Plataforma robótica industrial para ensayos de resolución de redundancia

Carlos J. Cartelli - Mauricio Anigstein
Laboratorio de Robótica. Facultad de Ingeniería, Universidad de Buenos Aires.
{cjcartelli@yahoo.com}-{manigst@fi.uba.ar}

Resumen— **El trabajo describe un sistema para ensayar algoritmos de resolución de redundancia cinemática de manipuladores. A diferencia de la mayoría de los trabajos existentes sobre el tema, las trayectorias calculadas durante cada intervalo de cómputo no sólo se simulan, sino que además se ejecutan en línea en un robot industrial típico. De manera simultánea al movimiento se registran los resultados reales de cada ensayo. Contando con esta nueva herramienta es posible analizar, evaluar y comparar de manera ágil distintas opciones de resolución de redundancia en un manipulador industrial.**

I. Introducción

Como un manipulador necesita de forma general seis grados de libertad para posicionar y orientar su herramienta, aquellos manipuladores que poseen siete grados de libertad se dicen redundantes. Sin embargo, la redundancia cinemática existe siempre que el manipulador tiene más grados de libertad que los que requiere la tarea a ejecutar. En consecuencia, estrictamente hablando, no existen manipuladores cinemáticamente redundantes sino que es la tarea a realizar la que les puede conferir esa propiedad [1]. Cuando la redundancia se evidencia, la tarea puede realizarse por infinitas combinaciones de movimiento de los ejes. El criterio de selección de una de esas posibles combinaciones define el problema de resolución de redundancia. Por un lado, la redundancia le aporta al robot mayor versatilidad en tareas de movimiento, y también mayor confiabilidad ya que dado el caso de falla en alguna articulación ciertas tareas podrían, de todas maneras, ser completadas con éxito. Pero también implica que el robot tiene una estructura mecánica más complicada que la necesaria para cumplir el objetivo primario, y además requiere un software de control más complejo para ejecutar movimientos. La literatura del tema [2], [3], [1], [4] plantea opciones para que, mientras se cumple la trayectoria deseada, se utilicen los grados de libertad remanentes para, por ejemplo: (i) minimizar velocidades de los ejes, (ii) alejarse de configuraciones singulares, y (iii) evitar obstáculos y topes de articulaciones. Sin embargo, los lenguajes de programación empleados en manipuladores industriales de seis ejes exigen definir completamente las seis dimensiones, tres para posición y tres para orientación, de todos los puntos objetivos del movimiento.

Por lo tanto se evidencian dos problemas no resueltos:
1. Cómo implementar algoritmos de resolución de redundancia sobre una plataforma industrial que no cuenta con comandos nativos para ese propósito, y
2. Cómo desarrollar esos comandos mediante una codificación amigable para el usuario/operador.

Por este motivo es conveniente contar con un sistema centrado en un robot industrial que permita concretar ensayos de los distintos algoritmos que se van estudiando y desarrollando en el Laboratorio. El artículo describe la implementación de este sistema, y muestra los resultados obtenidos con el mismo.

A. Organización del artículo

En la próxima sección se introduce el método de resolución de redundancia utilizando prioridad de tareas y la pseudoinversa de la matriz jacobiana [5]. Se describen también tres criterios para satisfacer una tarea secundaria aprovechando la redundancia. La sección siguiente presenta y justifica la arquitectura de hardware y software del sistema de ensayos. La sección IV muestra los tres criterios aplicados y ensayados sobre el sistema, y los resultados obtenidos. Finalmente la sección V expone las conclusiones y trabajos futuros.

II. Teoría de resolución de redundancia por prioridad de tareas

En los manipuladores industriales cada movimiento a realizar en el espacio cartesiano requiere que el controlador en tiempo real genere una trayectoria deseada $\boldsymbol{r}^d(t)$ y resuelva el problema inverso de posición. Se obtiene así una trayectoria de ejes deseada $\boldsymbol{\theta}^d(t)$ que le permite cumplir $\boldsymbol{r}^d(t)$ [6]. Una opción alternativa para generar la trayectoria de ejes es la técnica de resolución del movimiento mediante control de velocidad [7]. Se parte de la ecuación de velocidades del manipulador en donde la velocidad $\dot{\boldsymbol{r}}$ (vector de dimensión seis) del TCP (punto de interés de la herramienta manipulada) en el espacio de la tarea puede obtenerse en base a la matriz jacobiana $\boldsymbol{J}$. Esto es,

$$\dot{\boldsymbol{r}} = \boldsymbol{J}\dot{\boldsymbol{\theta}} \tag{1}$$

donde $\dot{\boldsymbol{\theta}}$ es el vector de dimensión seis formado por la velocidad de cada uno de los ejes del manipulador. La técnica de resolución consiste en invertir (1) para obtener el vector $\dot{\boldsymbol{\theta}}$, luego integrarlo y conformar en cada instante el objetivo de control de los actuadores del robot. En efecto, un manipulador industrial típico de seis ejes tiene una matriz jacobiana $\boldsymbol{J}$ cuadrada, que en general puede invertirse si el manipulador no se encuentra en una singularidad.

Pero para poder generar movimientos con redundancia será necesario declarar tareas de dimensión menor que seis. En estos casos la matriz jacobiana $\boldsymbol{J}$ de la tarea es rectangular, y por lo tanto su inversa no está definida. Sin

embargo la solución general de (1) puede calcularse [2] para obtener

$$\dot{\boldsymbol{\theta}} = \boldsymbol{J}^{+}\dot{\boldsymbol{r}} + (\boldsymbol{I} - \boldsymbol{J}^{+}\boldsymbol{J})\boldsymbol{v} \tag{2}$$

donde $\boldsymbol{J}^{+}$ es la pseudo-inversa de $\boldsymbol{J}$. El primer término de la ecuación representa la solución de norma mínima para $\dot{\boldsymbol{\theta}}$, y el segundo la solución homogénea. Esta última es no nula si $(\boldsymbol{I} - \boldsymbol{J}^{+}\boldsymbol{J}) \neq \boldsymbol{0}$, dando como resultado la posibilidad de múltiples soluciones para $\dot{\boldsymbol{\theta}}$. Como este término se proyecta en el espacio nulo de $\boldsymbol{J}$, permite configurar el manipulador sin afectar la *tarea principal* que consiste en seguir la trayectoria deseada. En estos casos se manifiesta la redundancia cinemática del problema, evidenciada por la capacidad de elección 'arbitraria' del vector $\boldsymbol{v}$ dentro del espacio de velocidad de los ejes.

Resolver un problema de redundancia cinemática con este planteo consiste en elegir en tiempo real un vector $\boldsymbol{v}$. Esta selección es la que puede plantearse como una capacidad extra del manipulador para cumplir además *criterios o tareas secundarias*, utilizando los grados de libertad que la redundancia le aporta. Se describen a continuación tres criterios secundarios que se ensayaron sobre el sistema.

A. Minimización de velocidad de ejes

Este caso es el más simple de implementar en base a (2). Basta con elegir $\boldsymbol{v} = \boldsymbol{0}$, con lo que se obtiene

$$\dot{\boldsymbol{\theta}} = \boldsymbol{J}^{+}\dot{\boldsymbol{r}} \tag{3}$$

B. Objetivo de posición de ejes

Este caso plantea cumplir la tarea principal pero a su vez buscar que el manipulador se acerque a una posición de ejes $\boldsymbol{\theta}_{2e}$ determinada. La elección de esa posición depende del objetivo de la tarea secundaria. Por ejemplo, una posición $\boldsymbol{\theta}_{2e}$ en la que el manipulador no colisione con algún obstáculo durante la tarea principal. Si $\boldsymbol{\theta}_{2e}$ se define constante, $\boldsymbol{v}$ puede elegirse como

$$\boldsymbol{v} = \boldsymbol{H}(\boldsymbol{\theta}_{2e} - \boldsymbol{\theta}) \tag{4}$$

donde $\boldsymbol{H}$ es una matriz cuadrada con factores de ganancia en su diagonal. Esta elección minimiza la norma de $\dot{\boldsymbol{\theta}} + \boldsymbol{v}$ por tratarse de un caso particular del analizado en [2], e implica que la posición de ejes del manipulador se acerca a $\boldsymbol{\theta}_{2e}$ mientras se cumple también el objetivo principal. Reemplazando (4) en (2), se obtiene

$$\dot{\boldsymbol{\theta}} = \boldsymbol{J}^{+}\dot{\boldsymbol{r}} + (\boldsymbol{I} - \boldsymbol{J}^{+}\boldsymbol{J})\boldsymbol{H}(\boldsymbol{\theta}_{2e} - \boldsymbol{\theta}) \tag{5}$$

C. Movimiento de punto obstáculo

En este caso se cumple la tarea principal pero tratando además de que un punto del manipulador se mueva a una velocidad $\dot{\boldsymbol{r}}_{\boldsymbol{2a}}$. La principal aplicación de esta técnica es evitar que el punto elegido se acerque a un obstáculo. La solución que minimiza la norma del error de velocidad del punto obstáculo surge de elegir $\boldsymbol{v}$ como

$$\boldsymbol{v} = [\boldsymbol{J}_{2a}(\boldsymbol{I} - \boldsymbol{J}^{+}\boldsymbol{J})]^{+}(\dot{\boldsymbol{r}}_{2a} - \boldsymbol{J}_{\boldsymbol{2a}}\boldsymbol{J}^{+}\dot{\boldsymbol{r}}) \tag{6}$$

donde $\boldsymbol{J}_{2a}$ es el jacobiano que transforma velocidades de ejes en velocidad del punto obstáculo, esto es

$$\dot{\boldsymbol{r}}_{2a} = \boldsymbol{J}_{2a}\dot{\boldsymbol{\theta}} \tag{7}$$

Finalmente, reemplazando (6) en (2), y como la matriz $(\boldsymbol{I} - \boldsymbol{J}^{+}\boldsymbol{J})$ es simétrica e idempotente [8], resulta

$$\dot{\boldsymbol{\theta}} = \boldsymbol{J}^{+}\dot{\boldsymbol{r}} + [\boldsymbol{J}_{2a}(\boldsymbol{I} - \boldsymbol{J}^{+}\boldsymbol{J})]^{+}(\dot{\boldsymbol{r}}_{2a} - \boldsymbol{J}_{\boldsymbol{2a}}\boldsymbol{J}^{+}\dot{\boldsymbol{r}}) \tag{8}$$

III. Arquitectura de hardware y software

Como los lenguajes de programación industrial de robots de seis ejes imponen la definición completa de cada punto objetivo de trayectoria, no es posible plantear tareas de dimensión menor que seis con los comandos nativos que las plataformas industriales incluyen. El propósito central de la arquitectura aquí desarrollada apunta a superar esa limitación, y así permitir ejecutar tareas de menor dimensión que confieran al manipulador la capacidad de redundancia. Con esa idea se diseñó el sistema de ensayos HIL (hardware-in-the-loop) presentado.

Desde el punto de vista de hardware, se plantea una celda robótica integrada por:
a. Un robot industrial antropomorfo ABB IRB140 con controlador SC4plus; y protocolo de comunicación RAP.
b. Una red de PCs de escritorio; con capacidad para computar, almacenar, procesar y mostrar datos de trayectorias deseadas y realizadas.
Los equipos se encuentran interconectados a través de una red Ethernet. Si bien la arquitectura empleada en el sistema es similar a la presentada en trabajos previos del Laboratorio [9], [10], en este caso los objetivos principales de la plataforma son:

- priorizar la comunicación de objetivos de posicionamiento desde las PCs hacia el controlador del robot,
- liberar capacidad de procesamiento en la PC que genera la trayectoria,
- obtener un sistema en donde los ensayos puedan realizarse de manera ágil, aplicando software de comunicación que resulte transparente al usuario, y
- proveer un sistema independiente de adquisición de las trayectorias efectivamente realizadas por el manipulador.

Basado en los objetivos del sistema, se adoptó un diseño de software que divide funcionalidades de forma tal que

- el controlador del robot concentra todo el software para realización de movimientos y adquisición simultánea de datos de trayectoria realizada, sirviendo de esclavo de la red de PCs, y
- las PCs actúan como maestro del control del movimiento, son la interfaz de usuario sobre la que se programan los algoritmos de resolución de redundancia y sirven de almacenamiento de datos de entrada y salida del sistema.

Dado que el vínculo físico entre ambos subsistemas se realiza mediante comunicación Ethernet convencional y no industrial, no hay garantías en cuanto a retardos de comunicación de datos. Para superar esa limitación, se ha programado un mecanismo de software que evita la pérdida de datos adquiridos, aún mientras prioriza el envío de objetivos de movimiento hacia el controlador del robot en cada intervalo de cómputo. La figura 1 muestra un diagrama simplificado de la arquitectura completa del sistema.

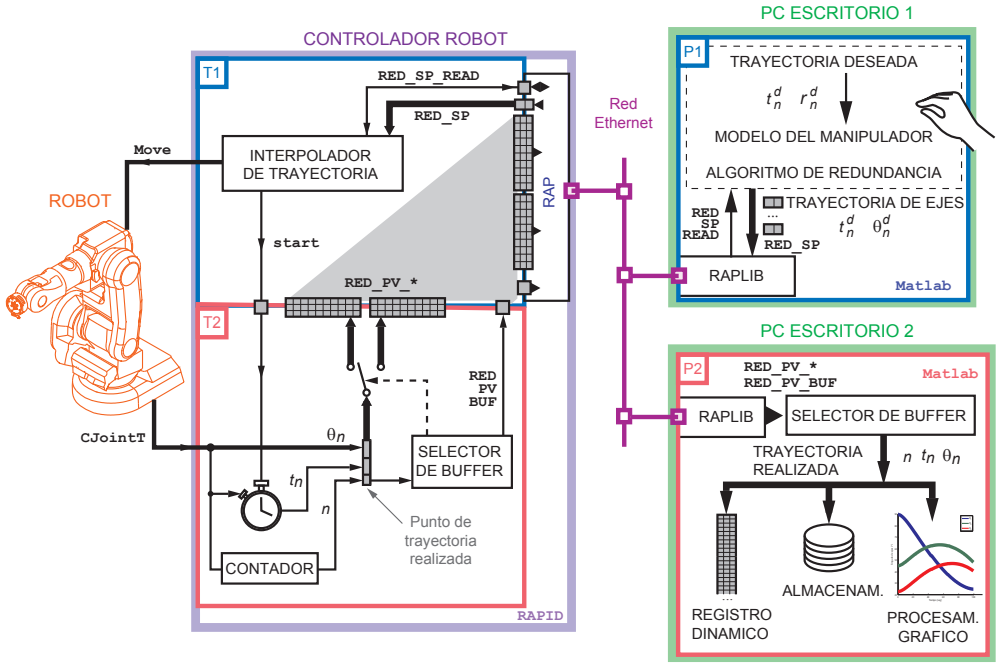

Fig. 1. Arquitectura de hardware y software del sistema de ensayos algoritmos de resolución de trayectoria con HIL.

A. *Software en el controlador del robot*

El software programado en el controlador del robot lleva a cabo dos operaciones principales de manera simultánea:
- Mueve el *TCP* buscando seguir la trayectoria de ejes deseada $\boldsymbol{\theta}^d(t)$.
- Adquiere muestras de la trayectoria de ejes realizada por el TCP $\boldsymbol{\theta}(t)$ y las comunica en línea hacia la interfaz de usuario.

Empleando la opción multitarea del controlador y eligiendo tiempos de muestreo mayores de $90ms$ pueden obtenerse movimientos suaves de los ejes, tal lo analizado en [10].

Del análisis de las operaciones requeridas surge manera natural una división de tareas en:
T1: movimientos en trayectoria de ejes deseada o *subsistema de navegación.*
T2: toma de muestras de trayectoria de ejes realizada o *subsistema de adquisición.*

A.1 Subsistema de Navegación (T1).

El robot recibe la trayectoria de ejes deseada como una secuencia de *puntos de trayectoria deseada*

$$..., [t_k^d, \boldsymbol{\theta}_k^d], [t_{k+1}^d, \boldsymbol{\theta}_{k+1}^d], ...$$

siendo t_k^d el tiempo en el que se desea que los ejes del robot se encuentren en posición $\boldsymbol{\theta}_k^d$. El sistema de navegación *traduce* los puntos de trayectoria deseada convirtiéndolos en comandos de movimiento para el robot.

La implementación de T1 es básicamente un lazo que:
1. lee un punto de trayectoria deseada (variable `RED_SP`)
2. ejecuta un movimiento hacia ese punto.

El control de recepción de los puntos de trayectoria deseada se realiza en línea mediante *flags de control de entrada* (`RED_SP_READ`). El susbsistema de navegación interpola cada uno de los puntos recibidos de forma de lograr una trayectoria se movimientos suave.

A.2 Subsistema de Adquisición (T2).

El subsistema de adquisición permanece en espera hasta que el usuario ordena el comienzo del ensayo. Cuando T1 detecta esa orden al recibir el primer punto de la trayectoria deseada, le indica dicho evento a T2 (usando el flag `start`) y T2 sale del modo de espera. El subsistema de adquisición comienza entonces a obtener muestras de la trayectoria efectivamente realizada por el robot. Cada *punto de trayectoria realizada* contiene
- un número identificador único y consecutivo n que facilita la detección alguna eventual pérdida de datos en la comunicación,
- el valor t_n de un reloj interno, y
- el valor de posición de ejes instantáneo $\boldsymbol{\theta}_n$

quedando la trayectoria realizada caracterizada por

$$..., [n, t_n, \boldsymbol{\theta}_n], [n+1, t_{n+1}, \boldsymbol{\theta}_{n+1}], ...$$

Este conjunto de datos es la salida de T2, y se encuentra disponible para ser leído desde la red de PCs mediante el vínculo establecido con T1. Dicha comunicación se realiza con un sistema de doble buffer que permite reducir la carga sobre el canal de comunicación Ethernet y así priorizar la recepción de puntos de trayectoria deseada.

B. *Software en la red de PCs*

En las PCs se eligió utilizar Matlab como software base de cómputo de trayectoria, principalmente por la simplicidad de realizar cálculos y programar algoritmos de tiempo discreto. Además se emplea el paquete RAPLIB [11] que es un desarrollo del Laboratorio para comunicación Matlab-RAP, cuya robustez ya fue validada en trabajos previos. El software adicional programado para abordar problemas de redundancia lleva a cabo dos operaciones principales de manera independiente:

P1: Resolución en línea del algoritmo de redundancia elegido y envío de objetivos de posicionamiento en cada período de muestreo al robot. *Subsistema Solver.*
P2: Lectura, visualización en línea y almacenamiento de secuencia de trayectoria realizada. *Subsistema de Almacenamiento.*
Estas operaciones pueden ser resueltas en múltiples tareas en una o más PCs, y constituyen un sistema de ensayos distribuido sobre la red de comunicación.

B.1 Subsistema Solver (P1) - Planteo en tiempo discreto.

Para implementar en un controlador industrial los algoritmos descriptos en la sección previa es indispensable plantearlos en tiempo discreto. Para eso se adoptó un tiempo de muestreo T_s que implica la secuencia $t_k = k.T_s$. La ecuación (2) expresada en tiempo discreto permite obtener los objetivos de trayectoria de los ejes por ejemplo como

$$\begin{aligned}\boldsymbol{\theta}_k^d = &\boldsymbol{\theta}_{k-1} + \boldsymbol{J}(\boldsymbol{\theta}_{k-1})^+[\boldsymbol{r}_k^d - \boldsymbol{r}(\boldsymbol{\theta}_{k-1})]+ \\ &[\boldsymbol{I} - \boldsymbol{J}(\boldsymbol{\theta}_{k-1})^+\boldsymbol{J}(\boldsymbol{\theta}_{k-1})]\boldsymbol{v}_k\end{aligned} \tag{9}$$

donde $\boldsymbol{v}_k$ se calcula dependiendo del criterio secundario que se aplique. Esta ecuación se resuelve en cada T_s, partiendo del modelo matemático de la cinemática del manipulador: su problema directo ($\boldsymbol{r}$) y jacobiano ($\boldsymbol{J}$) que se evalúan en línea, y el estado anterior de ejes ($\boldsymbol{\theta}_{k-1}$).

El paquete de software implementado permite intercambiar módulos de: (i) modelo del manipulador y (ii) algoritmo de resolución de redundancia. Ambos módulos fueron implementados como funciones Matlab y como parte de un simulador del manipulador. Esta estructura es fácilmente adaptable y permite realizar ensayos de manera ágil.

B.2 Subsistema de Almacenamiento (P2).

La PC dedicada a la adquisición implementa un paquete de software Matlab que accede a la memoria del robot, y alternativamente lee cada uno de los dos buffers que se actualizan y validan en línea gracias a T2. De esta forma se obtienen en línea muestras de la trayectoria realizada, y se la almacena en la PC como un registro dinámico. Al finalizar el movimiento estos datos son útiles para análisis fuera de línea.

IV. Ensayos

Para los primeros ensayos se eligió simplificar el modelo del manipulador IRB140 a un robot de tres ejes que actúa sobre un plano. De esta forma pudo verificarse la funcionalidad de la plataforma, y aplicar los criterios de resolución de redundancia presentados en la sección anterior. Actualmente se está trabajando con el modelo completo, que implica mayor carga de procesamiento, pero los conceptos que se presentan a continuación para el modelo simplificado -en cuanto a características, ventajas y desventajas de cada uno de los criterios- también se manifiestan.

Modelo. Para que el manipulador IRB140 pueda actuar como redundante, debe realizar una tarea que requiera menos de 6 grados de libertad. Por ejemplo, mover el TCP en un plano pero sin exigirle mantener una orientación determinada. En este caso la tarea corresponde a un espacio

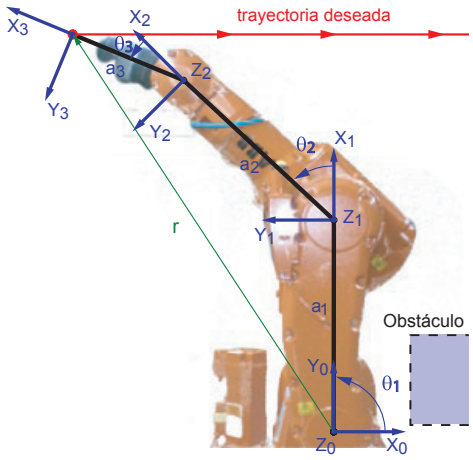

Fig. 2. Las ternas elegidas, la trayectoria deseada y el obstáculo ficticio a evitar, ilustrados sobre el manipulador y su modelo.

reducido de 2 dimensiones, por lo que un manipulador con 3 ejes de rotación normales a ese plano puede actuar como redundante. El plano elegido es el definido por el primer eje y el centro de la muñeca del manipulador de la celda de ensayos. A ese plano se le asignaron las coordenadas X_0Y_0, donde $\boldsymbol{X_0}$ es un versor que está sobre la línea normal a los dos primeros ejes del robot, con sentido del 1 al 2 y el versor $\boldsymbol{Y_0}$ está en el origen del 2, con sentido alejándose de la base. De esta manera los ejes 2, 3 y 5, activos durante la tarea con redundancia del manipulador, son las tres articulaciones del modelo de cálculo. Por otro lado los ejes 1, 4, y 6 no intervienen en el movimiento. Las longitudes de cada eslabón del modelo son los valores reales del manipulador, $a_1 = 360mm$, $a_2 = 380mm$ y $a_3 = 215mm$. Las ternas 1, 2, y 3 del modelo están ubicadas solidarias a los eslabones accionados por los ejes activos del manipulador, tal como se indica en la Fig.2. El modelo del manipulador resultante implica que el vector posición del TCP en la terna X_0Y_0 resulta

$$\boldsymbol{r} = \begin{bmatrix} a_1C_1 + a_2C_{12} + a_3C_{123} \\ a_1S_1 + a_2S_{12} + a_3S_{123} \end{bmatrix} \tag{10}$$

donde se empleó la notación genérica $C_{fgh} = cos(\theta_f + \theta_g + \theta_h)$ y $S_{fgh} = sen(\theta_f + \theta_g + \theta_h)$ con $\boldsymbol{\theta}_k = [\theta_1, \theta_2, \theta_3]_k^T$. Nótese que puede establecerse una relación unívoca entre los ángulos θ_1, θ_2, θ_3 del modelo y los ángulos reales de los ejes 2, 3, 5 del manipulador respectivamente. El jacobiano que permite calcular la velocidad de traslación del TCP para el modelo, proyectado en X_0Y_0 es

$$\boldsymbol{J} = \begin{bmatrix} -a_1S_1 - a_2S_{12} - a_3S_{123} & a_1C_1 + a_2C_{12} + a_3C_{123} \\ -a_2S_{12} - a_3S_{123} & a_2C_{12} + a_3C_{123} \\ -a_3S_{123} & a_3C_{123} \end{bmatrix}^T \tag{11}$$

Tarea principal. Consiste en seguir una trayectoria rectilínea en dirección de $\boldsymbol{X_0}$, a velocidad constante, con $T_s = 0.1seg$ en $100seg$. Se parte desde una posición definida por los ángulos

$$\boldsymbol{\theta}_0 = [90° \;\; 45° \;\; 22.5°]^T$$

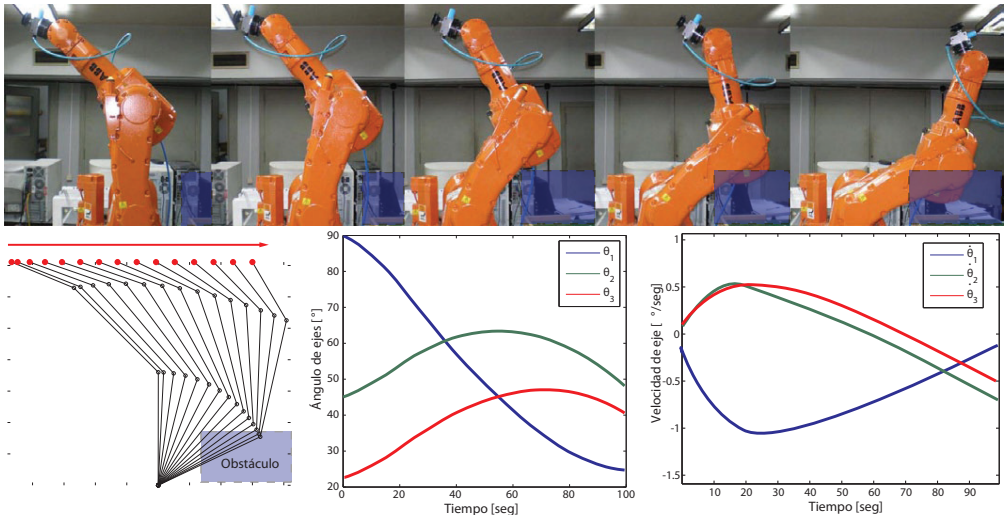

Fig. 3. Criterio minimización de velocidad de ejes. Resultados.

que corresponde a una posición en X_0Y_0

$$\boldsymbol{r}_0^d = [-467.3347mm \;\; 710.9775mm]^T$$

hasta posición final que tiene el mismo valor en Y_0 y es simétrica en X_0 respecto del eje 1 del modelo.

Tarea secundaria. Se plantea que el manipulador, siempre que el algoritmo elegido posea la capacidad de hacerlo, evite chocar contra un obstáculo ficticio dispuesto como se muestra en la Fig.2. Se presenta a continuación la aplicación de los tres criterios secundarios planteados en la Sec.2 aplicados al ensayo mencionado.

A. Criterio: Minimización de velocidad de ejes

En este caso no es posible realizar ninguna acción de configuración del algoritmo de resolución de trayectoria para evitar el obstáculo. La figura 3 muestra los resultados del ensayo de este algoritmo sobre el manipulador y los datos adquiridos. Pueden verse las velocidades de cada uno de los ejes, que en este caso son las mínimas posibles para cumplir con la tarea principal. Además, puede notarse que el robot chocaría contra el obstáculo planteado.

B. Criterio: Objetivo de posición de ejes

En este caso se ha configurado el algoritmo descripto por (5) eligiendo

$$\boldsymbol{\theta}_{2e} = [90° \quad -90° \quad 22.5°]^T$$

y

$$\boldsymbol{H} = 0.01 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Obsérvese que $\boldsymbol{\theta}_{2e}$ no tiene que ser necesariamente una posición válida para la trayectoria, y se ha elegido de tal manera de forzar al eje 2 a 'quebrarse' durante el movimiento. El factor 0.01 que se incluye exprofeso como parte de $\boldsymbol{H}$ resulta útil para la configuración del algoritmo. Permite ajustar el 'peso' o importancia relativa de la subtarea frente a la que naturalmente optimiza el primer factor de (2) (minimización de $||\dot{\boldsymbol{\theta}}||$). Pero también se evidencia el compromiso entre la magnitud de este factor y la velocidad objetivo resultante de los ejes. Por lo tanto, es importante ajustarlo para evitar que las velocidades de los ejes superen el máximo permitido por el controlador del manipulador. Su selección en la aplicación se hizo a partir de ensayos sobre el simulador. La figura 4 muestra los resultados del ensayo de este algoritmo sobre el manipulador y los datos adquiridos. Pueden verse las velocidades de cada uno de los ejes, mayores que en el caso anterior. Pero ahora, como puede observarse, el manipulador no choca contra el obstáculo planteado.

C. Criterio: Movimiento de punto obstáculo

Para completar la tarea con este criterio se elige como punto obstáculo el punto que está en el origen de la terna 1, al que se le asigna una velocidad que lo aleja del obstáculo a evitar. El jacobiano que permite calcular la velocidad del punto obstáculo es

$$\boldsymbol{J}_{2a} = \begin{bmatrix} -a_1 S_1 & 0 & 0 \\ a_1 C_1 & 0 & 0 \end{bmatrix} \tag{12}$$

y la velocidad objetivo adoptada

$$\dot{\boldsymbol{r}}_{2a} = 10[-1mm/seg \;\; 0.5mm/seg]^T$$

Obsérvese que la dirección de esa velocidad no es (necesariamente) realizable para el punto obstáculo. También aquí se elige un factor, que en este caso es 10, para ajustar la 'importancia' de la tarea secundaria. Esa elección se obtuvo de ensayos sobre el simulador. Este factor evidencia el compromiso entre cumplir la velocidad pedida al punto obstáculo y exigir mayores velocidades a los ejes mientras se cumple la tarea principal. Si bien este criterio implica un mayor costo computacional que los anteriores, puede ser resuelto en el tiempo de muestreo adoptado, en una PC convencional con Matlab. La figura 5 muestra los resultados del ensayo de este algoritmo sobre el manipulador y los datos adquiridos. Pueden verse las velocidades de cada uno de los ejes, mayores también que el primer

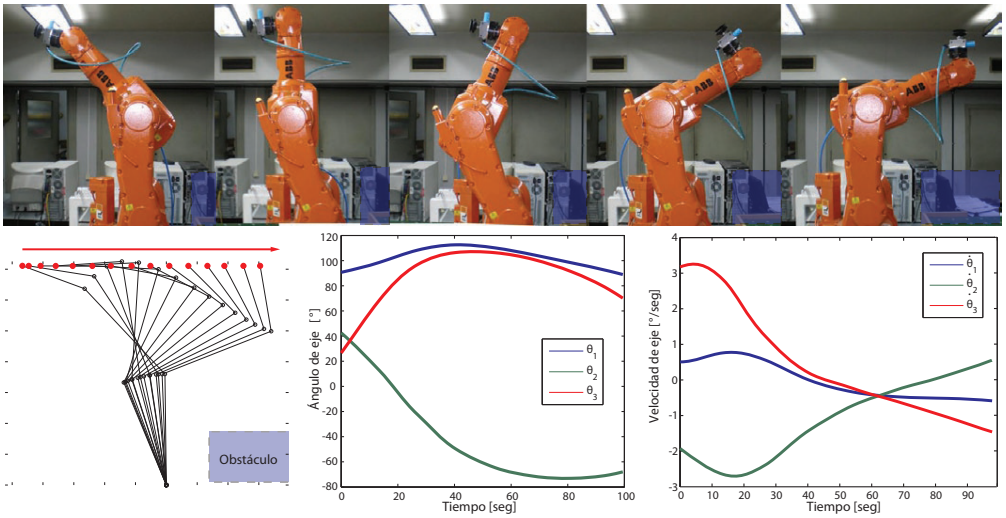

Fig. 4. Criterio Objetivo de posición de ejes. Resultados.

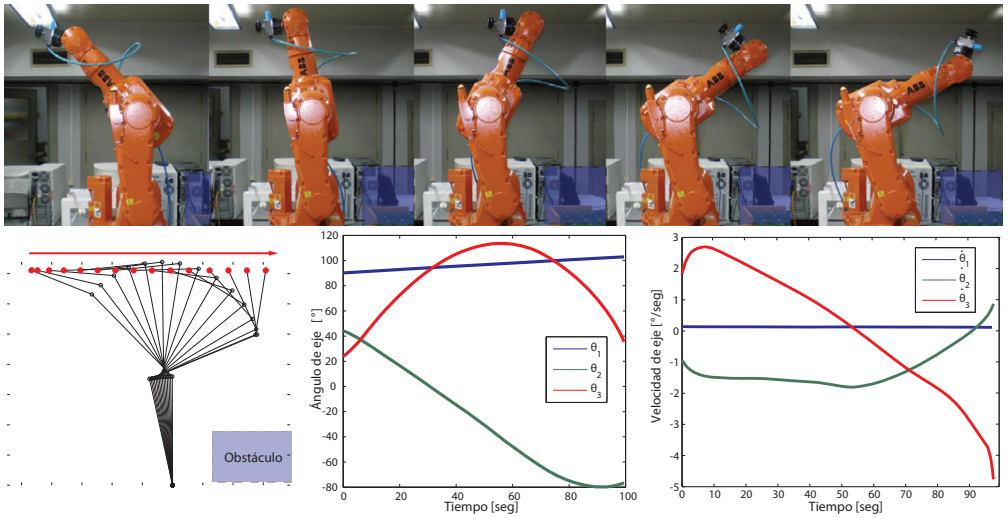

Fig. 5. Criterio Movimiento de punto obstáculo. Resultados.

caso. Pero, tal como en el segundo caso, se observa que el manipulador no choca contra el obstáculo planteado.

V. Conclusiones

Se creó una plataforma para ensayar algoritmos de resolución de redundancia cinemática sobre un manipulador industrial real. La arquitectura planteada resultó útil para el objetivo buscado, y fue posible ensayar diferentes criterios y modelos sin necesidad de cambios en la codificación de bajo nivel.

Se mostró sobre un ejemplo de aplicación la dificultad de selección de parámetros de configuración de cada uno de los algoritmos y criterios explicados, y la ventaja de recurrir a simulaciones para ajustarlos.

Actualmente se continúa trabajando con la misma plataforma, realizando otras tareas y utilizando el modelo completo del manipulador.

Referencias

[1] S. Chiaverini, "Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators," *IEEE Trans. on Robot. and Autom.*, vol. 13, pp. 398–410, 1997.

[2] T. Yoshikawa, *Foundations of robotics: analysis and control*. Cambridge, Massachusetts: The MIT Press, 1990.

[3] B. Siciliano and J. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," *Proc. Int. Conf. Adv.*, vol. 2, pp. 1211–1216, 1991.

[4] N. Mansard and F. Chaumette, "Task sequencing for high-level sensor-based control," *IEEE Trasnsactions on robotics and automation*, vol. 23, pp. 60–72, 2007.

[5] D. Whitney, "The mathematics of coordinated control of prosthetic arms and manipulators," *Trans. ASME Journal of Dyn. Systems, Measurement, and Control*, vol. 94, pp. 303–309, 1972.

[6] R. P. Paul, *Robot Manipulators: Mathematics, Programming and Control*. Massachusetts, USA: The MIT Press, 1981.

[7] D. Whitney, "Resolved motion rate control of manipulators and human prostheses," *IEEE Transactions Man, Machine Systems*, vol. 10, pp. 47–53, 1969.

[8] A. Maciejewski and A. Klein, "Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments," *The international Journal of Robotic Research*, vol. 4, pp. 109–117, 1985.

[9] H. Jurado and M. Anigstein, "Plataforma externa generadora de trayectorias para un robot industrial," in *AADECA 2012*, Bs. As., Octubre 2012.

[10] C. Cartelli, P. González, J. Giribet, and M. Anigstein, "Plataforma robótica de ensayos para sensores de actitud," in *RPIC 2013*, San Carlos de Bariloche, Octubre 2013.

[11] P. Ilardi, M. D'Ascanio, and M. Rouaux, *RAP Toolbox para MATLAB. Manual del Usuario*. Buenos Aires: FIUBA, 2003.