HERRAMIENTA DE SIMULACIÓN DE UNA CÉLULA ROBOTIZADA

J. F. Sarabia*, F. Rodríguez* , L. Iribarne+, M. Berenguel*
sarabia@larural.es, frrodrig@ual.es, liribarn@ual.es, beren@ual.es
*Área de Ingeniería de Sistemas y Automática, + Área de Lenguajes y Sistemas Informáticos.
Dpto. Lenguajes y Computación, Universidad de Almería, Carretera de la Playa, s/n, E-04120, Almería

Resumen

En este trabajo se describe una herramienta software que permite la programación y la simulación de una célula robotizada basada en robots manipuladores tipo Scorbot ER V-Plus junto con los periféricos más utilizados en este tipo de sistemas. Al igual que ocurre en el robot real, la programación del entorno se puede llevar a cabo en los lenguajes Scorbase y ACL. Una vez simulados los movimientos en el computador y aceptado el programa que los controla, dicho programa se puede transmitir al controlador del robot para poder apreciar el comportamiento del robot real.

Palabras Clave: Célula robotizada, simulación.

1 INTRODUCCIÓN

El continuo aumento en el nivel de automatización de los procesos de producción en entornos industriales hace necesario dotar a los centros docentes de laboratorios de prácticas con un equipamiento adecuado, que permita que la formación de los alumnos sea acorde a la evolución de la automatización en ambientes industriales. La finalidad principal de este tipo de laboratorios es la formación del alumno en sistemas y equipos similares a los que se encontrará en entornos reales. El equipamiento requerido en estos laboratorios suele tener un coste muy elevado, lo que provoca que en la mayoría de los casos exista la necesidad de adoptar soluciones que involucren tanto experiencias en simulación como en plantas reales. Una solución habitualmente implantada consiste en disponer de un número reducido de sistemas, cuyo tiempo de utilización se reparte entre los grupos de prácticas, que previamente han realizado experiencias en una versión simulada de ese sistema. Además, estas experiencias son muy útiles para depurar la aplicación y prevenir las posibles averías que suelen producirse cuando el alumno trabaja directamente con el sistema real sin haber simulado previamente las tareas que se quieren realizar.

En el ámbito de células robotizadas basadas en robots manipuladores, el uso de la simulación permite que el alumno se familiarice con el robot, sus movimientos, su programación y la relación con su entorno de trabajo.

Este trabajo presenta las características de una herramienta software que permite la programación y la simulación de células robotizadas basadas en el robot manipulador Scorbot ER V-PLUS, junto con los periféricos más utilizados en este tipo de sistemas. En la sección 2 se muestra la estructura de una célula robotizada real. Seguidamente, en la sección 3 se describe el desarrollo de la herramienta propuesta. En la sección 4 se hace una descripción detallada de la aplicación informática y finalmente, en la sección 5, se indican las conclusiones y los futuros trabajos orientados a la mejora de la herramienta de simulación de forma que se incluyan utilidades de telecontrol.

2 DESCRIPCIÓN DE LA CÉLULA ROBOTIZADA REAL

En la figura 1 se muestra un esquema de la configuración de la célula robotizada, que está constituida por los siguientes elementos:

- Brazo mecánico. Es el manipulador robótico Scorbot ER V-Plus de Eshed Robotics, articulado con 5 grados de libertad.
- Controlador multitarea en tiempo real, permite la ejecución simultánea de varios programas. Sus principales características se indican en la tabla 1

E/S	16 entradas (TTL, 12v ó 24v) 16 salidas (4 relé/12 colector abierto)
Comunicación	RS-232C
Programación	ACL y SCORBASE (Nivel 5)
CPU	Motorola 68010
Sistema coordenadas	X, Y, Z y articulaciones del robot Definiciones absolutas y relativas
Otras	Autónomo PID, en tiempo real, multitarea, PWM

Tabla 1: Características técnicas del controlador

- Botonera de enseñanza, que permite el control de las articulaciones, la definición de posiciones por medio del método de enseñanza y la ejecución de programas que se encuentren en el controlador.

- Cinta transportadora accionada por un motor AC controlado a través de un variador de velocidad conectado al controlador del robot, de modo que se puede controlar desde el mismo el movimiento (puesta en marcha, parada y velocidad) de la cinta transportadora.
- Sensores fotoeléctricos, que permiten conocer la posición en que están ubicados los objetos en la cinta transportadora.
- Sistema de visión, formado por una cámara CCD color y una tarjeta capturadora con *frame grabber* instalada en el computador de control.
- Computador con procesador Pentium 166 Mhz, conectado al controlador del robot vía RS-232.

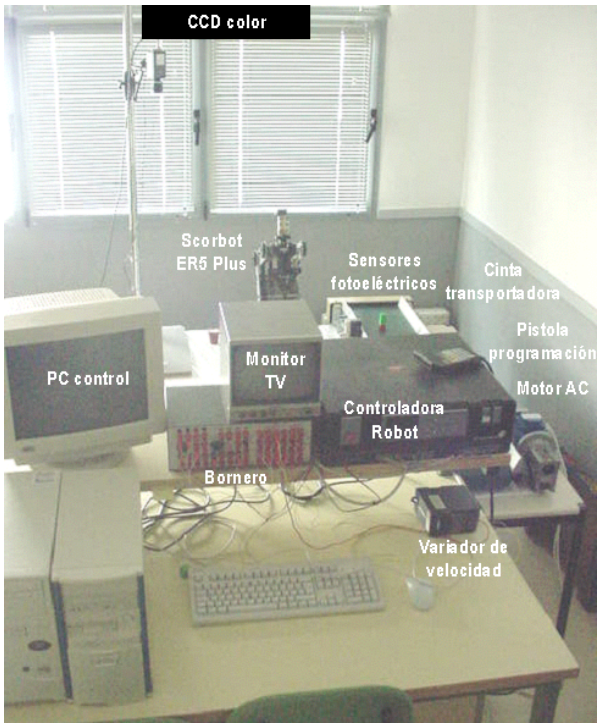

Figura 1: Configuración de la célula robotizada

3 DESCRIPCIÓN DEL MODELO DE LA APLICACIÓN

El desarrollo de la aplicación se asemeja a la construcción real de una célula robotizada. Lo primero que hay que modelar es el brazo robot, tarea que incluye la construcción del modelo gráfico del manipulador, la simulación de su movimiento respecto a un sistema de referencia fijo (cinemática) y la generación de trayectorias que debe seguir para alcanzar un objetivo determinado por el usuario. A continuación habrá que modelar el entorno de trabajo del brazo robot y sus interacciones con los distintos periféricos.

3.1 MODELO CINEMATICO DEL ROBOT

La cinemática de un brazo robot estudia la geometría del movimiento del mismo con respecto a un sistema de referencia fijo sin tener en cuenta las fuerzas que los originan. Relaciona la posición y orientación del extremo final del robot con permitiendo la descripción analítica del movimiento espacial del robot como una función del tiempo, y en particular por las relaciones entre la posición y la orientación del extremo final del robot con los valores que toman sus coordenadas articulares [1].[5][10]. En el estudio de la cinemática se plantean dos problemas:

- Determinación de la posición y orientación del elemento terminal a partir de las coordenadas articulares del robot (Cinemática directa)
- Determinación de las coordenadas articulares a partir de la posición y orientación del elemento terminal (Cinemática inversa)

Existen distintos métodos para resolver estos problemas. En el que se ha utilizado para el desarrollo de la aplicación se basa en la representación de Denavit-Hartenberg (D-H), que sitúa de forma sistemática un sistema de coordenadas en cada articulación del robot, de forma que se pueda relacionar el sistema S_i de la articulación i con el sistema de la articulación anterior, S_{i-1}, utilizando una matriz homogénea generada por el producto de cuatro transformaciones básicas: rotación alrededor del eje z_{i-1} un determinado ángulo θ_i, traslación a lo largo del eje z_{i-1} una distancia d_i, traslación a lo largo del x_i una distancia a_i y rotación alrededor del eje x_i un ángulo α_i. La posición del elemento terminal vendrá determinada por el producto de matrices homogéneas de transformación que dependerán del valor de cada una de las coordenadas articulares del robot, así como de sus características físicas (longitud, tipo de articulaciones, etc.).

La figura 2 muestra la situación de los sistemas de coordenadas siguiendo la representación de D-H para el robot manipulador Scorbot ER–V plus y en la tabla 2 se muestran los parámetros de D-H de cada articulación del robot. Sustituyendo estos parámetros se obtienen 5 matrices homogéneas que permiten el cálculo de la posición de la pinza del robot a partir de los ángulos de cada una de sus articulaciones.

	θ_I	d_i	a_i	α_i
Articulacion 1	θ_1	L_1	L_2	90
Articulacion 2	θ_2	0	L_3	0
Articulacion 3	θ_3	0	L_4	0
Articulacion 4	$\theta_4 + 90$	0	0	90
Articulacion 5	θ_5	L_5	0	0

Tabla 2: Parámetros D-H de Scorbot ER-V Plus.

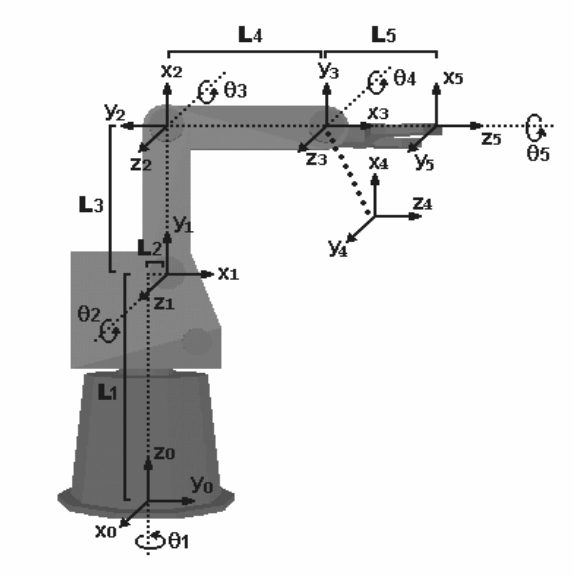

Figura 2: Representación D-H del Scorbot.

Para el estudio de la cinemática inversa se utiliza la técnica denominada desacoplo cinemático [5], que consiste en dividir el problema en dos más simples:

- Problema cinemático inverso de la posición, en el que obtienen los valores de las primeras tres articulaciones que posicionan al robot
- Problema cinemático inverso de orientación, en el que se obtienen los valores de las dos últimas articulaciones que orientan al extremo del robot.

3.2 GENERACIÓN DE TRAYECTORIAS

El generador de trayectorias se ocupa de las posiciones que debe seguir cada articulación a lo largo del tiempo para lograr los objetivos fijados por el usuario (punto de destino, tipo de trayectoria, velocidad, etc.). El generador implementado en la aplicación que se describe en este artículo incorpora dos tipos de trayectorias:

- Trayectorias punto a punto, en la que cada articulación evoluciona desde un punto inicial a un punto final sin realizar consideración alguna sobre el estado o evolución de las demás articulaciones [1].
- Trayectoria continua definida por el usuario que debe seguir el extremo del robot desde el punto inicial al final, por ejemplo, lineal.

Los algoritmos de generación de trayectorias aproximan el camino a seguir mediante funciones polinomiales. En el caso de la aplicación desarrollada se han utilizado interpoladores cúbicos, que aseguran que la trayectoria que une los puntos por los que tiene que pasar cada articulación presente continuidad en velocidad [10]. Por tanto, la trayectoria de la articulación *i* estará definida por el polinomio:

$$q_i = a_o + a_1 t + a_2 t^2 + a_3 t^3 \qquad (1)$$

De este modo, al tener cuatro parámetros disponibles se podrán imponer cuatro condiciones de contorno:

- Condiciones iniciales de posición, q_{i0}, y velocidad, q_{i0}'

$$q_{i0}(t_f) = q_{i0} \qquad q_{i0}(t_f)' = q_{i0}' \qquad (2)$$

- Condiciones finales de posición, q_{if}, y velocidad, q_{if}'

$$q_{if}(t_f) = q_{if} \qquad q_{if}(t_f)' = q_{if}' \qquad (3)$$

Para calcular la velocidad deseada de una articulación, sólo hay que derivar la ecuación de la trayectoria articular:

$$q_i' = a_1 + a_2 t + a_3 t^2 \qquad (4)$$

Combinando las ecuaciones y las condiciones de contorno se obtienen las cuatro ecuaciones siguientes

$$\begin{aligned} q_{i0} &= a_o + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3 \\ q_{i0}' &= a_1 + 2 a_2 t_0 + 3 a_3 t_0^2 \\ q_{if} &= a_o + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 \\ q_{if}' &= a_1 + 2 a_2 t_f + 3 a_3 t_f^2 \end{aligned} \qquad (5)$$

donde los parámetros a_0, a_1, a_2 y a_3 definirán la función que describe la trayectoria que debe seguir la articulación a lo largo del tiempo.

Las trayectorias continuas que se han implementado en esta aplicación son la trayectoria lineal y la circular. Para implementar la trayectoria lineal, conocido el punto inicial y el final, se calcula la ecuación de la recta que los une. A continuación se obtiene un vector de posiciones muestreando la ecuación de la recta obtenida. Para conocer las posiciones articulares en cada instante, sólo hay que aplicar las ecuaciones obtenidas en el estudio cinemático inverso. La trayectoria que se debe seguir entre los puntos muestreados será un polinomio de tercer orden. El resultado final será una trayectoria cuasi-lineal. El mismo procedimiento se utiliza para la trayectoria circular, sólo que la curva a calcular necesita un punto intermedio entre el inicial y el final.

3.3 MODELADO DEL ENTORNO

Los objetos generales que intervienen en el entorno de simulación por computadora han sido modelados haciendo uso de objetos primitivos de la clase triángulo, permitiendo representar objetos complejos como el robot manipulador hasta los más simples como los cubos que se utilizan como *testbeds* de prueba en las fases de simulación [6]. La utilización de triángulos para construir cada uno de los objetos hace que la ejecución de las simulaciones y el trabajo con el programa en general sea más fluido, aprovechando además el hardware gráfico de última generación que suele estar optimizado para el dibujo de triángulos [7], [8].

Para la modelización del ambiente descrito del robot manipulador se han considerado dos clases de objetos 3D:

- **Objetos de composición (OC).** Los objetos OC son aquellos objetos de la escena tridimensional que componen y dan forma al brazo de robot en tiempo real. El modelo 3D del robot manipulador se compone de una serie de objetos primitivos que se corresponden con las articulaciones del robot. Las dimensiones de estos objetos se han obtenido de la información que proporciona el fabricante del manipulador [2], permitiendo con esto que el modelo 3D tenga el mismo volumen de trabajo que el robot real. En la implementación de los objetos de composición (OC) del robot, cada articulación se dibuja por separado. Cada articulación tiene asociada una posición y orientación en el espacio tridimensional que ésta ocupa, respetando las características de movimiento y umbrales de rotación establecidos por el fabricante del robot manipulador [2]. Siguiendo la tendencia tradicional para la programación de objetos en tres dimensiones animados, se ha hecho uso una jerarquía de objetos para dar forma y comportamiento al robot manipulador. Cada objeto de la jerarquía hereda las características de movimiento y de rotación de los objetos ancestros. Por tanto, para obtener sensación de animación por computadora, a cada articulación se le asocia una matriz de transformación, resultante de la composición las transformaciones geométricas de las articulaciones ancestros más la transformación que corresponde a su ángulo de giro. Cada una de estas transformaciones vendrán representadas por matrices homogéneas como las comentadas en la sección 3.1 (cinemática directa).
- **Objetos de simulación (OS).** Los objetos OS son objetos básicos que pueden intervenir en la simulación e interactúan directa o indirectamente con el brazo de robot. A su vez, estos objetos han sido catalogados de dos tipos:
  - *Testbeds* 3D: son sólidos que se suelen utilizar para operaciones de *Pick & Place* en las operaciones de simulación por ordenador. Estos objetos también pueden utilizar para simular obstáculos, mesas, cajas, y otros.
  - Periféricos: son objetos que interactúan con el controlador por medio de sus E/S. Pueden ser sensores, cintas transportadoras, luces o zumbadores.

Para la modelización 3D del sistema de simulación comentado, tanto del entorno como del propio manipulador, se respetan las condiciones de animación reales; es decir, para el movimiento del robot dentro de su volumen de trabajo se han tenido en consideración las posibles colisiones con los objetos, con el suelo o con su propia estructura, en algunos casos respetando leyes físicas de los objetos que intervienen en la escena (gravedad de los objetos, fuerza de las pinzas, velocidad de desplazamiento, entre otros). Por ejemplo, antes del desplazamiento de cada articulación del brazo se comprueba si la posición en la que va a quedar el brazo implica el choque con algún objeto del entorno, si es así no se permite este movimiento. De esta forma, se evitan desperfectos ocasionados por una inadecuada ubicación del robot o por una errónea programación, indicándoselo al usuario de la herramienta.

Para la generación del entorno de simulación 3D se ha utilizando la API gráfica *OpenGL* [11]. Se ha seleccionado esta librería gráfica por ser una de las más utilizadas en diseño gráfico y por su facilidad y variedad de órdenes y funciones para el dibujo de primitivas gráficas como puntos, líneas y polígonos en tres dimensiones.

3.4 COMUNICACIÓN CON EL CONTROLADOR

La comunicación entre la aplicación y el controlador del robot se realizan mediante una conexión RS-232. El controlador trabaja internamente con el lenguaje de programación ACL [3], por lo que la aplicación debe enviar la orden al puerto serie con la sintaxis ACL. El controlador la interpreta y la ejecuta. Si la ejecución se ha realizado correctamente, envía a la aplicación la cadena de caracteres *"Done."*. En caso contrario, devuelve una cadena con la descripción del error que ha ocurrido (Figura 3). Para gestionar las entradas y salidas de las que dispone la célula, el controlador mantiene dos variables globales denominadas `IN` y `OUT`, que son vectores booleanos de 16 elementos, que representan las 16 entradas (vector `IN`) y las 16 salidas (vector `OUT`). Para trabajar con ellas sólo hay que leer el vector de entradas, asignándoselo a una variable del programa, y escribir en el de salida

Figura 3: Modelo de comunicaciones.

El fabricante del manipulador ofrece la posibilidad de programar sistema utilizando el lenguaje SCORBASE [4] que, aunque no es tan versátil como ACL, es recomendable como herramienta de aprendizaje e inicio. Las instrucciones Scorbase no se pueden enviar directamente al controlador, sino que

deben ser traducidas a ACL. Para ello se han implementado un conjunto de funciones en lenguaje C cuyos parámetros de entrada son los mismos parámetros de la instrucción y generan la cadena de caracteres equivalente en lenguaje ACL. Por ejemplo, la función correspondiente a la instrucción "`Ir posición`" necesita dos parámetros de entrada: posición a la que debe ir y la velocidad que debe llevar. En ACL equivale a dos instrucciones que se deben ejecutar de forma secuencial: `SPEED` velocidad y `MOVE` posición, tal y como se puede observar en el código de la función implementada:

```
void IrPosicion( int Posicion, int
Velocidad)
{  char Orden[30]="";

 strcpy( Orden,"SPEED " );
 strcpy( Orden, atoi(Velocidad) );
 EnviarPuerto( Orden );

 Orden[0] = '\0';
 strcpy(Orden,"MOVE ");
 strcpy(Orden, atoi(Posicion ));
 EnviarPuerto( Orden ); }
```

4 DESCRIPCIÓN DE LA APLICACIÓN

La aplicación propuesta se ha construido sobre el sistema operativo Microsoft Windows. La herramienta de programación utilizada ha sido Microsoft Visual C++ 6.0, que incluye, herramientas para la realización de programas C++ (orientados a objetos) y un conjunto completo de clases (*Microsoft Foundation Class, MFC*) que permiten crear de forma intuitiva las aplicaciones para Windows y manejar los componentes de Windows según su naturaleza de objetos [9].

4.1 PANTALLA PRINCIPAL

La figura 4 muestra el aspecto de la pantalla principal de la aplicación, diseñada para permitir al usuario realizar las siguientes tareas:

- Configuración de la pantalla de simulación. El usuario puede definir el número de vistas que necesita en esta pantalla. Además, cada una de estas vistas es totalmente configurable, permitiendo modificar los parámetros de la cámara para poder visualizar el entorno desde cualquier punto de vista. Se incluyen unas vistas configuradas por defecto, como la vista cenital, vista izquierda, etc. También se puede cambiar el tipo de sombreado que se va a utilizar en cada una de las vistas (alambre, sólido o con suavizado), lo que permite poder utilizar la aplicación en computadores de bajas prestaciones (usando el sombreado en alambre).

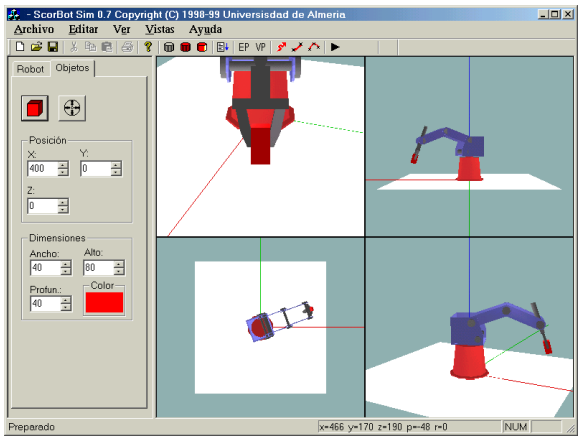

Figura 4: Pantalla principal

- Definición del entorno de simulación. La aplicación permite utilizar entornos personalizados para cada tipo de problema: objetos y periféricos.
- Definición de posiciones del robot. Las posiciones se pueden definir por el método de enseñanza o introduciéndolas directamente desde el teclado. Para mover el manipulador por el entorno se puede utilizar el teclado o ratón.
- Ejecución de programas. Permite la ejecución de los programas paso a paso o ciclo completo.
- Información. Esta pantalla muestra en cada instante varios elementos de información, como la posición del la pinza del robot y el valor de cada una de las coordenadas articulares. También muestra información acerca del estado de cada una de las entradas y salidas del controlador.

4.2 GENERACIÓN DEL ENTORNO

Como se ha descrito anteriormente, se han especificado dos tipos de objetos con los que puede trabajar el simulador: los objetos *testbed* 3D y los periféricos. En el caso de los *testbed* 3D, su configuración es bastante simple, ya que sólo hay que indicar las dimensiones del objeto, su posición y su orientación. Se puede crear este tipo de objetos directamente desde la pantalla principal, habiéndose incluido la posibilidad selección de objetos existentes y cambio de sus propiedades como se muestra en la figura 6.

La configuración de los periféricos no es tan simple, ya que es necesario indicar además de los parámetros anteriores, otros parámetros referentes a la forma del objeto y su conexión con las E/S del controlador. Por ejemplo, la cinta transportadora es el periférico más común en este tipo de entornos, cuya pantalla de configuración se muestra en la figura 5. Se observan cinco conjuntos de parámetros para su configuración:

- *Características físicas*. En esta ficha se confecciona el modelo 3D de la cinta transportadora. El usuario dibuja un polígono que se corresponde con la proyección frontal de la cinta, que junto con la longitud se utilizará para generar el modelo.
- *Posición y orientación*. Se definen tanto la situación como la orientación de la cinta con respecto al eje de referencia fijo del sistema situado en la base del brazo robot.
- *E/S*. Se definen las salidas del controlador que controlan el arranque, parada y velocidad de la cinta. También habrá que completar una tabla de verdad que relacione cada configuración de las salidas con la velocidad de la cinta.
- Sensores. La aplicación también permite la posibilidad de situar sensores en la cinta transportadora, tales como sensores de distancia o fotoeléctricos para detección de objetos.

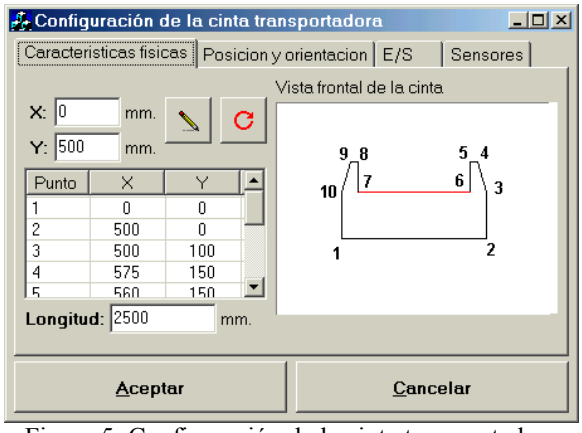

Figura 5: Configuración de la cinta transportadora

4.3 PANTALLA APRENDIZAJE DIRECTO

La programación por aprendizaje directo consiste en determinar las acciones y movimientos del brazo manipulador utilizando un dispositivo de enseñanza diseñado especialmente para este cometido. La célula dispone de una botonera de enseñanza que se ha simulado en una pantalla de la herramienta (Figura 7) de forma que el usuario puede mover las articulaciones del robot simulado (modo *off-line*) como las del robot real (modo *on-line*). Permite el posicionamiento por ejes, es decir, moviéndose cada articulación por separado, o por coordenadas XYZ del extremo del brazo de forma que las articulaciones se adaptan a este movimiento coordinando sus trayectorias. Esta pantalla presenta también la utilidad de poder observar el movimiento del brazo real (modo *on-line*) en el caso de disponer una cámara de video CCD y una tarjeta capturadora de imágenes. Sólo hay que configurar una de las vista de simulación para que muestre la imagen captada por la cámara tal y como se muestra en la figura 8.

Figura 6: Definición de objetos

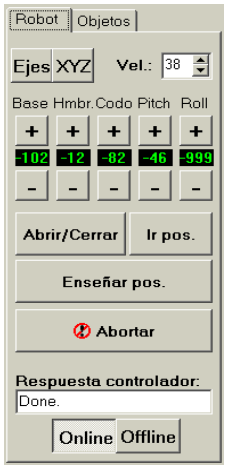

Figura 7: Botonera de enseñanza

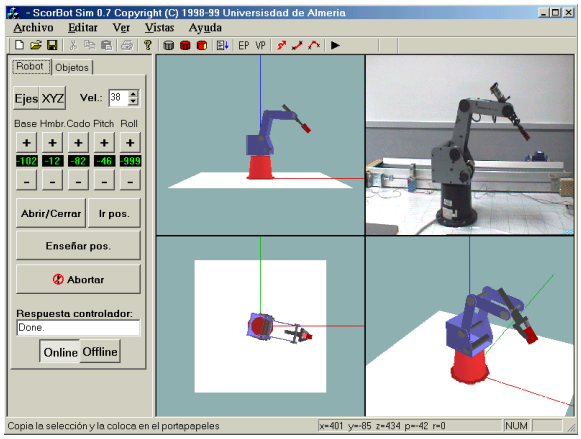

Figura 8: Pantalla principal en modo *on-line*.

4.4 EDICIÓN Y EJECUCIÓN DE PROGRAMAS TEXTUALES

Como se ha comentado en el apartado 3.4, los programas se escriben en ACL o en Scorbase. La programación en ACL se realiza mediante un compilador que comprueba la sintaxis de los programas en busca de errores e interpreta el programa para su simulación. En el caso de Scorbase se ha diseñado una pantalla de programación interactiva, que muestra las posibles instrucciones y permite, por medio de cuadros de diálogo, que el usuario introduzca los parámetros de cada una. Para esto no ha sido necesaria la creación de un interprete, traduciendo directamente las instrucciones a lenguaje ACL. Si se utiliza el lenguaje Scorbase, antes de escribir el programa hay que definir las posiciones. La aplicación presenta dos formas de hacerlo:

- Una pantalla de edición de posiciones donde se introducen los cinco elementos (X,Y,Z, elevación, giro) y un número que la identifique.
- Enseñando las posiciones en la pantalla de aprendizaje directo

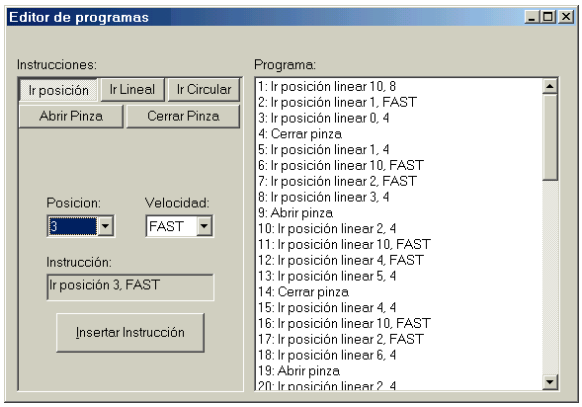

Figura 9: Pantalla de programación en Scorbase.

En ambos modelos de programación, la ejecución de los programas puede hacerse en simulación y sobre el robot real y se permite la ejecución paso a paso, para facilitar la depuración del programa.

5 CONCLUSIONES Y TRABAJOS FUTUROS

Las principales conclusiones que se pueden extraer después de trabajar con la aplicación desarrollada como herramienta de prácticas son:

- Es difícil disponer de un puesto robot para cada grupo de prácticas debido a su elevado coste; por tanto, esta herramienta permite que el alumno desarrolle las prácticas propuestas en un puesto básico de laboratorio para probarlas a continuación en la célula robotizada real.
- El trabajo en una célula robotizada implica el movimiento de objetos en un entorno en el que existen distintos sistemas físicos, por lo que se pueden producir impactos entre ellos debido a una programación incorrecta. Esto supone un deterioro del material e incluso averías en algunos equipos. Con la simulación se evitan estos problemas, ya que no se ejecuta ninguna acción antes de haber comprobado su correcto funcionamiento.
- Este tipo de herramientas permite configurar la célula con una serie de periféricos (cintas transportadoras, mesas giratorias, sensores de posición, etc.) e, incluso, otros brazos manipuladores de forma que aunque no se disponga de ellos en la célula real, el alumno puede aprender las técnicas de sincronización entre todos ellos.
- La utilización de este tipo de herramientas ha tenido una buena aceptación por parte del alumnado ya que cada uno (o pareja de prácticas) dispone de un puesto para trabajar, evitando así el trabajar por turno o todo el grupo simultáneamente en la célula robotizada. Además, el disponer de una interfaz gráfica amigable facilita el aprendizaje de los conceptos y técnicas relacionados con la materia.

Actualmente, se está trabajando en la mejora de prestaciones de la herramienta entre las que destacan:

- Ampliación de periféricos. Se deben poder introducir otros periféricos para ampliar el número de entornos robotizados que se pueden modelar, como una mesa giratoria, una base lineal deslizante para la ampliación del volumen de trabajo del manipulador o alimentadores de piezas.
- Ampliación del tipo de brazos manipuladores, de forma que se pueda simular el trabajo con otros robots comerciales que presenten diferentes características.
- Creación de entornos multirobot, introduciendo más de un brazo manipulador en el entorno; lo que permite la simulación de cadenas de montaje, realización de practicas sobre coordinación de robots, etc.
- Posibilidad de teleoperación vía Internet, utilizando una arquitectura cliente-servidor, de forma que el usuario pueda trabajar con la célula robotizada de forma remota. Se utiliza la herramienta de simulación para programar la célula y estudiar su comportamiento. Cuando éste sea correcto, se puede observar la ejecución real del mismo transmitiendo una señal de vídeo.

Agradecimientos

El trabajo se enmarca en las líneas de investigación del Grupo de Automática, Electrónica y Robótica (TEP-197) del Plan Andaluz de Investigación.

Referencias

[1] Barrientos, Peñín, Balaguer, Aracil; Fundamentos de Robótica; Ed. Mc Graw Hill, 1997.
[2] Eshed Robotec, Manual de usuario del SCORBOT - ER V. 1994.
[3] Eshed Robotec, ACL Lenguaje de Control Avanzado. Guía de Referencia. 3ª Edición. Noviembre 1994.
[4] Eshed Robotec, SCORBASE Nivel 5. Guía de referencia, 3ª ed., 1.996
[5] Fu K.S.; et al..; Robótica. Control, detección, visión e inteligencia. Ed. Mc Graw Hill, 1990.
[6] Iribarne, L. Fundamentos en Informática Gráfica. Servicio de publicaciones de la Universidad de Almería, 2001.
[7] Kerlow, I. V. The Art of 3-D: Computer Animation and Imaging, 2nd Edition. John Wiley & Sons, 2000.
[8] O'Rourke, M. Principles of Three-Dimensional Computer Animation: Modeling, Rendering, and Animating With 3d Computer Graphics. W.W. Norton & Company, 1998.
[9] Pappas, C.H.,Murray, W.H.; Visual C++ 6.0, Ed. Mc Graw Hill, 1999
[10] Spong M.W. and M. Vidyasagar. Robot Dynamic and Control. John Wiley and Sons, 1989.
[11] Woo M., et al. OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.2. Addison-Wesley, 1999.