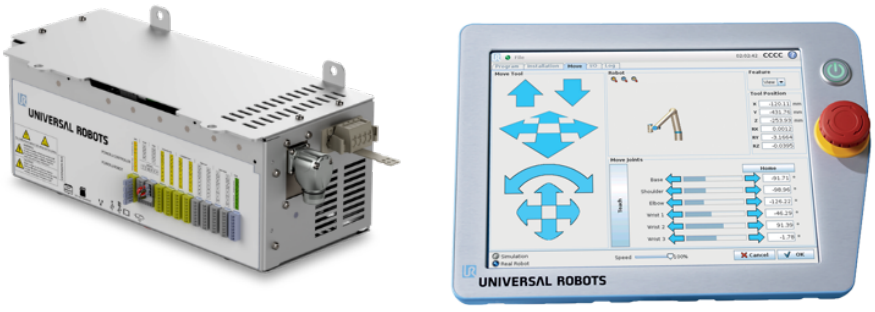


Figura 2.2: Andamento del carico massimo sopportabile rispetto all'offset del centro di gravitá

Come si puó notare in Figura 2.2, l'UR5 é in grado di sopportare un carico massimo di 5kg finché l'estensione del braccio non supera i 350mm. Da qui in poi al crescere dell'offset seguirá una diminuzione del carico massimo applicabile. L'UR5 é collegato alla cosiddetta **control box** che é un dispositivo collegato alla rete elettrica in grado di fornire energia sia al robot che alle periferiche collegate ad esso. Sulla control box é presente poi un'uscita ethernet per la connessione e il controllo remoto, a cui é stato collegato il computer da cui vengono lanciati tutti gli applicativi sviluppati.



(a) Control box **(b)** Teach pendant

Figura 2.3

In Figura 2.3 vengono mostrati la control box e il **Teach Pendant**, un dispositivo touchscreen collegato alla control box che consente il controllo del robot. Sul Teach Pendant é eseguita un'interfaccia chiamata **Polyscope** che permette la programmazione dei movimenti del robot e il cambiamento di alcune sue impostazioni. Per poter controllare l'UR5 tramite ROS é, tuttavia, necessario installare sul Teach Pendant il plugin **externalcontrol.urcap**, creare un nuovo programma su Polyscope che preveda l'utilizzo del plugin e impostare l'indirizzo ip del computer remoto da cui verranno lanciati i nodi ROS.

2.2 FT300-S

Il sensore FT300-S di Robotiq é stato installato all'estremitá dell'UR5 e collegato alla control box per ricevere l'alimentazione necessaria.

Figura 2.4: FT300-S

É importante notare come la sua presenza non precluda la possibilitá di installazione di un end effector, che puó essere facilmente posizionato 'al di sopra' del sensore. L'FT300-S é in grado di rilevare forze e torsioni nel range di $\pm 300N$ e $\pm 30Nm$ rispettivamente. Le misurazioni del sensore hanno un rumore di fondo intrinseco, é quindi necessario scartare tutti i dati al di sotto delle soglie consigliate nel manuale [8] in quanto non attendibili. Per interfacciarsi con il sensore sono disponibili due modalitá di comunicazione: **ModbusRTU** e **data stream**. La prima viene utilizzata per inviare comandi al sensore (es. azzeramento) e per richiedere informazioni su di esso, la seconda per ottenere un flusso continuo di dati relativi alle misurazioni effettuate. Per collegare il sensore al PC sono state provate due alternative:

- collegamento via USB tra sensore e control box e via ethernet tra control box e computer

- collegamento diretto via USB tra sensore e computer

2.2.1 Collegamento via USB tra sensore e control box e via ethernet tra control box e computer

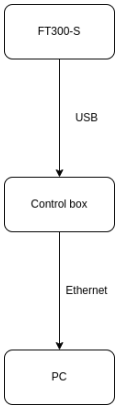


Figura 2.5: Schema collegamento

bla bla bla

2.2.2 Collegamento diretto via USB tra sensore e computer

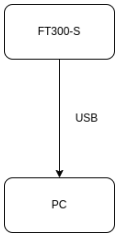


Figura 2.6: Schema collegamento

bla bla bla

Bibliografia

[1] *Quigley, Morgan, et al. "ROS: an open-source Robot Operating System." ICRA workshop on open source software. Vol. 3. No. 3.3.2009.*

[2] *ROS Tutorials*, http://wiki.ros.org/ROS/Tutorials.

[3] *Catkin Workspace*, http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment.

[4] *YouTube Tutorial*, https://www.youtube.com/playlist?list=PLLSegLrePWgIbIrA4iehUQ-impvIXdd9Q.

[5] *Esempio topic e service*, https://github.com/andreastocco01/ros/tree/main.

[6] *Creating ROS msg and srv* http://wiki.ros.org/ROS/Tutorials/CreatingMsgAndSrv.

[7] *Download material for UR5* https://www.universal-robots.com/cb3/.

[8] *Download FT300-S manual* https://robotiq.com/support/ft-300-force-torque-sensor.