UNIVERSITÀ DI BRESCIA
FACOLTÀ DI INGEGNERIA
Dipartimento di Ingegneria dell'Informazione

Laboratorio di Robotica Avanzata
Advanced Robotics Laboratory

Corso di Robotica
(Prof. Riccardo Cassinis)

Leading by nose

Elaborato di esame di:

Marco Maddiona, Riccardo Orizio, Mattia Rizzini, Maurizio Zucchelli

Consegnato il: **25 Giugno 2014**

Sommario

Il lavoro verte sulla creazione di un programma per il robot industriale Kawasaki che, corredato da un opportuno sensore di forza che fornisca informazioni riguardanti lo spostamento nelle tre direzioni lineari e nelle tre direzioni rotazionali, usi tali informazioni per muovere il robot in modo che risponda agli stimoli senza opporre resistenza (fino a certi limiti fisici di costruzione e legati alla destinazione d'uso), realizzando così la modalità di movimento leading-by-nose.

1. Introduzione

Il lavoro è stato diviso in 3 parti, come segue:

- Montaggio del sensore di forza sul robot industriale e suo collegamento fisico con il computer che ne effettua il controllo;
- Realizzazione di un protocollo di comunicazione tra il controllore del sensore ed il computer che comanda il robot per l'invio dei dati ricavati dal sensore di forza e la loro elaborazione;
- Utilizzo dei dati ricevuti per far muovere il robot in modo "morbido" e il più possibile naturale (senza scatti o movimenti non richiesti);

Il nostro progetto, qui presentato, prevede l'implementazione della terza parte del lavoro. La realizzazione include un certo numero di problematiche non semplici da affrontare, in particolare relative alla modalità di movimento naturale del robot.

2. Il problema affrontato

Come già detto, il nostro lavoro non riguarda né il montaggio né l'interfacciamento con il sensore di forza, ma soltanto l'utilizzo dei dati da esso prodotti in qualche forma al fine di produrre il movimento.

Tale movimento deve essere realizzato tenendo presente la destinazione d'uso finale del progetto, ovvero l'utilizzo del robot in campo medico. In particolare, il robot dovrà trovarsi ad operare in spazi molto ristretti, per cui la velocità di movimento deve essere artificialmente limitata a valori molto inferiori rispetto a quelli di cui il robot sarebbe capace.

Il sensore di forza è in grado di fornire sei valori, dei quali 3 traslazionali (sugli assi x, y, z) e 3 rotazionali (intorno ai tre assi), come coordinate nel sistema di riferimento dello strumento, perché li verrà montato il sensore. Dopo un'attenta valutazione, abbiamo deciso di operare sulla base di 6 valori di velocità, che ci verranno forniti direttamente tramite il protocollo di rete progettato da un altro gruppo di lavoro. Richiedere direttamente la forza ci porterebbe a dover effettuare troppi calcoli sul controllore del Kawasaki e questo avrebbe come conseguenza il rallentamento del suo movimento e la probabile incapacità di "inseguire" in modo efficace i dati forniti dal sensore. Se invece richiedessimo le posizioni, il server dovrebbe stimare tali valori senza avere alcuna informazione sulla posizione attuale del robot, commettendo così errori cumulativi.

3. La soluzione adottata

Il nostro lavoro è stato sviluppato utilizzando il linguaggio *AS*, linguaggio di programmazione proprietario ed in uso al solo robot Kawasaki. Il programma viene eseguito direttamente sul computer di controllo del robot.

3.1. Realizzazione del programma e modalità di verifica

Dal momento che le due parti di lavoro relative alla messa in funzione del sensore sono state eseguite in parallelo con la nostra, non ci è stato possibile eseguire dei test con dati forniti direttamente dallo stesso sin dall'inizio. Per ovviare al problema abbiamo realizzato un programma di tipo *PC* (*Process Control*, non troppo differente da un *thread*) da eseguire sul robot e che generi dei punti sulla base dei quali calcolare le componenti di velocità.

Abbiamo supposto, nella realizzazione, che gli assi del sensore di forza siano solidali con quelli dello strumento (pinza) del robot Kawasaki.

La velocità viene calcolata sulla base di un tempo di percorrenza arbitrario, secondo la formula:

$$velocità\ generata = \frac{punto\ generato - punto\ attuale}{tempo\ arbitrario}$$

Per semplificare i calcoli, il tempo arbitrario è fissato ad un secondo, così che la velocità possa essere generata mediante una semplice differenza

$$velocità\ generata = punto\ generato - punto\ attuale$$

Tale velocità viene quindi scomposta nelle 6 componenti (3 componenti traslazionali e 3 rotazionali) e salvata in sei variabili globali che vengono lette dal programma che effettua il movimento del robot. A queste variabili, nel programma completo, dovranno poi essere assegnati i dati ricevuti dal sensore di forza.

Il programma che genera queste velocità si chiama *morz_pc_gen*[1], e produce dati secondo due leggi distinte. La legge "traslazionale" porta il robot a muoversi secondo la figura geometrica presentata in figura 1, con l'aggiunta di un movimento verticale sinusoidale.

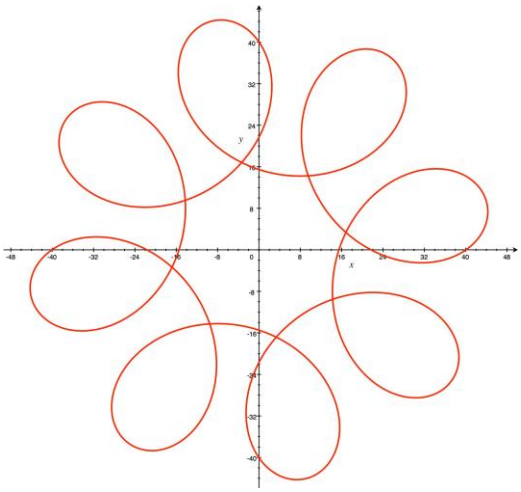

Fig. 1 - Figura geometrica usata per i test

Per verificare la correttezza del movimento abbiamo dotato il robot di un pennarello (tramite un supporto da noi costruito rappresentato in figura 2), facendo così in modo che tracciasse su di un foglio la propria traiettoria.

La traiettoria percorsa sopprimendo il movimento sull'asse z è presentata in figura 3 (una scansione del foglio su cui il robot ha disegnato la forma geometrica).

La versione con movimento sull'asse z corrisponde ad un insieme di punti giacenti su tale traiettoria, ma abbiamo ritenuto non significativa la sua inclusione nella relazione.

---

[1] Questo programma rappresenta in realtà una seconda versione. La prima versione era divisa in due programmi, *morz_pc_m_gen* e *morz_pc_r_gen*, che generavano rispettivamente le velocità traslazionali e quelle rotazionali.

L'unione dei due programmi ha consentito di sincronizzare la generazione dei dati.

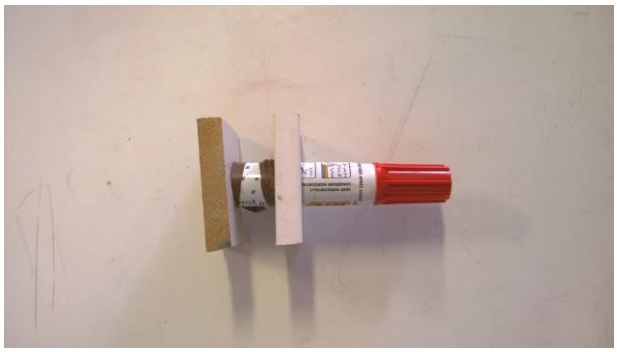

Fig. 2 - Pennarello e relativo supporto

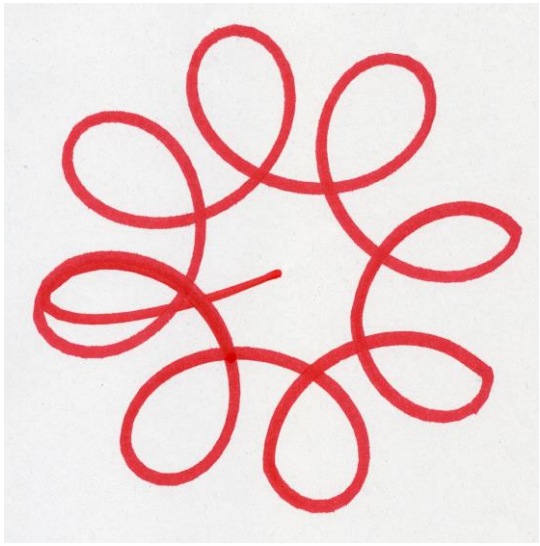

Fig. 3 - Primi disegni di Kiwi

La linea che parte dal centro del disegno è dovuta al fatto che il robot, nel suo stato iniziale, non si trova sulla figura e deve quindi "inseguire" il primo punto generato, convergendo poi alla figura attesa.

Le velocità sono generate sulla base del sistema di riferimento della base del robot e vengono poi trasformate per coincidere con il sistema di riferimento dello strumento (pinza). La trasformazione prevede l'utilizzo degli angoli di Eulero $Z_1Y_2Z_3$ ed è effettuata mediante la matrice di figura 4[2].

$$Z_1Y_2Z_3 = \begin{bmatrix} c_1c_2c_3 - s_1s_3 & -c_3s_1 - c_1c_2s_3 & c_1s_2 \\ c_1s_3 + c_2c_3s_1 & c_1c_3 - c_2s_1s_3 & s_1s_2 \\ -c_3s_2 & s_2s_3 & c_2 \end{bmatrix}$$

Fig.4 – Trasformazione di Eulero

Il robot può muoversi soltanto sulla base di distanze percorse, ovvero indicandogli dei punti di destinazione. Di conseguenza, il programma che comanda il movimento del robot deve calcolare tali punti di destinazione sulla base delle velocità fornite.

La velocità di movimento lineare viene calcolata sulla base delle tre componenti di velocità ricevute, come:

$$v = \sqrt{\left|\overrightarrow{v_x} + \overrightarrow{v_y} + \overrightarrow{v_z}\right|_2}$$

Per attenuare ulteriormente movimenti repentini e simulare anche una resistenza da parte del robot stesso al movimento, non utilizziamo il modulo della velocità, ma la sua radice quadrata. La funzione radice dovrebbe essere sufficiente a limitare la velocità senza dover applicare un valore massimo invalicabile, l'unico intervallo in cui questa velocità risulterebbe superiore a quella reale sarebbe $v^2 \in [0,1]$, ma essendo velocità molto basse e ricevendo valori quantizzati, riteniamo il caso inverificabile e comunque non problematico. Abbiamo provato anche a limitare la velocità riducendola proporzionalmente con delle costanti, ma si è rivelato poco pratico e meno efficiente rispetto alla radice quadrata che, seppur più vincolante rispetto ad un parametro numerico, molto più corretto per il funzionamento da noi previsto.

---

[2] I pedici si riferiscono agli angoli di Eulero utilizzati come parametri delle funzioni c o s.

$s = \sin(x)\,;\ c = \cos(x)\,;$

$1 = \alpha; 2 = \beta; 3 = \gamma;$

Ogni punto di destinazione è calcolato comandando al robot di muoversi per una distanza nella direzione indicata dalle velocità traslazionali così calcolata

$$d = \frac{v}{morz\_v\_scale}$$

e di un angolo in gradi sessagesimali pari a

$$\phi_i = \frac{\omega_i}{morz\_r\_scale}$$

avendo

$$\omega_i = velocità\, rotazionale$$

$$i = \{x, y, z\}$$

La legge "rotazionale", dopo vari test effettuati con i nostri *PC program*, si è dimostrata non facilmente simulabile a causa dei diversi sistemi di riferimento utilizzati per la descrizione della posizione dello strumento (pinza) del robot e per i comandi di movimento rotazionale, forniti rispettivamente come angoli di Eulero e come angoli sessagesimali.

In particolare non risulta possibile comandare in modo affinato la velocità ed il movimento del robot sugli assi di rotazione.

Ogni volta che le variabili globali vengono aggiornate dal *PC program* si attiva un segnale interno del robot, più precisamente è stato utilizzato il segnale *2048*, che permetterà al programma principale di riconoscere l'avvenuto aggiornamento dei valori ed iniziare i calcoli per effettuare il nuovo movimento. Il movimento è svolto mediante il comando *XMOVE point TILL signal* che indica al robot in che posizione muoversi e di continuare seguendo quel movimento fintanto che non viene rilevato un nuovo fronte nel segnale apposito. Abbiamo inoltre utilizzato il segnale interno *2042*, con il quale lasciamo il robot libero di muoversi secondo le indicazioni ricevute, per la durata di un impulso di periodo pari alla metà del tempo necessario a percorrere la distanza precedentemente calcolata alla velocità comandata al robot secondo la formula riportata.

Il punto di destinazione del movimento viene calcolato in base alla velocità ricavata in precedenza, indicando gli spostamenti da svolgere dal punto in cui si trova il robot in quell'istante mediante i comandi *SHIFT* (per gli spostamenti traslazionali) ed *RX*, *RY*, *RZ* (per le rotazioni).

Il comando *XMOVE* è simile ad *LMOVE* in quanto comanda il movimento del robot in forma lineare. Il comando *JMOVE* non può essere utilizzato in quanto porta il robot a compiere dei movimenti extra al fine di portarsi nella posizione finale (il controllo svolto da questo comando è di posizione e non di traiettoria, questo porterebbe a movimenti dei giunti inappropriati rispetto alle posizioni precedenti).

L'utilizzo di *XMOVE* è quindi più vincolante, ma più appropriato ai fini di questa applicazione.

3.2. Interfacciamento con il sensore "libero" e calibrazione dei parametri

Al termine delle simulazioni abbiamo potuto testare il comportamento del nostro programma con il sensore e verificarne quindi la correttezza. Tale test ha portato ad una fase di adattamento minima dei parametri utilizzati all'interno del programma.

Questi primi test assumono un valore limitato, dal momento che i dati ottenibili dal sensore vincolato al robot saranno necessariamente differenti da quelli ottenibili manipolando il sensore liberamente (in particolare ne costituiranno un sottoinsieme). Ci hanno tuttavia fornito la confidenza necessaria ad effettuare dei test nel caso reale, ovvero con il sensore montato nella posizione finale, verificando così la correttezza completa del programma.

Questo passo finale ha comportato l'integrazione del nostro programma, *morz_follow*, con il *PC program* sviluppato dall'altro gruppo, *qggp_sensorUDP*. Come ci si poteva aspettare i valori reali, ottenuti tramite il sensore di forza, si sono rilevati piuttosto differenti rispetto ai valori generati dal simulatore da noi sviluppato, ovvero il *PC program morz_pc_gen*.

È stato necessario effettuare una fase di calibrazione al fine di concordare i valori di velocità utilizzati dal programma con i membri dell'altro gruppo.
Il fattore di scala per le velocità rotazionali (*morz_r_scale*) è stato fissato al valore 1000, mentre quello per le velocità traslazionali (*morz_v_scale*) è stato fissato al valore 100.
Come ultimo passo, per rendere il movimento più fluido, sono stati calibrati anche i parametri di accelerazione e decelerazione del robot. L'obbiettivo che ci eravamo prefissi era quello di prediligere un movimento più lento e fluido, piuttosto che più veloce e discontinuo, al fine di riprodurre i movimenti tipici di un lavoro di precisione e delicatezza umano. È stato quindi necessario diminuire l'accelerazione e la decelerazione dei movimenti del robot con i seguenti comandi[3]:

```
ACCEL 5 ALWAYS
DECEL 5 ALWAYS
```

Abbiamo notato che valori inferiori a 1 rendevano i movimenti eccessivamente lenti, rendendo così il robot inutilizzabile, mentre valori superiori a 10 rendessero i movimenti troppo veloci (e quindi inadatti allo scopo previsto). Dopo una breve fase di sperimentazione si è convenuto che 5 potesse essere il valore più adatto per ottenere la tipologia di movimento da noi desiderata.

Dopo avere riscalato le velocità (longitudinali e rotazionali) ed impostato adeguatamente i parametri di accelerazione e decelerazione siamo riusciti nell'intento di fare muovere il robot secondo i canoni che ci eravamo prefissi.

3.3. Interfacciamento con il sensore nella posizione finale

Il sensore montato nella posizione "reale", come prevedibile, risulta anche più vincolato rispetto ad i valori di velocità riproducibili.

Siamo stati fortunatamente in grado di provare il programma anche in questa condizione, mediante un supporto non definitivo per il sensore montato sulla flangia del robot, e questo ha portato ad un'ulteriore fase di calibrazione del programma (e correzione di alcuni errori riguardo la trasformazione di Eulero implementata) consentendoci così di poter presentare un lavoro completo.

4. Modalità operative

Per poter eseguire il programma nella modalità da noi utilizzata per il test, è necessario eseguire il *PC program* tramite il comando:

pcexec morz_pc_gen

L'esecuzione del programma di comando del robot avviene tramite il comando:

exec morz_follow

L'interfacciamento con il sensore di forza dovrebbe poter avvenire semplicemente sostituendo il *PC program morz_pc_gen* con un programma che assegni alle variabili di velocità i valori veri ottenuti dal sensore di forza, piuttosto che quelli simulati.

L'interfacciamento con il sensore di forza è stato svolto da un altro gruppo, che ha realizzato un server esterno al calcolatore del robot avente il compito di leggere i valori dal sensore di forza e mandarli via rete allo stesso. Qui, un *PC program* riceve i dati relativi alle velocità e li salva nelle opportune variabili globali da noi utilizzate. Allo stato attuale, il server non è facilmente trasferibile sul calcolatore usato per interfacciarsi con il robot Kawasaki ed è dunque necessario l'uso di un calcolatore esterno. Per avviare il *PC program* che legge i dati inviati dal server bisogna usare il comando:

pcexec qggp_sensorUDP

[3] Il valore espresso rappresenta la percentuale dell'accelerazione/decelerazione massima in uso.

4.1. Modalità di calibrazione

Se necessario, il movimento del robot può essere calibrato modificando i parametri elencati e spiegati nella Sezione 3.2.

4.2. Avvertenze

Certi movimenti effettuati dal robot potrebbero risultare troppo snodati per le articolazioni umane (polso e gomito), a causa del fatto che, per effettuare un movimento rotazionale, il robot muove quasi sempre tutti i suoi giunti.

5. Conclusioni e sviluppi futuri

Il movimento da noi ottenuto risulta piuttosto "morbido", ma deve essere tenuto presente che il robot Kawasaki in uso non è progettato per seguire dei movimenti umani.

Questo ragionamento si applica in particolare ai movimenti rotazionali che, come evidenziato nella sezione di *Avvertenze*, vengono eseguiti dal robot in modo molto diverso rispetto a come verrebbero eseguiti da un operatore umano.

L'applicazione potrà essere estesa con l'inserimento di diverse modalità di utilizzo del robot, per esempio se ne potrebbero inserire due distinte nelle quali è possibile far muovere il robot in modo molto più libero così che possa compiere grandi distanze in breve tempo ed una, quella attuale, dove si fa muovere il robot in modo molto più raffinato e preciso.

Indice