IMPLEMENTACIÓN DE CONTROL DE FUERZAS EN ROBOTS INDUSTRIALES: UN CASO

A. F. Brumovsky, [1] **V. M. Liste,** [1] **y M. Anigstein** [2]

Laboratorio de Robótica-FIUBA, abrumov@fi.uba.ar, vliste@fi.uba.ar, manigst@fi.uba.ar

Resumen: Para que la implementación de control de fuerzas en robots industriales pueda realizarse, es necesario que el controlador permita modificar trayectorias cartesianas en tiempo real. Los controladores de los robots industriales más populares tienen esta capacidad desde hace aproximadamente veinte años. Sin embargo, robots industriales de última generación pueden no tenerla (aun cuando la declaren en sus manuales). El presente artículo analiza justamente un caso. Se describen prestaciones reales del controlador de un robot industrial, limitaciones resultantes de esas prestaciones y desarrollos realizados para lograr una solución alternativa que permita implementar de todas maneras un sistema de control de fuerzas en el robot. El trabajo puede ser útil para potenciales usuarios y al mismo tiempo resulta didáctico desde un punto de vista teórico y general.

Keywords: control de fuerza/torque, robots industriales, corrección de trayectoria cartesiana en tiempo real.

1. INTRODUCCIÓN

Los movimientos que realiza un robot industrial, resultan de la ejecución de un programa de usuario por el controlador del robot. El programa define la posición y orientación de los puntos hacia los que el robot debe moverse y el tipo de trayectoria a seguir. Este método funciona muy bien para aplicaciones en que las trayectorias son totalmente predecibles. Sin embargo, muchas aplicaciones involucran cierto grado de incerteza en las posiciones/orientaciones y movimientos. En estas aplicaciones, pueden utilizarse entradas de sensores para proveer información al robot de manera que pueda adaptarse a las condiciones cambiantes.

Cuando la entrada de sensores se usa para modificar movimientos del robot mientras éstos están efectivamente ejecutándose, el proceso se denomina "real-time path control". Cuando este modo de control de trayectoria está en operación, *VAL-II* (lenguaje de programación, sistema de control y sistema operativo de Westinghouse/Unimation) espera una instrucción con una corrección de las referencias que contiene 6 argumentos

- 3 componentes de desplazamiento $< \delta x >, < \delta y >, < \delta z >$
- 3 componentes de rotación $< rx >, < ry >, < rz >$

y la ejecuta cada 28 mseg. pasando las correcciones al controlador del robot. Pueden ser referidas a la terna *mundo* o *herramienta* y ser acumulativas o no. El robot no necesita estar moviéndose realmente para responder a las correcciones.

Los párrafos anteriores fueron extraídos del Manual de Programación del PUMA de Unimate Industrial Robots (1986), (Masagué, 1996). Son

[1] Tesistas de grado en Ing. Electrónica FIUBA.
[2] Proyecto UBACyT I003.

igualmente válidos, pero con una sola corrección en rotación por ser de 4 ejes, para otro robot industrial clásico, el AdeptOne SCARA (Adept, 1984) que utiliza también *VAL-II*. El lenguaje PDL2 de Comau SMART (1999) denomina a la corrección de trayectoria en tiempo real "sensor tracking", y el lenguaje RAPID de ABB la llama "contour tracking" y la incorpora en su módulo de Movimientos Avanzados (2000). PDL2 permite correcciones relativas y absolutas en traslación y orientación, proyectadas en cualquier sistema de referencia (mundo, herramienta, usuario), con y sin movimiento programado. RAPID requiere expresar las correcciones de tiempo real en "el sistema coordenado de la trayectoria".

1.1 Objetivo del artículo

En el presente trabajo se discuten aspectos críticos del controlador de un robot industrial, que definen la posibilidad de articular los resultados de los trabajos de investigación e implementar sistemas de control de fuerzas. Para hacerlo, se trata a modo de ejemplo el caso de un robot de última generación deficiente en este aspecto aun cuando declare la capacidad de corrección de trayectoria en tiempo real en sus manuales. Pese a estudiar comportamientos de un robot particular, la tarea de análisis junto con la explicación de los desarrollos realizados, resulta útil desde un punto de vista teórico y general.

En un artículo anterior (Liste y Anigstein, 2004*b*) se presentaron resultados obtenidos con un sistema de control de fuerzas implementado sobre un robot industrial y utilizando las funciones de corrección de trayectoria en tiempo real del módulo de Movimientos Avanzados del robot. Las aplicaciones mencionadas no incluían tareas en que se controlaran torques. Tampoco movimientos en que la herramienta modificara su orientación mientras se desplazaba. En este trabajo se describen limitaciones de las funciones de corrección de trayectoria del robot que no fueron mencionadas antes, y que justamente son las que impiden realizar ese tipo de tareas. Frente al reducido rango de aplicación resultante, se presenta entonces el desarrollo de un nuevo paquete de software para reemplazar las funciones de corrección de trayectoria (Movimientos Avanzados) del robot. El nuevo software permite el abordaje de tareas más generales de control de fuerzas.

El trabajo se organiza como sigue. En el próximo capítulo se hace una rápida descripción del robot con el sensor de fuerza/torque. Se explican las funciones del controlador, diseñadas por el fabricante para la corrección de trayectorias en tiempo real y se señalan y discuten defectos y omisiones. En el capítulo 3 se describe la alternativa propuesta, dirigida a lograr una solución que permita incorporar de todas maneras un sistema de control de fuerzas general en el robot. En el capítulo 4 se describen las nuevas funciones desarrolladas y se muestran aplicaciones.

2. ESQUEMA DEL CONTROL DE FUERZAS ANTERIOR

Se utiliza un robot industrial IRB 140 con controlador *S4Cplus M2000/BaseWare OS 4.0* y un sensor de fuerza/torque de 6 direcciones marca ATI - modelo SI-65-5, ambos provistos por ABB. El sensor fue montado en la muñeca del robot y conectado al controlador del mismo a través de una interfaz analógica (Fig. 1).

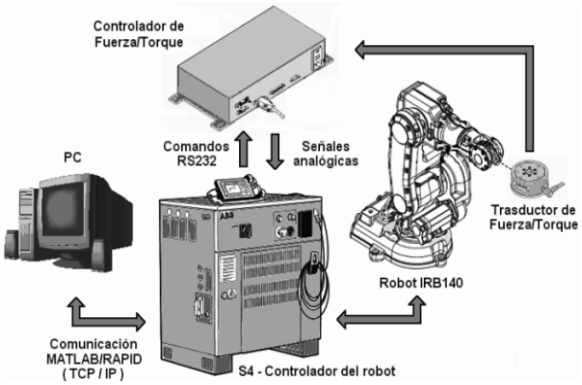

Fig. 1. Esquema de control

Se desarrolló un programa en RAPID consistente en una serie de procesos que pueden ser accedidos via Ethernet desde Matlab mediante un protocolo RPC/RAP (Roaux *et al.*, 2003) que permite conectarse con el controlador del robot en tiempo real. Se creó además un toolbox en Matlab (Liste y Anigstein, 2004*a*) con un conjunto de funciones primitivas a partir de las cuales pueden generarse funciones de complejidad creciente.

El controlador de ABB, como en general todos los controladores industriales, está basado en control de posición y no permite acceder a los torques de control. En consecuencia, la estrategia de control que se implementó, crea una estructura de control de fuerzas alrededor del lazo de control de posición y se asemeja al control de rigidez de Salisbury (1980). Utiliza una matriz de *acomodamiento*, $\mathbf{G} = diag[g_1, \ldots, g_6]$, para transformar el error de fuerza/torque $\left(\mathbf{F} - \mathbf{F}^d\right)$, en un vector $\delta\mathcal{X}^d$ que corrige las referencias

$$\delta\mathcal{X}^d = \mathbf{G}\left(\mathbf{F} - \mathbf{F}^d\right) \quad ; \quad \mathbf{F} = \left[\, \mathbf{f}^t \;\; \mathbf{M}^t \,\right]^t \qquad (1)$$

2.1 Módulo de Movimientos Avanzados

El controlador posee un módulo de Movimientos Avanzados ofrecido por el proveedor para implementaciones con sensores, que permite realizar

correcciones sobre la trayectoria cartesiana que se está ejecutando a través de las funciones *CorrCon* y *CorrWrite* y mediante la ejecución de una interrupción cíclica cuya frecuencia máxima es de 4 Hz. La terna de correcciones queda definida por la terna herramienta (*tool*) y la dirección de movimiento (Fig. 2).

- $\mathbf{x}_{corr}$ es tangente a la trayectoria
- $\mathbf{y}_{corr} = \mathbf{x}_{corr} \times \mathbf{z}_{tool}$
- $\mathbf{z}_{corr} = \mathbf{x}_{corr} \times \mathbf{y}_{corr}$

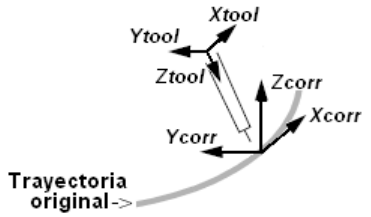

Fig. 2. Terna del generador de correcciones

2.2 Limitaciones

Las limitaciones causadas por la baja frecuencia de correcciones permitida, ya fueron tratadas en un trabajo reciente (Liste y Anigstein, 2006).

Se describen a continuación dificultades asociadas a la particular forma de definir la terna: (i) la terna para realizar las correcciones es única, depende de la trayectoria y no puede ser elegida por el usuario; (ii) cambia al cambiar el sentido del movimiento; (iii) la dirección **z** de la herramienta no puede coincidir con la dirección del movimiento. Entonces, los movimientos de aproximación a un punto objetivo no pueden hacerse como es usual en la dirección **z** de la herramienta, y la terna de correcciones no puede ser en general la terna de acomodamiento (*comply*), necesaria para tareas con control de fuerzas (Mason, 1981), (Craig, 1986), (De Schutter and Van Brussel, 1988). Sin embargo, estas limitaciones pudieron resolverse en los trabajos que se mencionaron antes, a costa de hacer más pesado y menos transparente el código de los programas desarrollados. Pero existe otra consecuencia de la definición de la terna que no tiene solución: no se pueden hacer correcciones con las funciones *CorrCon* y *CorrWrite* si no hay trayectoria. Por ejemplo, no se pueden controlar fuerzas con la herramienta en una posición fija.

2.2.1. Limitaciones fundamentales. A estos inconvenientes se suman otras deficiencias fundamentales que no figuran en los manuales del robot, que no se mencionaron en los trabajos anteriores y que decidieron la búsqueda de una solución alternativa. Con las funciones *CorrCon* y *CorrWrite*: (i) se pueden realizar correcciones a la trayectoria sólo en dos direcciones de traslación; (ii) no se puede corregir la orientación. Es decir, pueden corregirse sólo dos de los seis grados de libertad (g.d.l.).

3. SOLUCIÓN PROPUESTA

La solución alternativa a diseñar debe permitir: (1) realizar correcciones en los seis g.d.l., (2) definir la terna de correcciones en forma arbitraria y por lo tanto coincidente con la terna comply adecuada para cada tarea con control de fuerzas. En la figura 3 se muestra: (i) un

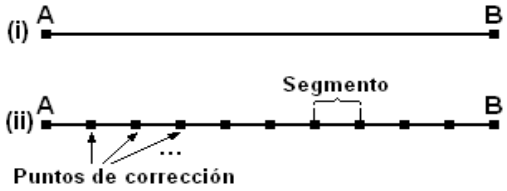

Fig. 3. Movimiento lineal segmentado

movimiento lineal entre dos puntos A y B, y (ii) el mismo movimiento segmentado o dividido en una sucesión de movimientos lineales. El primero es el caso del generador de correcciones clásico de los controladores, que realiza las correcciones durante el movimiento AB. El segundo corresponde a la solución adoptada, donde la trayectoria AB es segmentada en una sucesión de movimientos lineales en los que cada punto destino es programado como *punto de paso*. Cada punto de paso es corregido en traslación y orientación en respuesta a las señales de fuerza y torque sensadas.

Una desventaja del nuevo método es que la frecuencia de correcciones debió bajarse a 2 Hz para que el robot pueda efectivamente seguir los sucesivos puntos de paso.

3.1 Transformación de las correcciones

Cada tarea particular con control de fuerzas tiene una terna comply que resulta conveniente, y que se elige siguiendo la metodología usual (ver bibliografía en la sec. 2.2). En el software desarrollado, las ternas de sensado y tool se hacen coincidir con la terna comply elegida. Las correcciones a la trayectoria que se expresan en la terna comply, sirven entonces directamente para corregir los puntos de paso. Luego, esas correcciones son transformadas a la terna de referencia de la tarea y acumuladas en la misma (Paul, 1981).

3.2 Cálculo de correcciones

Para distribuir la carga de cómputo, a Matlab se le asignó: (i) el cálculo del vector fuerza/torque,

a partir de la lectura de los valores sensados y comparación con los deseados; (ii) el cálculo de las correcciones en la trayectoria utilizando la matriz de acomodamiento (Ec. 1); (iii) la transformación y acumulación de correcciones en la terna de referencia. Estas correcciones son comunicadas a través de RPC/RAP al programa en RAPID que se encuentra ejecutando los segmentos de la trayectoria original. Con esta información se modifica la terna de referencia y en consecuencia las coordenadas del punto de paso siguiente.

4. NUEVAS FUNCIONES DESARROLLADAS Y APLICACIONES

4.1 Movimiento lineal con correcciones

Una función básica de las tareas con control de fuerzas es el movimiento lineal con *objetivos* en posición y fuerzas, y condiciones de *terminación*. Es invocada por Matlab como CF_moveL(parametros); (parametros):(controlar,objetivo,terminacion,delta_f,v_move). Los tres primeros son arreglos de 6 elementos asociados a los 6 g.d.l. controlar: 1, si en la dirección se controla fuerza/torque; 0, si se controla traslación/rotación. objetivo: la referencia para el control. terminacion: la condición de terminación en la dirección (0, si no hay condición). Los dos últimos son escalares.delta_f: máxima variación de fuerzas permitida. Se utiliza para ajustar la velocidad del movimiento. v_move: [optativo] permite ingresar la velocidad deseada para el movimiento (tiene prioridad sobre la calculada con delta_f).

4.1.1. Un primer ejemplo de aplicación. La tarea consiste en buscar una superficie cuya posición y orientación se conoce con incerteza, luego buscar un agujero sobre la superficie mientras se identifica su dirección normal, reorientar la herramienta (perno) e insertarla en el agujero. Como la reorientación no se realiza *dentro* de los movimientos, no se controlan torques y se utilizan correcciones de traslación sólo en dos direcciones, la tarea podría ser realizada también con el esquema de control de fuerzas anterior.

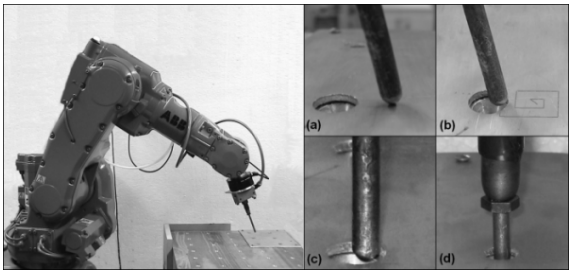

Fig. 4. Inserción con identificación de normal.

En la figura 4, la imagen izquierda muestra al robot al iniciar la tarea y en las 4 imágenes de la derecha puede observarse:

(a) el final de las primeras dos subtareas: (1) CF_moverL_stop_on_force(offset, F_test). offset es el desplazamiento de búsqueda de la superficie en mm. en la dirección del eje del perno. F_test es la fuerza umbral para la detención por choque y define la velocidad de avance; (2) [k_est,dir_normal] = CF_identificar_normal(). k_est es la constante elástica del sistema (10 Kgf/mm en el ejemplo) que se utiliza para definir la ganancia del lazo de fuerza para los movimientos que siguen (Liste y Anigstein, 2004*b*). dir_normal a la superficie (estimada con error de 2.5° en el ejemplo, ver Fig. 5). Se utiliza para definir la terna comply que se ubica en el extremo del perno con $\mathbf{z}_{comply} \parallel$ dir_normal.

(b) un patrón de búsqueda, CF_buscar_agujero(), que contiene una serie de CF_moveL(parametros) con parametros expresados en la terna comply, de la forma: controlar = [0 0 1 0 0 0] control de fuerza sólo en **z**; objetivo = [20 0 -1 0 0 0] desplazamiento de 20 mm. en **x** controlando 1 Kgf en -**z** y manteniendo constante la orientación; terminacion = [0 0 4 0 0 0] agujero encontrado cuando delta en **z** supera 4 mm (siempre magnitud de naturaleza contraria a la controlada). Además, en cada movimiento lineal del patrón, utilizando los datos de corrección se obtiene una mejor identificación de la normal. La figura 5 muestra los resultados de la búsqueda para una f_z^d = -1 Kgf y ganancia g_z = 1/k_est = 0.1 mm/Kgf. En el gráfico superior se indica la fuerza real aplicada en la dirección **z**, y sobre la finalización de cada tramo del patrón de búsqueda se indica el error entre la normal identificada en dicho tramo y un valor de referencia obtenido a partir de la medición de tres puntos de la superficie. Puede verse la evolución del error a partir de los 2.5° obtenidos con CF_identificar_normal(). En el gráfico inferior se indica la corrección acumulada en **z** durante la búsqueda. Nótese el incremento brusco, condición de terminación, al detectar el agujero (también la caída en f_z).

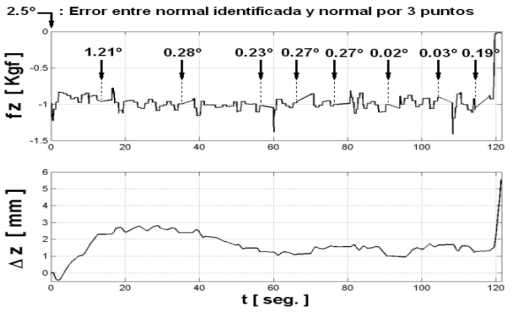

Fig. 5. Fuerza y correcciones en la búsqueda.

(c) encontrado el agujero se reorienta el perno.

(d) tarea de inserción ejecutándose, con CF_moveL (parametros) y parametros: controlar = [1 1 0 0 0 0] control de fuerza en **x** e **y**; objetivo = [0 0 100 0 0 0] desplazamiento de 100 mm. en **z** controlando 0 Kgf en **x**, **y**

y manteniendo constante la orientación; terminacion = [0 0 -4 0 0 0] fondo encontrado cuando la fuerza en -**z** supera 4 Kgf.

4.2 Movimiento con reorientación

En los movimientos lineales sin cambio de orientación, las mediciones con el sensor de fuerzas se realizan fuera de las zonas de aceleración para evitar que los efectos dinámicos se sumen a las fuerzas de contacto. Pero cuando hay reorientación durante el movimiento esto no es suficiente. Inevitablemente, las mediciones del sensor incluirán, además de las fuerzas de contacto, los esfuerzos estáticos provocados por el cambio de orientación de la herramienta. Es necesario entonces realizar una identificación previa de la masa y del momento de primer orden de la herramienta (Atkeson *et al.*, 1986).

Se desarrolló un procedimiento que aprovecha el mismo sensor de fuerzas y utiliza tres orientaciones distintas de la herramienta ((a),(b),(c) en la figura 6). La función CF_estimar_parametros_tool() devuelve además del peso y centro de gravedad en la terna comply, la dirección de la gravedad en la terna mundo. Los parámetros dinámicos iden-

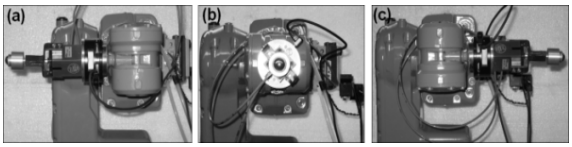

Fig. 6. Identificación de la bola patín.

tificados son utilizados en cada punto de sensado para restar de las mediciones los valores debidos al cambio de orientación.

4.2.1. Ejemplo: inserción con cambio de orientación y control de torque. En esta tarea no se identifica previamente la normal a la superficie. En cambio, se utiliza el torque provocado por el contacto con el agujero durante la inserción, para corregir la orientación del perno. Debe emplearse el nuevo esquema de control de fuerzas, incluyendo la compensación del peso de la herramienta.

La herramienta perno se identifica antes de iniciar la tarea. Luego, se define la terna comply en la

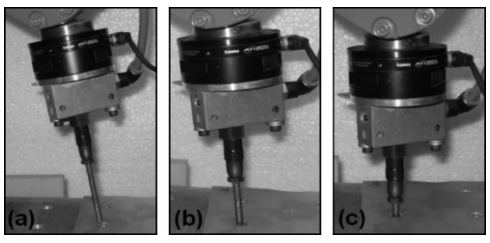

Fig. 7. Inserción por control de torque.

punta del perno, con **z** orientada como el eje del mismo. Ahora, la función CF_moveL(parametros) tiene los parametros: controlar = [1 1 0 1 1 0] control de fuerza y torque en **x** e **y**; objetivo = [0 0 100 0 0 0] desplazamiento de 100 mm. en **z** controlando 0 Kgf y 0 Nm en **x**, **y**; terminacion = [0 0 -4 0 0 0] fondo encontrado cuando la fuerza en -**z** supera 4 Kgf; v_move = [0.2] velocidad de inserción de 0.2 mm/seg. La figura 7 muestra tres instantes, (a),(b),(c) de la tarea realizada. En la figura 8, se observa como el perno orientado inicialmente con un error de 10º, comienza a reorientarse debido al torque y deja de hacerlo cuando éste se anula. Sin pérdida de generalidad y para ganar claridad en la interpretación de los resultados, los ejes **x**, **y** de la terna comply se eligieron en este y el siguiente ejemplo, de tal manera que la reorientación ocurra alrededor de uno de ellos.

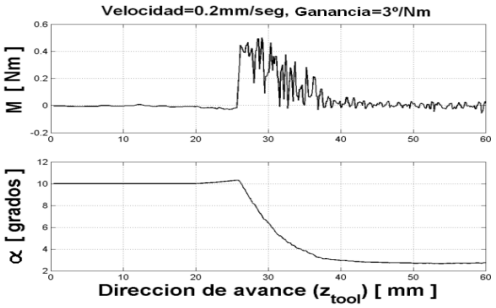

Fig. 8. Momento y reorientación en inserción.

4.2.2. Ejemplo: seguimiento del contorno de una superficie. Debe realizarse el seguimiento de una superficie de forma desconocida, manteniendo siempre la herramienta normal a la misma. Se utiliza como herramienta una pieza cilíndrica con una bola patín en el extremo.

Se ubica la terna comply en el centro de la bola y con **z** orientada como el eje del cilindro. La herramienta se identifica como se indica en la sec. 4.2 ya que es preciso compensarla durante la tarea. Se utiliza una función similar al movimiento lineal con control de fuerzas, pero desafectando el control de orientación de la matriz de acomodamiento (Ec. 1), porque el contacto no provoca torque en la terna comply. En cambio, se utiliza la fuerza sensada, **f**, normalizada, para estimar la dirección normal: dir_normal = $-\mathbf{f}/\|\mathbf{f}\|$. El cálculo incluye un desestimador de datos con un umbral del 40% sobre el módulo de la fuerza deseada, f^d, en el contacto. Los parametros de la función CF_moverL_superficie(parametros) son: controlar = [0 0 1 0 0 0] control de fuerza en **z** y orientación no utilizada; objetivo = [0 160 -2 0 0 0] desplazamiento de 160 mm. en **y** controlando $f^d = 2$ Kgf en $-\mathbf{z}$; terminacion = [0 0 0 0 0 0] sin condición (termina al cumplir el desplazamiento); v_move = [0.5] velocidad de desplazamiento 0.5 mm/seg. En la figura 9

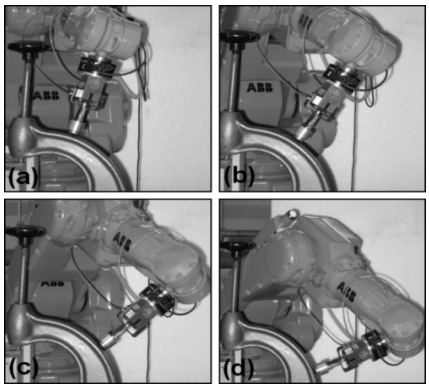

Fig. 9. Seguimiento de contorno.

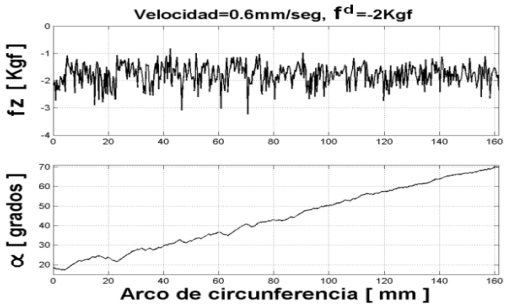

Fig. 10. Fuerza y reorientación.

se ven cuatro imágenes, (a),(b),(c),(d) del robot durante la tarea. En la figura 10, el gráfico superior muestra la fuerza de contacto durante el seguimiento (la herramienta nunca se separa de la superficie) y el inferior permite ver la evolución en la reorientación de la herramienta.

5. CONCLUSIONES

Se analizó la capacidad de corregir trayectorias cartesianas en tiempo real, necesaria para implementar sistemas de control de fuerzas en los robots industriales. Se trató el caso de un robot de última generación que tiene funciones especialmente ofrecidas por el fabricante con este objetivo. Se explicaron las limitaciones de esas funciones, que hacen depender de la trayectoria la terna de correcciones e impiden controlar: (i) torque, (ii) orientación, (iii) fuerza si el robot no está en movimiento.

Se propuso un nuevo esquema para la corrección de trayectorias cartesianas que permite superar las limitaciones apuntadas y abordar problemas generales de control de fuerzas. Se describió el nuevo software desarrollado y se mostraron algunas aplicaciones típicas: (i) búsqueda e inserción de un perno en un agujero, sin y con control de torque/orientación; (ii) seguimiento de una superficie con la herramienta siempre normal a la misma.

De todas maneras, la solución desarrollada no resuelve el problema de la muy baja frecuencia de correcciones permitida por el controlador del robot. En estas condiciones, las bajas velocidades de desplazamiento logrables en las tareas de control de fuerzas resultarían ineficientes en un ambiente fabril. Sería necesaria una frecuencia de correcciones por lo menos diez veces superior a la disponible.

REFERENCIAS

ABB, Flexible Automation (2000). *Advanced Motion, Product Specification RobotWare for BaseWare OS*. 4.0 ed. Vasteras, Sweden.

Adept (1984). *The AdeptOne Controller*. Adept Technology, Inc. Sunnyvale, California.

Atkeson, C.G., C.H. An and J.M. Hollerbach (1986). Estimation of inertial parameters of manipulator loads and links. *Int J Robot Res* **5**(3), 101–119.

Comau (1999). *Robot SMART S2, Guía de Utilización*. versión 4.0x ed.. Industrial Robots. Turin, Italia.

Craig, J. J. (1986). *Introduction to Robotics, Mechanics & Control*. Addison-Wesley, Mass.

De Schutter, J. and H. Van Brussel (1988). Compliant robot motion I. A formalism for specifying compliant motion tasks. *Int J Robot Res* **7**(4), 3–17.

Liste, V. y M. Anigstein (2004*a*). Control de fuerzas en robots, un toolbox en Matlab. *XIX Congr Arg Contr Automát*. ID41. AADECA. Buenos Aires.

Liste, V. y M. Anigstein (2004*b*). Sobre control de fuerza/torque. *III Jornadas Argentinas de Robótica*. JAR0427. San Juan.

Liste, V. y M. Anigstein (2006). Sobre control de fuerzas en robots, rigidez del sistema y frecuencia de correcciones. *XX Congr Arg Contr Automát*. ID47. AADECA. Buenos Aires.

Masagué, G. (1996). Estudio de la automatización de una tarea utilizando un robot PUMA 500. Tesina de Graduación, FIUBA - Avietsa SA.

Mason, M. T. (1981). Compliance and force control for computer controlled manipulators. *IEEE Trans Syst Man Cybern* **6**, 418–432.

Paul, R. P. (1981). *Robot Manipulators: Mathematics, Programming and Control*. The MIT Press, Cambridge, Mass.

Roaux, M., P. Ilardi and M. D'Ascanio (2003). Desarrollo de interfaz Matlab/RAP para el controlador de ABB M2000. Trab. Final de Taller de Prog. III y Robótica, FIUBA.

Salisbury, J. K. (1980). Active stiffness control of a manipulator in cartesian coordinates. In: *19th IEEE Conf Decision Cont*. Albuquerque, Nueva México.

Unimate (1986). *Programming Manual, User's Guide to VAL II*. version 2.0 ed.. Industrial Robots. Danbury, Connecticut.