UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALE

SIMULADOR PARA SISTEMA ROBOTIZADO EN CALZADO

AUTORA: ALBA MONTIEL MIRALLES

TUTOR: LEOPOLDO ARMESTO ÁNGEL

COTUTOR:

Curso Académico: 2013-14

ÍNDICE GENERAL

MEMORIA

ÍNDICE

1 OBJETO DEL TRABAJO

El principal objetivo para el cual se realiza este trabajo, a parte de una ligera introducción en el mundo laboral y acercamiento hacia los problemas reales que surgen en una empresa, es la finalización del Grado en Tecnologías Industriales, para poder acceder posteriormente al máster en Ingeniería Industrial.

Este trabajo final de grado tiene como finalidad la utilización de una herramienta software de simulación para diseñar un sistema automático robotizado aplicado a un proceso de producción en un entorno industrial. En concreto se trata de la aplicación de un spray de acabado que se aplica al calzado previa venta al público. La automatización de este, junto a otros procesos de acabado como puede ser el pulido, está siendo desarrollada en la empresa INESCOP (Elda) dado que hoy en día se pretende que dichos procesos sean realizados completamente por robots industriales.

Analizaremos las ventajas que ofrecen las herramientas actuales para la simulación “offline” en los entornos productivos frente a la programación realizada en el entorno real. Para poder acometer este objetivo será necesaria la elección de una herramienta utilizada en la actualidad para llevar a cabo el caso práctico. Para implementar el proceso se deben de asimilar y comprender los métodos de programación que admite la herramienta de simulación, además de las distintas funcionalidades y opciones de configuración de este software para realizar el proceso de forma virtual.

De esta forma se pretende llevar a cabo el proceso real de una forma simulada, habiendo realizado previamente la configuración total del sistema: definición de herramientas del robot, especificación de sistemas de referencia, definición de piezas y objetos, definición de partes fijas, la definición de puntos y el trazado de las distintas trayectorias.

1.1 Alcance del proyecto

A lo largo de este proyecto se procederá a la simulación de diferentes situaciones mediante una herramienta software, que posteriormente podrán ser de aplicación al entorno industrial real. Cuando se habla de simulación aparece un amplio abanico de posibilidades a desarrollar y de problemas a enfrentar.

El alcance de este proyecto va a limitar esas posibilidades al diseño 3D del robot para la posterior resolución de su problema cinemático (tratándolo como un mecanismo de cadena cerrada). En adición, posteriormente se analizarán diferentes situaciones que se pueden encontrar en la celda de trabajo real como: posibles colisiones con el entorno, programación para el seguimiento de trayectorias por el robot o de qué modo crear controles de usuario con la herramienta de simulación. Cualquier acción adicional que pudiera ser realizada queda fuera del alcance de este proyecto.

2 INTRODUCCIÓN AL PROBLEMA

2.1 Antecedentes

Se empieza comentando los antecedentes referidos al objeto de este estudio, es decir, los de la robótica. El inicio de la robótica actual puede fijarse en la industria textil del siglo XVIII, cuando Joseph Jacquard inventa en 1801 una máquina textil programable mediante tarjetas perforadas.

Luego, la Revolución Industrial impulsó el desarrollo de estos agentes mecánicos muy ingeniosos que tenían algunas características de robots. Jacques de Vauncansos construyó varios músicos de tamaño humano a mediados del siglo XVIII. En 1805, Henri Mailardert construyó una muñeca mecánica que era capaz de hacer dibujos.

La palabra robot se utilizó por primera vez en 1920 en una obra llamada “Los Robots Universales de Rossum”, escrita por el dramaturgo checo Karel Capek. Su trama trataba sobre un hombre que fabricó un robot y luego este último mata al hombre. La palabra checa ‘Robota’ significa servidumbre o trabajo forzado, y cuando se tradujo al inglés se convirtió en el término robot.

Más tarde, Isaac Asimov comenzó en 1939 a contribuir con varias relaciones referidas a robots. A él se le atribuye el acuñamiento del término Robótica y con él surgen las denominadas “Tres Leyes de Robótica” que son las siguientes:

1. Un robot no puede actuar contra un ser humano o, mediante la inacción, que un ser humano sufra daños.
2. Un robot debe obedecer las órdenes dadas por los seres humanos, salvo que estén en conflicto con la primera ley.
3. Un robot debe proteger su propia existencia, a no ser que este en conflicto con las dos primeras leyes.

Son varios los factores que intervienen para que se desarrollaran los primeros robots en la década de los 50’s. La investigación en inteligencia artificial desarrolló la manera de emular el procesamiento de información humana con computadoras electrónicas e inventó una variedad de mecanismos para probar sus teorías. Las primeras patentes aparecieron en 1946 con robots muy primitivos para traslado de maquinaria de Devol. También en ese año aparecen las primeras computadoras. En 1954, Devol diseña el primer robot programable.

En 1960 se introdujo el primer robot “Unimate”, basado en la transferencia de artículos. Un año después, en 1961, un robot Unimate se instaló en la Ford Motors Company para atender una máquina de fundición de troquel.

En 1966 Trallfa, una firma noruega, construyó e instaló un robot de pintura por pulverización.

En 1971 el “Stanford Arm”, un pequeño brazo robot de accionamiento eléctrico, se desarrolló en la Stanford University y siete años después, en 1978, se introdujo el robot PUMA para tareas de montaje por Unimation, basándose en diseños obtenidos en un estudio de la General Motors.

Actualmente, el concepto de robótica ha evolucionado hacia los sistemas móviles autónomos, que son aquellos que son capaces de desenvolverse por sí mismos en entornos desconocidos y parcialmente cambiantes sin necesidad de supervisión. En los setenta, la NASA inicio un programa de cooperación con el Jet Propulsión Laboratory para desarrollar plataformas capaces de explorar terrenos hostiles.

Hoy en día, la robótica se debate entre modelos sumamente ambiciosos, como es el caso del IT, diseñado para expresar emociones, el COG, también conocido como el robot de cuatro sentidos, el famoso SOUJOURNER o el LUNAR ROVER, vehículo de turismo con control remotos, y otros mucho más específicos como el CYPHER, un helicóptero robot de uso militar, el guardia de trafico japonés ANZEN TARO o los robots mascotas de Sony.

En cuanto a antecedentes industriales se ha de mencionar a INESCOP-Instituto Tecnológico del Calzado y Conexas. Esta es una organización sin ánimo de lucro creada en 1971 por iniciativa de la Feria Internacional del Calzado de Elda. Entre sus asociados se encuentran más de 500 empresas de toda España. Como organización desarrolla una serie de actividades de carácter científico y técnico que las industrias del sector calzado, debido a su pequeña dimensión, no pueden abordar a nivel individual. Como centro de innovación y tecnología sectorial, proporciona servicios directos, transfiere conocimientos e investiga sobre temas de interés general.

La sede central de INESCOP se encuentra en Elda (Alicante), y cuenta además con una red de 8 Unidades Técnicas en los principales núcleos productores de calzado: ELCHE, VILLENA, INCA, ARNEDO, ALMANSA, VALL D'UIXÓ,FUENSALIDA e ILLUECA.

Las empresas necesitan de la innovación para mantener y mejorar su competitividad, especialmente las pequeñas, y entre ellas las de calzado, ya que se desenvuelven en un ambiente sumamente agresivo y apenas dominan los elementos del mercado, sino que son dominadas por ellos.

Debido a la dependencia tradicional que el proceso de fabricación de calzado tiene respecto a la mano de obra, la tecnología y la robótica son factores clave en el desarrollo de este sector en nuestro país, frente a la competencia de aquellos que cuentan con bajos costos salariales.

El equipo de INESCOP ha llevado a cabo recientemente varios proyectos de investigación en los que la robótica y la automatización es uno de los aspectos abordados.

En cuanto a Automática y Control Numérico nace como empresa de base tecnológica, spin-off del Instituto Tecnológico del Calzado y Conexas (INESCOP).

A&CN, opera en el sector de la automatización industrial con especial énfasis en el sector del calzado e industria de componentes del calzado, realizando máquinas especiales y proyectos a medida de elevado nivel tecnológico. Su potencial en I+D+i, les permite ofertar soluciones concretas según necesidades, ya sea modernizando los equipos ya existentes en las empresas, mediante la incorporación de los más sofisticados avances tecnológicos, como emprendiendo nuevos proyectos que puedan permitir a las empresas diferenciarse de su competencia.

2.2 Motivación

La industria de calzado tiene la consideración de tradicional e intensiva en mano de obra, sin embargo, los procesos que van desde la venta, pasando por el diseño y la fabricación hasta la entrega del calzado al cliente final, son y pueden ser objeto de aplicación de las tecnologías más avanzadas disponibles o futuras, dotando de ese modo a la industria de una competitividad que permitiría continuar manteniendo el liderazgo en determinados segmentos en los que la cercanía al cliente o el elemento moda, constituyen un valor añadido altamente apreciado por el mercado.

Como ya está ocurriendo en otros sectores, el sector calzado está atravesando por una etapa crucial en un entorno económico cada vez más complejo: se enfrenta a nuevas fronteras de competitividad, en un escenario global donde los competidores del este y del lejano oriente se benefician de ventajas incomparables en cuanto a volúmenes y costes de mano de obra.

En este sentido, para seguir siendo competitivos y dar un futuro a esta tradición distintiva, resulta esencial que el sector calzado europeo modernice su base de fabricación y refuerce los vínculos entre la investigación y la innovación con el fin de contribuir al objetivo de hacer de la UE “la economía basada en el conocimiento más competitiva y dinámica del mundo, capaz de conseguir un crecimiento sostenible con mejores puestos de trabajo y mayor cohesión social”.

Actualmente existe una elevada componente manual, con ayuda de máquinas, en muchos de los procesos implicados en la producción de calzado. Algunos procesos de fabricación (para el calzado y sus componentes) son asistidos por maquinaria especializada (fabricación de la horma, montado y corte) y existen líneas de alto grado de automatización en la producción masiva de calzado técnico (calzado de seguridad). Pero la mayoría de la producción es todavía artesanal, siendo especialmente cierto en el caso de la producción de zapatos de alto valor añadido, en los que Europa-y España- mantienen su liderazgo.

El papel de la robótica en el proceso de modernización e innovación de las industrias es indiscutible. Desde su aplicación a la industria de automoción, ha supuesto una revolución a la hora de sustituir al hombre en operaciones repetitivas sin valor, muy complejas o con elevado riesgo, operaciones que se dan en ambientes tóxicos...etc., mejorando así la productividad y la calidad final del producto.

Hasta el momento su implantación se ha llevado a cabo sobre todo en entornos estructurados en donde el espacio de trabajo ha sido ideado para adecuarse al robot. Sin embargo históricamente ha habido un menor impacto –y una mayor dificultad- en entornos dinámicos, de elevada frecuencia de cambio en la geometría de los modelos a fabricar, donde el área de trabajo y la posición de los objetos no pueden controlarse exactamente, tal y como ocurre con multitud de procesos en una fábrica de calzado.

Esta limitación se debe en parte a que es necesario desarrollar robots con una elevada capacidad adaptativa, que resuelvan la incertidumbre debida a una elevada variabilidad del entorno, como ocurre con la gran cantidad de piezas y variables que se dan en un zapato.

La introducción de la robótica inteligente, contribuirá a superar la complejidad en la automatización de los procesos de esta industria en la que las series de producción son pequeñas. Las principales dificultades para lograr este objetivo son:

- El elevado número de productos *variantes.*
- *Fabricación a través de* procesos *complejos.*
- *Amplitud de tareas* requeridas en algunos procesos.
- *Elevada destreza manual requerida.*

2.3 Justificación

La principal justificación para la realización de este trabajo de fin de grado tiene un motivo académico pues se pretende obtener el título de Graduada en Tecnologías Industriales por la Escuela Técnica Superior de Ingenieros Industriales de Valencia.

En cuanto a la justificación industrial, se ha definido automatización como el conjunto de tecnologías asociadas a la aplicación de sistemas de tipo mecánico, electrónico y basado en ordenador, a la operación y control de la producción. Entre las tecnologías involucradas en dicho concepto encontramos los robots industriales, objeto de este trabajo.

El uso de los robots en industria se viene incrementando considerablemente en los últimos años, así como su principal uso e implantación en líneas de producción. Hoy en día hay más de 800 mil unidades de sistemas robóticos a nivel mundial.

Su gran aceptación se debe a las numerosas ventajas que conlleva. Encontramos que con un sistema automatizado, a cada producto que viene de la línea de producción se le puede garantizar su calidad, es decir, las máquinas producirán productos terminados que serán determinados sólo por el valor de la materia prima que fue ingresada. El producto, en definitiva, no está sujeto a error humano. Luego con la implantación de robots en la industria conseguimos una producción más eficiente, con un coste más reducido y de buena calidad.

Los fabricantes y expertos en calzado saben muy bien que la automatización de los procesos de fabricación de calzado de calidad alta-media tiene una elevada dificultad por la elevada variabilidad de piezas que existen y las series cada vez más cortas de fabricación (ya no se puede hablar de fabricación masiva).

Por lo tanto es obligatorio realizar las siguientes reflexiones:

1. ¿Cuál es el tiempo de reconfiguración del robot?

Habría que analizar si merece la pena automatizar el proceso o no, teniendo en cuenta que puede que el operario pierda demasiado tiempo en reconfigurar el robot, más del que emplearía en aplicar el spray, con lo que no tendría sentido robotizar el proceso.

En la actualidad, la mayoría de aplicaciones robot se basan en procedimientos de Teaching, en donde el robot es guiado por las zonas en las que se quiere realizar la operación y las trayectorias se memorizan y se ejecutan cuando corresponda. En los casos en los que no hay mucha variabilidad de piezas (automoción, soldadura de muebles metálicos, pintado...etc.) se pueden asumir los tiempos de teaching. Sin embargo en nuestro caso hay que analizarlo mediante la siguiente fórmula:

*(Tiempo de teaching / nºpiezas) * Coste-hora especialista*

La cifra resultante debe ser menor que el coste máximo asumible. Por otro lado, el tiempo de adaptación o reconfiguración debe ser inferior al periodo operativo de la cadena de producción. En nuestro caso, tomando la producción de 800 pares por día, 8 horas diarias y dos piezas diferentes (modelos y tallas distintas) el tiempo de reconfiguración es de 18 segundos, que es el resultado de dividir el tiempo en segundos entre el número de pares por el número de piezas. Esto significa que el tiempo de reconfiguración entre dos piezas diferentes debe ser inferior a 18 segundos.

2. ¿Por qué escoger un robot en lugar de una máquina?

En este caso hay que analizar si es mejor emplear un robot que las máquinas que ya existen en el mercado. El proceso que vamos a automatizar se realiza actualmente de forma manual, con lo que esta cuestión no se aplicaría a nuestro caso.

Las máquinas orientadas a una aplicación presentan una ventaja en cuanto a que los elementos están optimizados y simplifican la operación, pero presentan el inconveniente de la rigidez operativa y el coste del desarrollo, ya que sólo son válidas para una determinada aplicación, con lo que podemos encontrarnos con limitaciones importantes (por ejemplo: una máquina para aplicar adhesivo puede funcionar bien para zapato de caballero, pero presentaría dificultades para un calzado de señora con un quiebre muy pronunciado). Por el contrario, un robot es mucho más versátil, permitiendo más aplicaciones y posibilidades con un simple cambio de herramienta.

Como se ha comentado, la cuestión está en “los periodos de reconfiguración o adaptación”. La generación de trayectorias fuera de línea y la adaptación de geometrías mediante visión artificial, permiten que la robotización sea viable en este tipo de aplicaciones.

3.Dada la elevada variabilidad de piezas que existe ¿se puede emplear un robot para todos los modelos?

Esto obliga a plantearse dos posibles escenarios: o el proceso es totalmente robotizado o los dos procesos, uno robotizado y otro manual, coexisten. En el segundo caso ¿seguiría siendo rentable?

Los fabricantes tienen la necesidad de hacer frente a estos retos para mejorar su productividad. La robótica hoy en día permite automatizar procesos antes imposibles, aunque aún haya muchos procesos imposibles de abordar, tanto a nivel técnico como a nivel económico.

Al fin y al cabo se llega a la conclusión de que resultaría rentable implantar un sistema robótico siempre que este tenga la capacidad de adaptarse en tiempo real a estas condiciones de trabajo, entre 14 y 18 segundos para:

- Coger el zapato de una zona determinada sin dañar la piel.
- Identificar el zapato.
- Identificar la operación a realizar, lijado, aplicación adhesivo, tintado, aplicación de spray, pulido…
- Identificar la posición real del zapato en la cadena. (Visión: Digitalización rápida optimizada)
- Adaptar la trayectoria sobre la posición real del zapato (roto traslación).
- Generar la trayectoria de la operación a realizar.
- Habilitar los elementos auxiliares, motores, spray, etc.
- Ejecutar la operación.

2.3.1 Justificación del uso de la simulación

Cuando se realiza un estudio completo para la automatización de un proceso industrial es necesario el uso de la simulación del proceso mediante ordenador. Ésta es la encargada de representar un escenario con un robot y cómo éste interactúa con el mundo.

Nos brinda la oportunidad de realizar un análisis previo del comportamiento que tendrá el robot una vez puesto en marcha a fin de analizar si podrá realizar una determinada tarea teniendo en cuenta aspectos tan importantes como posibles colisiones con elementos situados en su entorno y su capacidad para seguir una determinada trayectoria previa definición de las coordenadas y orientación con las que deberá alcanzarla.

Hoy en día, hay numerosas herramientas software que nos permiten analizar el proceso a través de la simulación completa de las células robotizadas mediante ordenadores.

El concepto de simulación es posible gracias a la programación “offline” del sistema mediante el software específico.

Las características de estas aplicaciones suelen ser muy avanzadas de modo que nos permiten ajustar al máximo todos los parámetros de los movimientos y acciones de nuestro robot, incluso de trasladar esta programación directamente al robot y evitar la reprogramación de todo el proceso una vez montado físicamente el sistema en el entorno de producción.

Estas técnicas de programación conocidas como programación fuera de línea (offline programming, OLP) utilizadas de forma adecuada nos permitirían realizar las siguientes tareas antes de trabajar en un entorno productivo:

- Desarrollar modelos, probarlos y optimizarlos, antes de ser utilizados en la fabricación de piezas y herramientas.

- Conocer cuál será el comportamiento de los sistemas antes de construirlos, sin perder de vista que los valores finales de los simuladores serán aproximaciones de los valores reales.

- Reprogramar el proceso fuera de una línea de fabricación que se encuentre ya en producción, si por alguna razón, cambian las necesidades.

- Anticipar el funcionamiento y puesta en servicio de las líneas de producción, ya que es un sistema independiente y puede realizarse en paralelo con el montaje de células robóticas.

- Diseñar correctamente las trayectorias del elemento terminal (pinza o garra) así como sus velocidades y aceleraciones. Durante el modelado los programas simuladores informan de cualquier colisión o perdida de proximidad entre los elementos del modelo y del entorno

3 ANÁLISIS PREVIO DE LOS SISTEMAS ROBÓTICOS

3.1 Universo del robot industrial

"Un robot industrial es un manipulador multifuncional reprogramable diseñado para desplazar materiales, piezas, herramientas o dispositivos especiales, mediante movimientos variables programados para la ejecución de una diversidad de tareas".

Robotics Industries Association (RIA)

El proceso de planificación de un proceso industrial para la producción en cadena, o cualquier tipo de actividad que tenga un una función determinada en un sistema automático, puede plantearse como una acción automática ejecutada por un manipulador robótico industrial. Así mismo, debemos considerar que cada acción dentro de una cadena de producción automatizada tiene una serie de condiciones que el robot manipulador debe cumplir.

Un robot está formado por los siguientes elementos: estructura mecánica, transmisiones, sistema de accionamiento, sistema sensorial, sistema de control y elementos terminales.

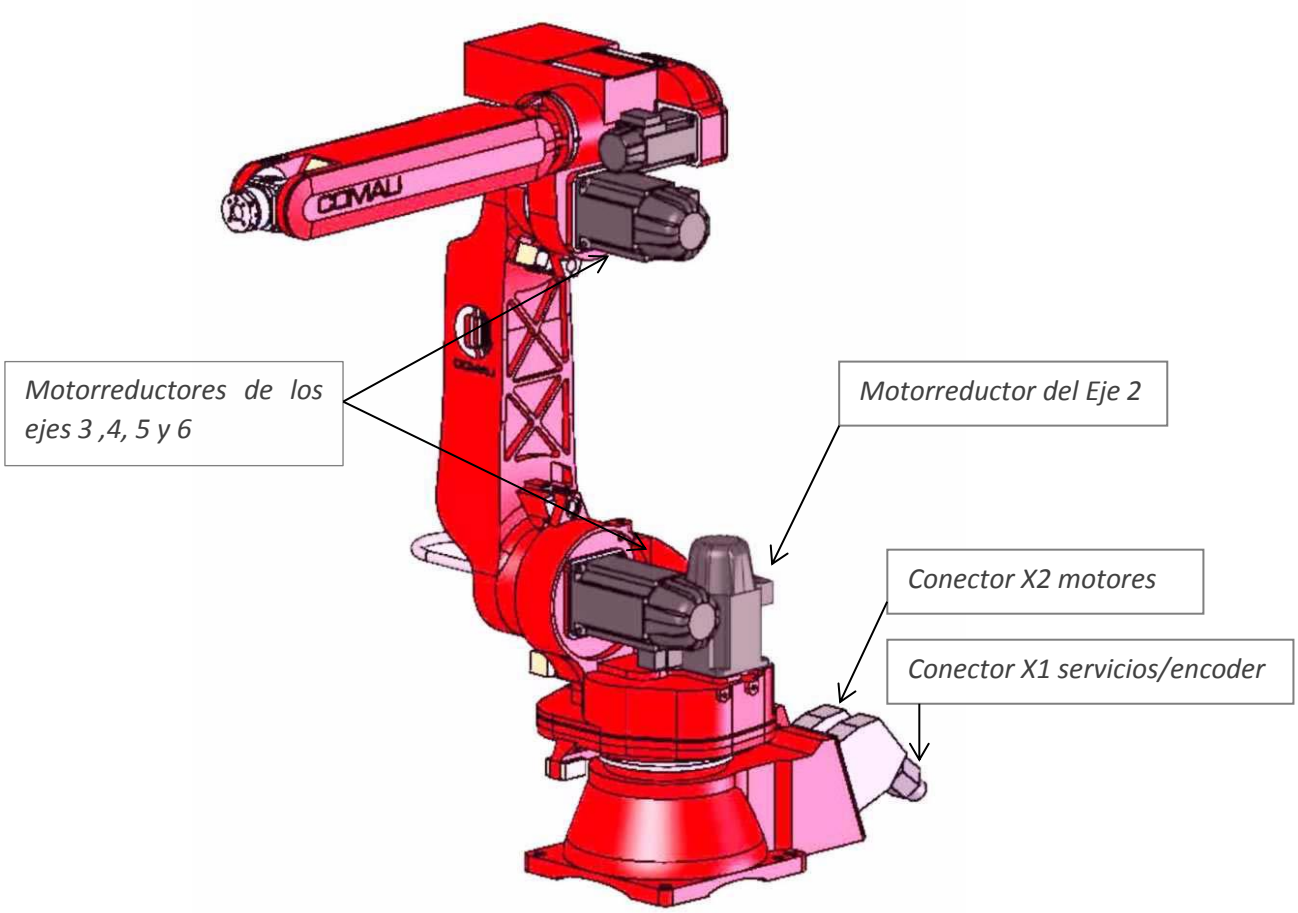


Fig.3.1 Ejemplo de estructura mecánica y elementos constitutivos del manipulador.

Mecánicamente, un robot está formado por una serie de elementos o eslabones unidos mediante articulaciones que permiten un movimiento relativo entre cada dos eslabones consecutivos. El movimiento de cada articulación puede ser de desplazamiento, de giro o una combinación de ambos. De este modo son posibles seis tipos diferentes de articulaciones (Fig.3.2), aunque en la práctica los robots sólo suelen emplear la de rotación y la prismática.

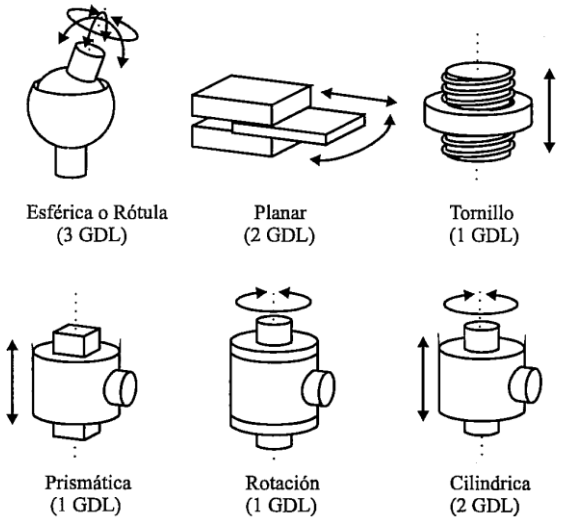


Fig. 3.2 Distintos tipos de articulaciones para robots.

Cada uno de los movimientos independientes que puede realizar cada articulación con respecto a la anterior se denomina grado de libertad. En la figura anterior se indica el número de GDL de cada tipo de articulación.

Puesto que para posicionar y orientar un cuerpo de cualquier manera en el espacio son necesarios seis parámetros (tres para definir la orientación y tres para definir la posición) si se pretende que un robot posicione y oriente su extremo de cualquier modo en el espacio se necesitarán al menos seis GDL. Existen también casos opuestos en los que se precisan más de seis GDL para que el robot pueda tener acceso a todos los puntos de su entorno. Cuando el número de grados de libertad del robot es mayor que los necesarios para realizar una determinada tarea se dice que el robot es redundante.

Las transmisiones son los elementos encargados de transmitir el movimiento desde los actuadores hasta las articulaciones. Se incluirán junto con las transmisiones a los reductores, encargados de adaptar el par y la velocidad de la salida del actuador a los valores adecuados para el movimiento de los elementos del robot.

Actualmente en el mercado son habituales los robots con accionamiento directo, en el que el eje del actuador se conecta directamente a la carga o articulación, sin la utilización de un reductor intermedio. Suele utilizarse para robots con accionamiento eléctrico.

Este tipo de accionamiento aparece a raíz de la necesidad de utilizar robots en aplicaciones que exigen combinar gran precisión con alta velocidad.

Entre los motores empleados para accionamiento directo se encuentran los motores síncronos y sin escobillas (*brushless).*

Los actuadores tienen por misión generar el movimiento de los elementos del robot según las órdenes dadas por la unidad de control. Los actuadores utilizados en robótica pueden emplear energía neumática, hidráulica o eléctrica. Cada uno de estos sistemas presenta características diferentes siendo necesario evaluarlas antes de elegir el actuador más conveniente.

La información que la unidad de control del robot puede obtener sobre el estado de su estructura mecánica es fundamentalmente la relativa a su posición y velocidad. En la siguiente tabla se resumen los sensores más comúnmente empleados para obtener información de presencia, posición y velocidad en robots industriales.

Presencia	· Inductivo · Capacitivo · Efecto hall · Célula Reed · Óptico · Ultrasonido · Contacto	
Posición	Analógicos	· Potenciómetros · Resolver · Sincro · Inductosyn · LVDT
	Digitales	· Encoders absolutos · Encoders incrementales · Regla óptica
Velocidad	· Tacogeneratriz	

Fig. 3.3 Tipos de sensores internos de robots

Los elementos terminales también llamados efectores finales (end effector) son los encargados de interaccionar directamente con el entorno del robot. Pueden ser tanto elementos de aprehensión como herramientas. Si bien un mismo robot industrial es, dentro de unos límites lógicos, versátil y readaptable a una gran variedad de aplicaciones, no ocurre así con los elementos terminales, que son en muchos casos específicamente diseñados para cada tipo de trabajo.

3.2 Fundamentos matemáticos de movimiento del robot.

La manipulación de piezas llevada a cabo por un robot implica el movimiento espacial de su extremo. Asimismo, para que el robot pueda recoger una pieza, es necesario conocer la posición y orientación de ésta con respecto a la base del robot. Se aprecia entonces la necesidad de conocer una serie de herramientas matemáticas que permitan especificar la posición y orientación en el espacio de piezas, herramientas y en general de cualquier objeto.

Estas herramientas han de ser lo suficientemente potentes como para permitir obtener de forma sencilla relaciones espaciales entre distintos objetos y en especial entre éstos y el manipulador.

3.2.1 Representación de la posición

Para localizar un cuerpo rígido en el espacio es necesario contar con la localización espacial de sus puntos. En el caso de un espacio tridimensional será necesario emplear tres componentes. La forma más intuitiva y utilizada de especificar la posición de un punto son sus coordenadas cartesianas.

Los sistemas de referencia se definen mediante ejes perpendiculares entre sí con un origen definido. Estos se denominan sistemas cartesianos. Trabajando en el espacio 3D, el sistema cartesiano OXYZ está compuesto por una terna ortonormal de vectores de coordenadas OX, OY y OZ, tal y como se aprecia en la figura 3.4.

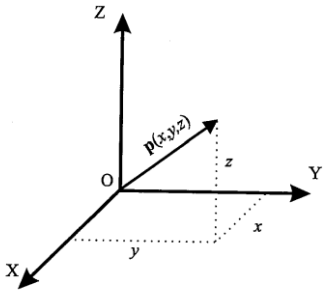


Fig. 3.4 Terna ortonormal a derechas

Un vector p cualquiera vendrá definido con respecto al sistema de referencia OXYZ mediante las coordenadas correspondientes a cada uno de los ejes coordenados, p(x,y,z).

3.2.2 Representación de la orientación

Un punto queda totalmente definido en el espacio a través de los datos de su posición. Sin embargo para el caso de un sólido es necesario además definir su orientación con respecto a un sistema de referencia. En el caso de un robot ocurre lo mismo, no es suficiente solo con especificar cuál debe ser la posición de su extremo sino que, en general, es también necesario indicar su orientación.

Una orientación en el espacio 3D viene definida por tres grados de libertad. Para describir de forma sencilla la orientación de un objeto respecto a un sistema de referencia, asignaremos

solidariamente al objeto un nuevo sistema y estudiaremos la relación espacial existente entre los dos sistemas. Esta relación vendrá dada por la posición y orientación del sistema asociado al objeto respecto al de referencia. Para el análisis de los distintos métodos de representar orientaciones se supondrá que ambos sistemas coinciden en el origen, y que por tanto, no existe cambio alguno de posición entre ellos.

3.2.2.1 Matrices de rotación

Las matrices de rotación son el método más extendido para la descripción de orientaciones, debido principalmente a la comodidad que proporciona el uso del algebra matricial. En un espacio tridimensional, supónganse los sistemas OXYZ y OUVW coincidentes en el origen (Fig3.5.a). Siendo el sistema OXYZ el sistema de referencia fijo y OUVW el solidario al objeto cuya orientación se desea definir. Los vectores unitarios del sistema OXYZ serán i_x, j_y, k_z, mientras que los del OUVW serán i_u, j_v, k_w. Un vector p del espacio podrá ser referido a cualquiera de los sistemas de la siguiente manera:

$$\mathbf{p}_{uvw} = [p_u, p_v, p_w]^T = p_u \cdot \mathbf{i}_u + p_v \cdot \mathbf{j}_v + p_w \cdot \mathbf{k}_w$$
$$\mathbf{p}_{xyw} = [p_x, p_y, p_z]^T = p_x \cdot \mathbf{i}_x + p_y \cdot \mathbf{j}_y + p_z \cdot \mathbf{k}_z$$

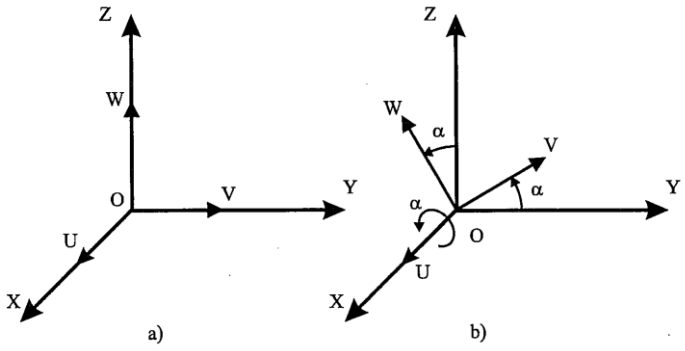


Fig. 3.5 a) Sistema de referencia OXYZ solidario al objeto OUVW
b) Representación de la orientación con giro del eje OU coincidente al OX.

Obteniéndose la siguiente equivalencia:

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \mathbf{R} \begin{bmatrix} p_u \\ p_v \\ p_w \end{bmatrix}$$

$$\mathbf{R} = \begin{bmatrix} \mathbf{i}_x \mathbf{i}_u & \mathbf{i}_x \mathbf{j}_v & \mathbf{i}_x \mathbf{k}_w \\ \mathbf{j}_y \mathbf{i}_u & \mathbf{j}_y \mathbf{j}_v & \mathbf{j}_y \mathbf{k}_w \\ \mathbf{k}_z \mathbf{i}_u & \mathbf{k}_z \mathbf{j}_v & \mathbf{k}_z \mathbf{k}_w \end{bmatrix} \qquad \mathbf{R}(\mathbf{x},\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\operatorname{sen}\alpha \\ 0 & \operatorname{sen}\alpha & \cos\alpha \end{bmatrix}$$

Donde R es la matriz de rotación que define la orientación del sistema OUVW respecto a OXYZ y la matriz R(x, α) la correspondiente a la orientación del sistema OUVW con giro de OU.

Tambien recibe el nombre de matriz de cosenos directores y se trata de una matriz ortonormal tal que la inversa de R es igual a su traspuesta. La principal utilidad de esta matriz de rotacion corresponde a la respresentacion de la orientacion de sistemas girados unicamente sobre uno de los ejes principales del sistema de referencia.

3.2.2.2 Ángulos de Euler

Para la representacion de orientación en un espacio tridimensional mediante una matriz de rotación es necesario definir nueve elementos. Aunque la representación de las matrices de rotación presente múltiples ventajas, existen otros metodos de definición de orientación que hacen únicamente uso de tres componentes para su descripción. Este es el caso de los llamados ángulo de Euler que utilizaremos para la definicion de la trayectoria a seguir por el robot.

Todo sistema OUVW, solidario al cuerpo cuya orientacion se quiera describir, puede definirse con respecto al sistema OXYZ mediante tres ángulos: ϕ,θ,ψ, denominados ángulos de Euler. Girando sucesivamente el sistema OXYZ sobre unos ejes determinados de un triedro ortonormal los valores de ϕ,θ,ψ, se obtendrá el sistema OUVW. Es necesario, por tanto, conocer además de los valores de los ángulos, cuáles son los ejes sobre los que se realizan los giros.

3.2.2.2.1 Ángulos de Euler ZXZ

Es una de las notaciones más habituales entre las que realizan los giros sobre ejes previamente girados. Si se parte de los sistemas OXYZ y OUVW, inicialmente coincidentes, se puede colocar al sistema OUVW en cualquier orientacion siguiendo los siguientes pasos:

a. Girar el sistema OUVW un ángulo ϕ con respecto al eje OZ, convirtiéndose así en el OU'V'W'.

b. Girar el sistema OU'V'W' un ángulo θ con respecto al eje OU', conviertiéndose así en el OU''V''W''.

c. Girar el sistema OU''V''W'' un ángulo ψ con respecto al eje OW'' conviertiéndose finalmente en OU'''V'''W'''.

Es importante que estas operaciones se realicen en la secuencia especificada, pues las operaciones de giros consecutivos sobre ejes no son conmutativas.

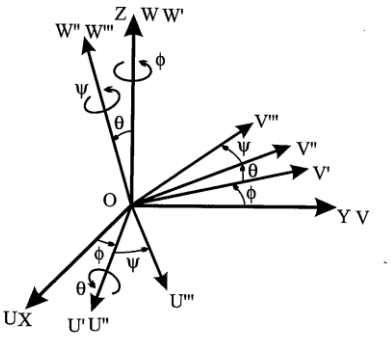


Fig. 3.6 Ángulos Euler ZXZ

3.2.3 Traslaciones y rotaciones

Una vez vistos los métodos para representar la posición u orientación de un sólido en el espacio que utilizaremos nos falta hallar el método de representar conjuntamente ambas, es decir, la localización de un sólido en el espacio. Para ello se introduce el concepto de coordenadas homogéneas.

Un espacio n-dimensional se encuentra representado en coordenadas homogéneas por n+1 dimensiones. De modo que un vector p(x,y,z) vendrá representado por p(wx,wy,wz,w) donde w tiene un valor arbitrario y representa un factor de escala que suele tomar el valor 1.

Las traslaciones pueden entenderse como movimientos directos sin cambios de orientación. Generalmente se usan coordenadas homogéneas para representar la traslación mediante una matriz de transformación homogénea, y poder así expresarla como una transformación lineal sobre un espacio de dimensión superior.

$$T_{\mathbf{v}} = \begin{bmatrix} 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & v_y \\ 0 & 0 & 1 & v_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Tv es la denominada matriz básica de traslación. Un vector cualquiera p desplazado según Tv tendrá como componentes el resultado de la multiplicación de esta matriz por la representación en coordenadas homogéneas del vector.

$$T_{\mathbf{v}}\mathbf{p} = \begin{bmatrix} 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & v_y \\ 0 & 0 & 1 & v_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} p_x + v_x \\ p_y + v_y \\ p_z + v_z \\ 1 \end{bmatrix} = \mathbf{p} + \mathbf{v}.$$

Debido a que la suma de vectores es conmutativa, la multiplicación de matrices de traslación es también conmutativa, a diferencia de lo que sucede con matrices arbitrarias, que no necesariamente representan traslaciones.

La rotación es el movimiento de cambio de orientación de un cuerpo o un sistema de referencia de forma que una línea (*eje de rotación*) o un punto permanece fijo. Se pueden definir tres matrices homogéneas básicas de rotación según se realice ésta respecto a cada uno de los tres ejes coordenados OX, OY y OZ de un sistema de referencia OXYZ.

$$T(x,\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -sen(\alpha) & 0 \\ 0 & sen(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T(y,\phi) = \begin{bmatrix} \cos(\phi) & 0 & sen(\phi) & 0 \\ 0 & 1 & 0 & 0 \\ -sen(\phi) & 0 & \cos(\phi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T(z,\theta) = \begin{bmatrix} \cos(\theta) & -sen(\theta) & 0 & 0 \\ sen(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A su vez un vector rotado r(rx, ry, rz) por T vendrá expresado por r' tal que:

$$\begin{bmatrix} r'_x \\ r'_y \\ r'_z \\ 1 \end{bmatrix} = \mathbf{T} \begin{bmatrix} r_x \\ r_y \\ r_z \\ 1 \end{bmatrix}$$

La traslación y la rotación son transformaciones que se realizan en relación a un sistema de referencia, por lo tanto, si se quiere expresar la posición y orientación de un sistema O'UVW, originalmente coincidente con el de referencia y que ha sido rotado y trasladado según este, habrá que tener en cuenta si primero se ha realizado la rotación y después la traslación o viceversa, pues se trata de transformaciones espaciales no conmutativas.

4 HERRAMIENTAS DE TRABAJO

4.1 Presentación del manipulador

SMART NS es la familia de robot COMAU apta para aplicaciones de manipulación liviana y soldadura de arco. Para la ejecución del proceso se ha escogido el robot SMART5 SiX.

El SMART5 SiX es un robot antropomorfo equipado con 6 ejes que soporta una carga máxima en pulso de 6kg y una carga adicional en el antebrazo de 10kg, ofreciendo una extensión máxima horizontal de 1400mm con una repetibilidad o capacidad para posicionarse en un punto previamente accesado de +/- 0.05mm.

VERSIÓN		SIX 6-1.4
Estructura / n° ejes		Antropomorfo / 6 ejes
Carga en el pulso		6 kg(1)
Carga adicional en el antebrazo		10 kg(2)
Par eje 4		11,7 Nm
Par eje 5		11,7 Nm
Par eje 6		5,8 Nm
Carrera /(Velocidad)	Eje 1	+/- 170°(140°/s)
	Eje 2	+155°/-85°(160°/s)
	Eje 3	0°/-170°(170°/s)
	Eje 4	+/-210°(450°/s)
	Eje 5	+130°/-130°(375°/s)
	Eje 6	+/- 2700°(550°/s)
Extensión máxima horizontal		1400 mm
Repetibilidad		+/- 0,05 mm
Peso robot		160 kg
Brida portaherramientas		ISO 9409-1-40-4-M6
Motores		AC brushless
Sistema de medición de la posición		con encoder
Potenza totale installata		3 kVA / 4,5 A
Grado de protección		IP65
Temperatura de ejercicio		0 ÷ + 45 °C
Temperatura de almacenamiento		-40 °C ÷ +60 °C
Color robot (estándar)		Rojo RAL 3020
Posición de montaje		Al piso / Techo (Inclinación máx. 45°)

Tabla 3.1 Características y prestaciones SMART5 SiX

Entre algunas de las características más interesantes que caben destacar a parte de las mostradas en la tabla 3.1 se encuentran:

– Predisposición para el montaje de numerosos dispositivos opcionales.

– Elevada capacidad de orientación del pulso en espacios reducidos, gracias a sus dimensiones reducidas.

–Intercambiabilidad entre robots de la misma versión garantizada. Un robot puede ser sustituido rápidamente sin requerir importantes intervenciones para corregir el programa.

– Ausencia de dispositivos específicos para el balanceo de los ejes.

Con todos los modelos y versiones, las cargas declaradas (al pulso y adicionales) pueden ser movidas al máximo de las prestaciones en el interior de todo el volumen de trabajo, gracias a un software específico que, permitiendo alcanzar las máximas velocidades en las aplicaciones en las cuales las carreras del robot sean lo suficientemente amplias, maximiza las aceleraciones en función de la carga declarada y del ciclo.

Cada robot está equipado con un Sistema de Control que satisface las normativas de seguridad de la Comunidad europea y los estándares más importantes. Los cables de conexión entre el control y el robot cuentan con conectores del tipo "plug-in". La predisposición para una serie de opciones, permite utilizar los robots en condiciones de seguridad, respetando las más severas normativas europeas e internacionales.

En las siguientes imágenes se muestran los planos del robot con las medidas generales especificadas para distintas vistas y detalles del robot.

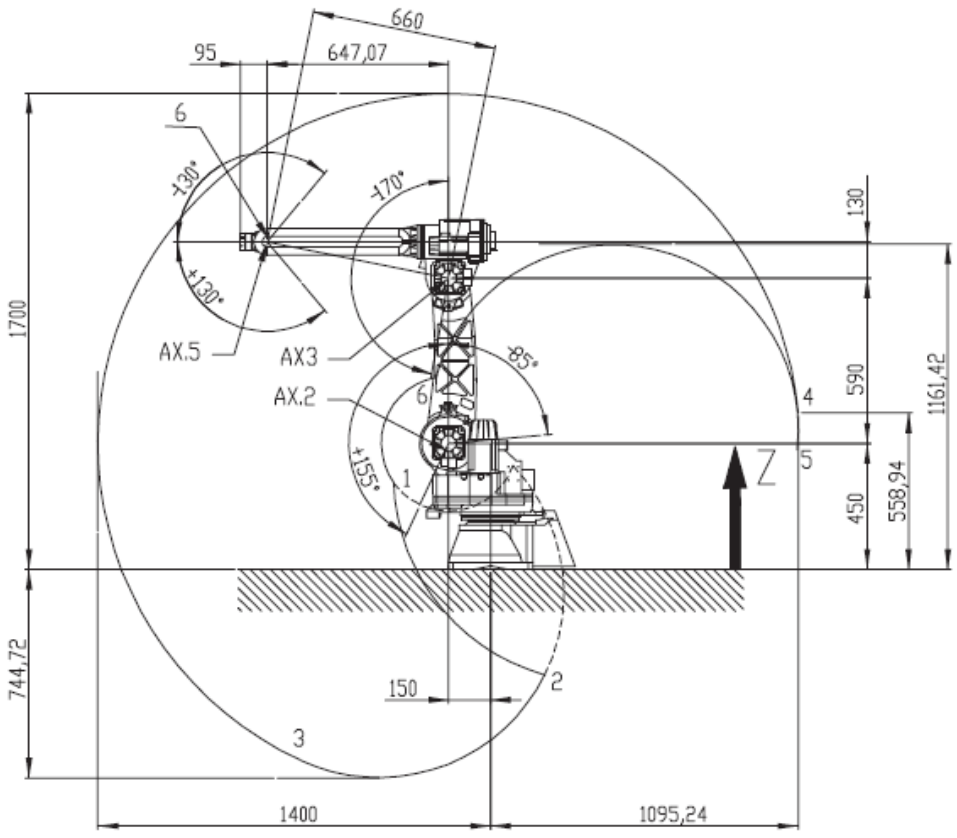


Fig. 3.7 Área operativa vista perfil

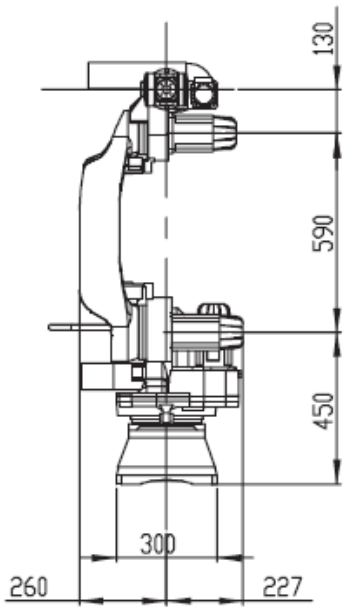


Fig. 3.8 Área operativa vista trasera

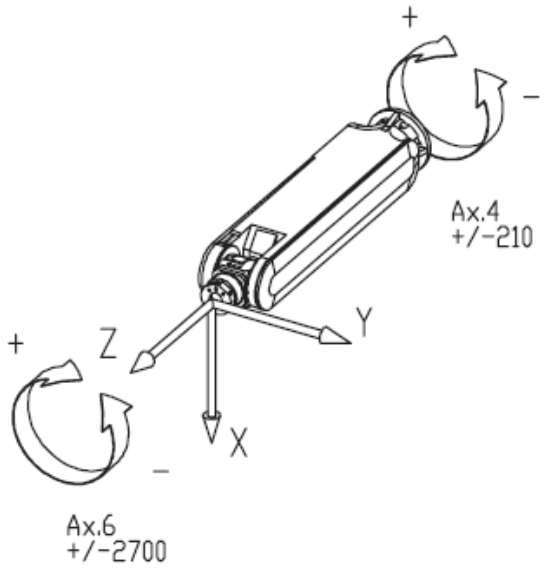


Fig 3.9 Área operativa brazo robot

4.2 Controlador del robot

La Unidad de Control Robot C5G, para desplazar manualmente los robots, crear, modificar y ejecutar programas, modificar movimientos paso a paso, proporcionar funciones de control y monitorización del sistema, prevé el uso de las siguientes interfaz de usuario:

- Terminal de Programación (iTP/WiTP).
- Interfaz en Ordenador Personal (WinC5G).

4.2.1 Terminal de programación.

El Terminal de Programación de la Unidad de Control Robot C5G, permite controlar manualmente los movimientos del robot, programarlo y ejecutar y modificar los movimientos paso a paso. Proporciona funciones de control y monitorización del sistema, además de comprender los dispositivos de seguridad (dispositivo de habilitación y pulsador de emergencia). Es fácil de utilizar y se adapta al uso tanto diestro como zurdo.

Dispone de:

–Pantalla (Display) gráfica a colores TFT de 6,4”; resolución 640x480 pixel.

–Teclas.

–Pulsadores y LED.

–Puerto USB.

Fig. 3.10 Terminal programación

4.2.2 Interfaz en Ordenador Personal

El programa WinC5G representa la interfaz en Ordenador Personal, hacia la Unidad de Control Robot C5G.Agrupa en su interior una serie de funciones como:

- Visualización de los ficheros presentes en la Unidad de Control.
- Posibilidad de editarlos, traducirlos a formato ejecutable (.COD) y ejecutarlos.
- Búsqueda y la visualización de errores.
- Posibilidad de enviar órdenes a la Unidad de Control.
- Transformación de programas ya existentes en función de nuevos puntos de referencia.

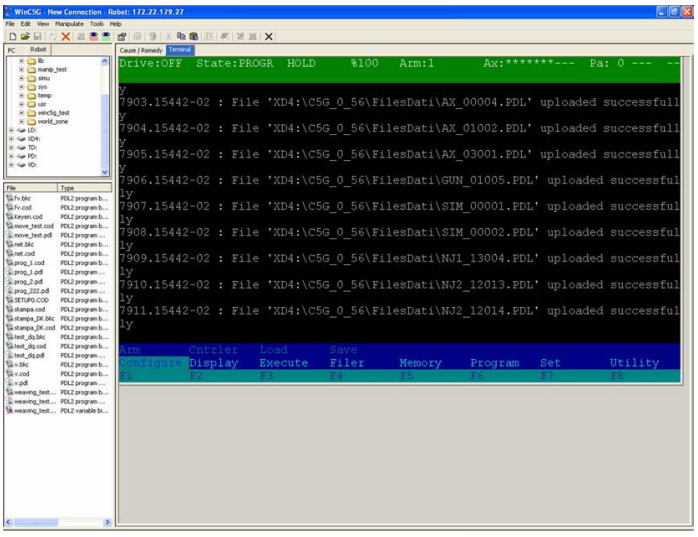


Fig. 3.11 Interfaz WinC5G

4.3 Soluciones alternativas

El proceso que se desea realizar es la aplicación de un spray de acabado al calzado mediante un sistema robotizado. Para ello en la solución propuesta es el propio robot el que transporta la horma de calzado con el zapato hacia la herramienta de aplicado del spray.

En el entorno productivo existe una solución alternativa que sería que el robot fuese el que transportase la pistola del spray. Para ello debería existir un soporte para el calzado similar a una cinta transportadora, de manera que se provea al robot de una cadena en serie de piezas a tratar y este tenga que definir una trayectoria alrededor de la propia pieza. La implantación de esta solución resultaría más costosa, tanto económicamente, ya que habría que incorporar un nuevo elemento a la celda de trabajo (cinta transportadora o similar), como en la ejecución, ya que el manejo por parte del robot de la pistola de aplicación de spray presentaría bastantes problemas. Aun así se podría estudiar su viabilidad. Se procedería primeramente por estudiar la trayectoria alrededor de cada zapato que debería seguir el robot con la pistola incorporada, para continuar estudiando la distancia de aplicación del spray entre zapato y pistola para un buen resultado y el estudio de otros factores del entorno que influirían en nuestro proceso.

Otra solución alternativa referida al ámbito de la simulación consistiría en utilizar otras herramientas de simulación como aquellas desarrolladas por el propio fabricante u otras herramientas como el nuevo simulador de Solid Works. Para todas ellas deberíamos asumir un coste económico en concepto de licencias. El procedimiento a seguir sería similar al propuesto en este proyecto dado que deberíamos introducirnos en el mundo del simulador para conocer sus funciones y posibilidades.

4.4 Justificación de la elección de V-REP

El mercado actual en el ámbito de la robótica industrial ofrece diversas posibilidades en cuanto a la elección de una herramienta de simulación, generalmente asociada a una marca de robots concreta.

La mayoría de las herramientas de simulación comerciales están desarrolladas por la propia marca del robot manipulador con el que se pretende trabajar, teniendo en cuenta que los fabricantes de robots proveen el software de simulación adaptado a sus modelos de controladoras. Este hecho tiene cierta lógica ya que el fabricante es el proveedor de la controladora y es el que mejor conoce todas las funcionalidades que ofrece su dispositivo a la hora de trasladarlas a un software de estas características.

Sin embargo estos software diseñados por el propio fabricante presentan también ciertas desventajas, sobre todo a nivel económico, dado que un simulador de estas características solamente sirve para un modelo de una marca concreta. Si quisiéramos el mismo simulador para otro modelo del mismo proveedor deberíamos volver a pagar por ello. Por ello se plantea la utilización de una herramienta de simulación open source que permita simular el proceso real de fabricación y manejar robots industriales de diferentes fabricantes como puedan ser KUKA, ABB o COMAU.

V-REP-Virtual Robot Experimentation Platform- es una plataforma de simulación suiza que bajo el lema de "Create. Compose. Simulate. Any robot" nos ofrece una amplia gama de posibilidades a la hora de experimentar con robots industriales en un entorno de simulación tal y como se muestra en el vídeo cuyo enlace se adjunta:

https://www.youtube.com/watch?v=gBYqOBdIcaY#t=203

Este software ofrece un desarrollo rápido de algoritmos, simulaciones de automatización de fábrica, prototipado rápido y verificación, monitoreo remoto, seguridad de doble control, etc.
No hemos de perder de vista que la principal ventaja de este simulador es que es de código abierto, es decir, la licencia del software es de dominio público, con lo cual con la descarga desde la página web de la versión educacional ya vamos a poder realizar un análisis previo de nuestro sistema robótico.

Además el simulador de robot V-REP ya cuenta con el entorno de desarrollo gráfico integrado. Este se basa en una arquitectura de control distribuido: cada objeto/modelo puede ser controlado individualmente a través de una secuencia de comandos incrustada, un plug-in, un nodo de ROS, un cliente de API remota o una solución personalizada. Esto hace que V-REP muy versátil e ideal para aplicaciones multi-robot. Los controladores pueden ser escritos en C / C + +, Python, Java, Lua, Matlab, Octave o Urbi.

Por último otra de las ventajas en la utilización de este simulador que ha aparecido durante el desarrollo de este proyecto es la existencia de un foro en el cual se da soporte técnico ante cualquier dificultad o incidencia que encontremos en el manejo del software.

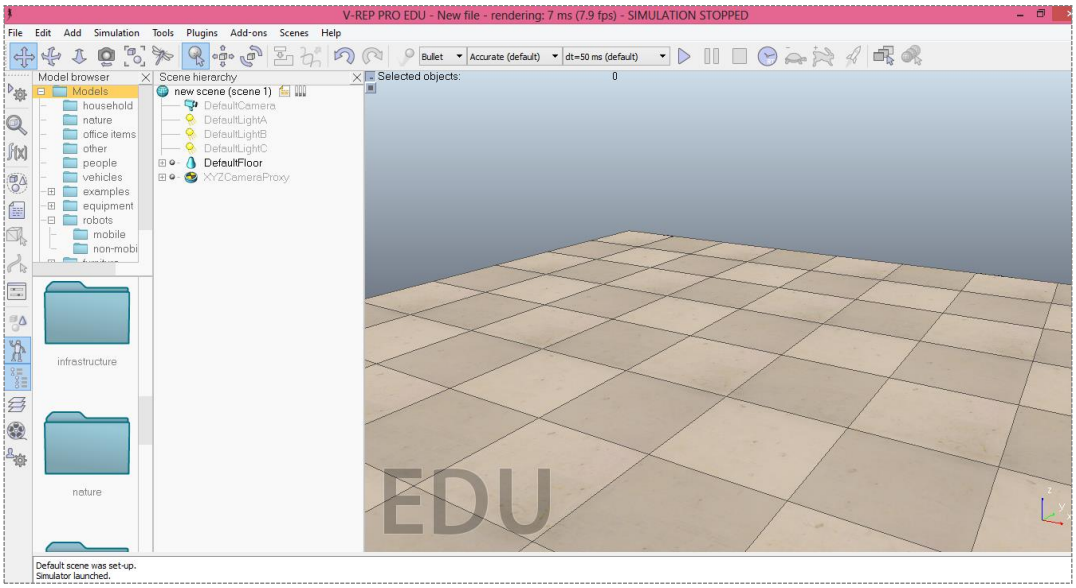


Fig. 4.1 Entorno gráfico inicial del simulador V-REP

4.5 Acciones previas

Previamente al trabajo de simulación es necesario un trabajo de diseño del robot que se quiere simular. Para ello se ha utilizado la herramienta Solid Works, dado que V-REP no es un software destinado a realizar esta aplicación. Se procede con el mismo al diseño de las piezas y posterior ensamblado con las pertinentes relaciones de posición para alcanzar la configuración deseada.

Partiendo del diseño ya realizado de cada pieza se obtiene en primer lugar un ensamblado como el que se muestra en la figura 4.2.

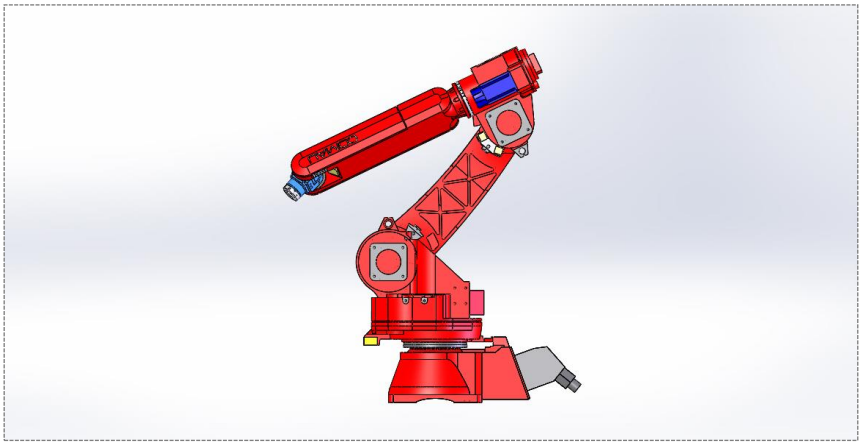


Fig.4.2 Vista del ensamblado previo obtenido en SW

En principio el diseño exportado a V-REP fue el mostrado anteriormente. Más tarde se detectó que al no estar los ejes en la posición definida como inicial en las especificaciones del robot, a la hora de poner las restricciones de giro a los ejes no se partía de una posición correcta, lo que hacía menos preciso el diseño y más sensible a error la simulación a la hora de la resolución cinemática. Se optó por realizar una corrección del diseño inicial y repetir el proceso de ensamblaje para la posición correcta de los ejes, mejorando de este modo también la apariencia estética del resultado, que se muestra en la figura 4.3.

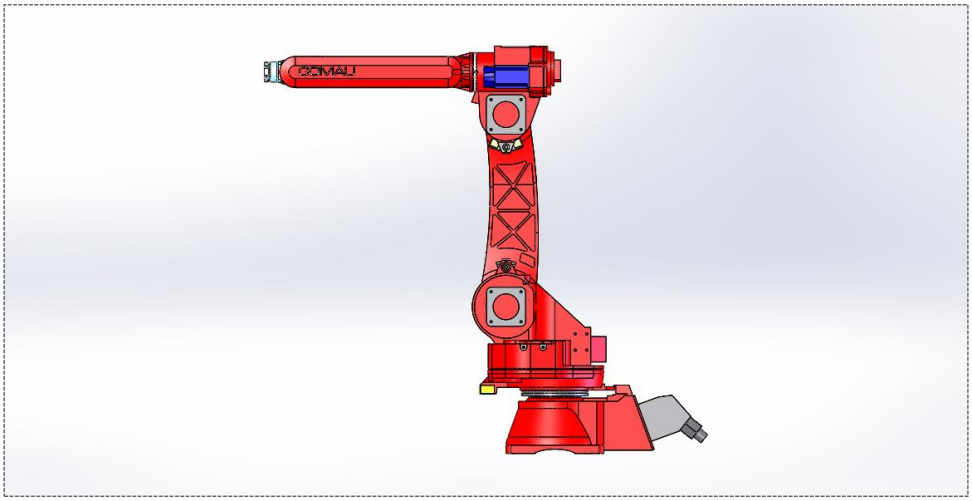


Fig. 4.3 Vista del ensamblado final exportado a la herramienta de simulación.

5 SIMULACIÓN DEL SISTEMA ROBÓTICO

5.1 Importación de archivos

Una vez conseguido el diseño correcto del robot se procede a exportar las piezas desde el SolidWorks al V-REP. El proceso a seguir es:

- Guardar el ensamblaje montado en SW con el formato .stl admitido por nuestro simulador. Se generará un archivo .stl para cada pieza del ensamblaje.

- Una vez dentro de V-REP se procede de la siguiente manera: Menú desplegable File→Import→Mesh y selección de los archivos anteriormente guardados.

Es recomendable ir importando cada pieza en orden, es decir, primero la base y seguidamente cada uno de los eslabones y brazos. Se ha de prestar atención a la importancia de haber realizado el ensamblaje el SW. Si se hubiese importado la geometría de las piezas directamente en el formato admitido por el simulador se debería haber colocado cada una en su posición y orientación correspondiente y con sus relativas relaciones de posición. V-REP no es un programa de diseño y esta tarea resulta bastante tediosa a no ser que se cuente con una tabla con las coordenadas y orientaciones ya proporcionada previamente. No se contaba con ella en el caso tratado. Al haber realizado el ensamblaje en el programa de diseño gráfico simplemente se importan las piezas de dicho ensamblaje una a una y estas se colocarán en el espacio en la posición y orientación con que fueron exportadas.

Por último se ha de prestar atención al cuadro de dialogo de importación en el cual se elige la unidad de importación (mm, o las correspondientes en otro caso) y cuál será la orientación del sistema de referencia de la pieza (con el eje Y apuntando hacia arriba o el Z).

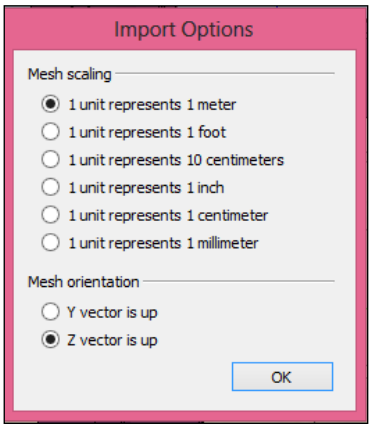


Fig. 5.1 Cuadro de diálogo de importación

Por conveniencia, tras importar cada una de las piezas, se accederá al menú: *Edit→ Reorient bounding box→ with reference of world.* De esta forma la orientación de las piezas estará en la configuración que se denotará como *HOME*, siempre cero.

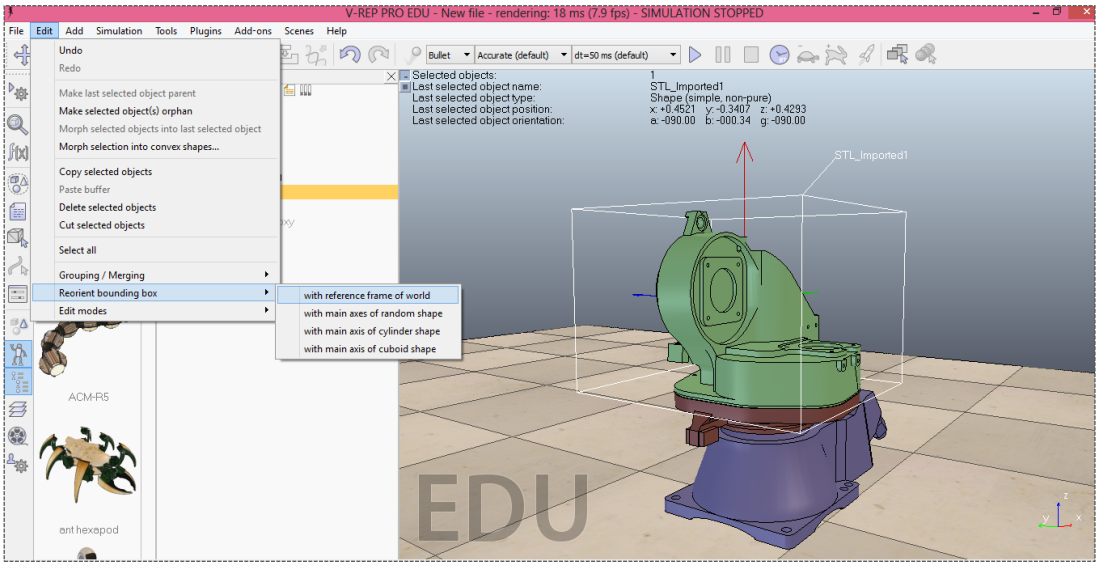


Fig. 5.2 Proceso de importación y orientación de los ejes respecto a la configuración *HOME*

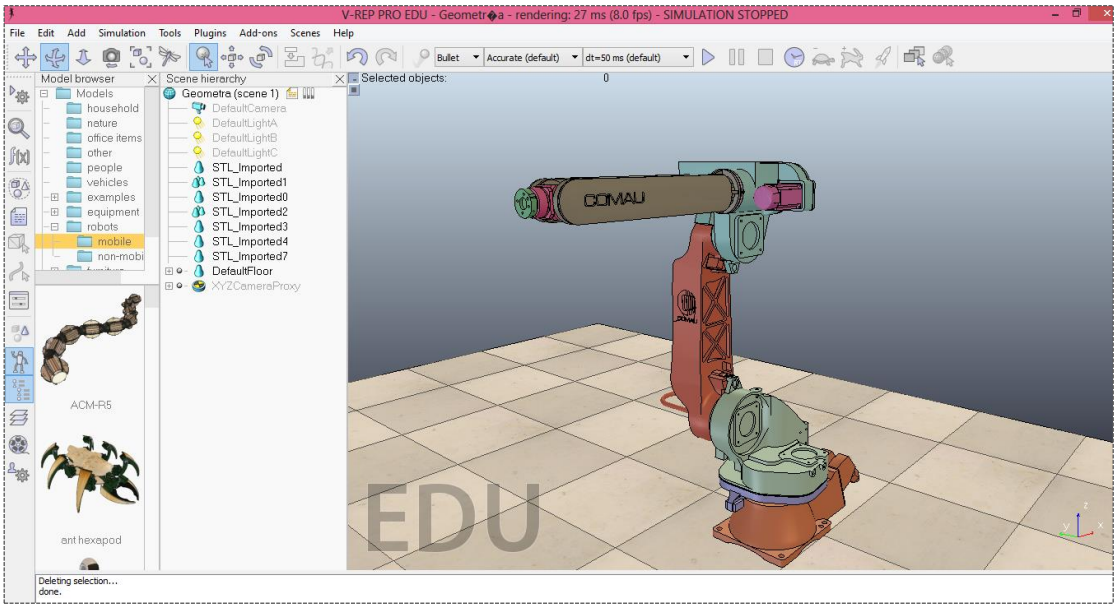


Fig. 5.3 Resultado del proceso de importación de piezas

En esta última imagen es observable como al realizar la importación aparece un elemento en la jerarquía de la escena o *"Scene hierachy"* nombrado como *STL_Imported* correspondiente a cada una de las piezas importadas. Se le puede cambiar el nombre a cada pieza simplemente haciendo doble click sobre el mismo y reescribiendo.

5.2 Diseño de la celda de trabajo.

Para la realización del diseño se tendrán en cuenta tanto las limitaciones de la zona operativa del robot como la disposición de los elementos que deberán componer la celda para la aplicación industrial descrita.

De este modo se definen varias zonas dentro del sistema, por las cuales los objetos manipulados van a ser trasladados según el estado del proceso robótico:

-Posición del robot manipulador. Se definirá en esta posición el origen de coordenadas para el sistema robotizado.

-La zona de soporte de calzado. Mediante un soporte giratorio se proveerá de la horma del zapato al robot.

-La zona de trabajo. Se realizará un proceso determinado con los objetos trasladados a la zona por el robot manipulador.

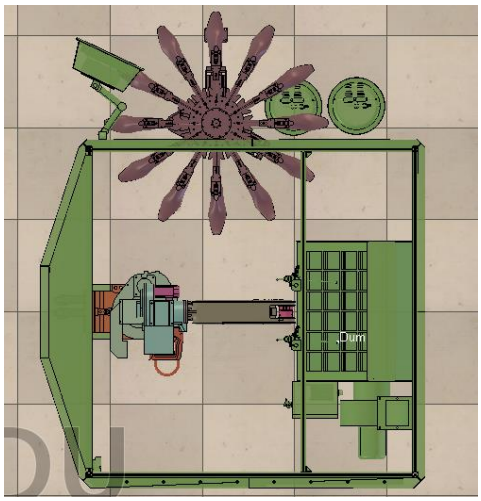


Fig. 5.4 Diseño celda vista superior

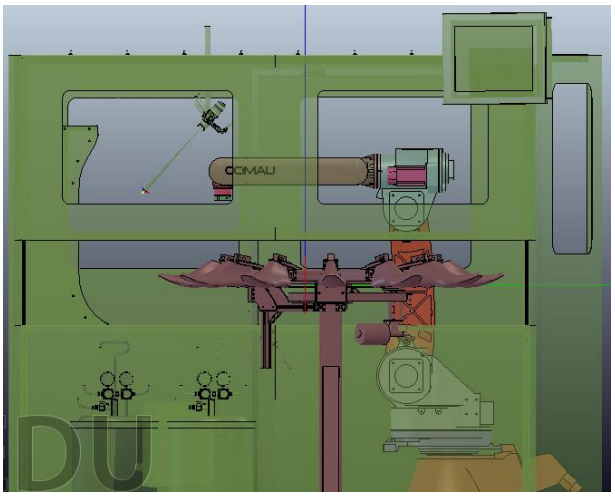


Fig. 5.5 Diseño celda vista lateral

Fig. 5.6 Imagen de la celda de trabajo real

5.3 Definición de las articulaciones

Una articulación es un objeto que tiene al menos un grado intrínseco de libertad. Las articulaciones se utilizan para construir mecanismos y para mover objetos, luego su utilización y correcta definición resultan de vital importancia.

Para añadirlas a la escena se accede al menú desplegable *Add→Joint*. Se ofrecen tres posibilidades: articulación de revolución, articulación prismática o articulación esférica. Es elegida la articulación de revolución o rotación. Una vez añadida se debe colocar en la posición y orientación exacta. Al no conocerla dicho parámetros se procede de la siguiente manera:

1. Identificación de la pieza que está directamente vinculada a la articulación.
2. Se aísla la pieza en una escena diferente (copiando la pieza de la escena actual, se crea una nueva escena y se pega).
3. Accediendo al modo de edición de triángulos o edición de vértices se simplifica la pieza hasta obtener con un tubo (o similar) donde estará situada la articulación. Se sale del modo de edición.
4. Ahora el resto de la pieza original debe tener la misma posición y orientación que el conjunto que sea agregado posteriormente.
5. Se añade una articulación.
6. Seleccionada la articulación se accede al cuadro de diálogo de traslaciones y orientaciones. Haciendo clic en *'Apply to selection'* para la parte de orientación y posición de la sección de coordenadas la articulación adopta la posición correcta.

En las siguientes imágenes se muestra la secuencia del procedimiento:

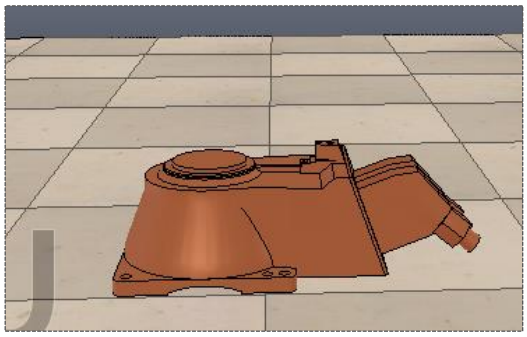


Fig.5.7 Pieza aislada en nueva escena.

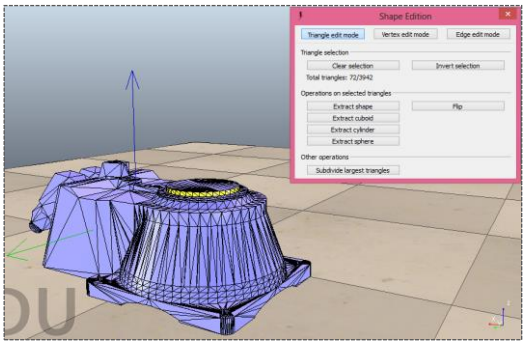


Fig. 5.8 Modo edición de triángulos.

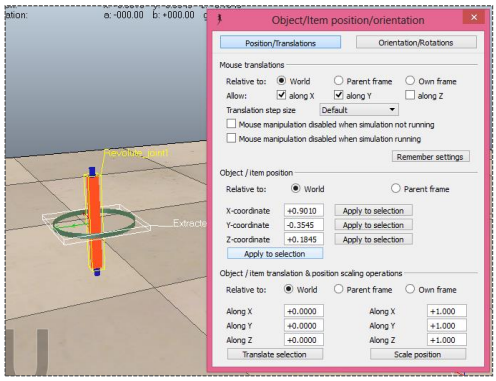


Fig. 5.10 Posicionamiento articulación

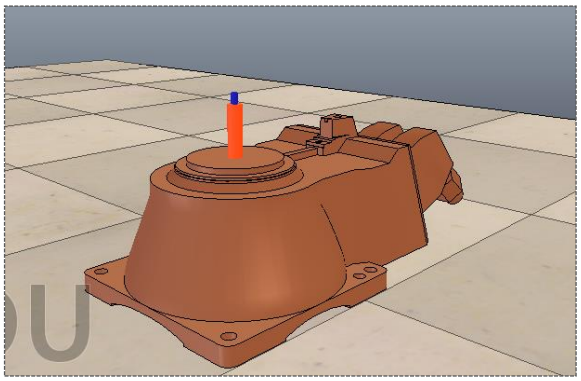


Fig. 5.11 Resultado para pieza base

El conjunto ahora tiene la posición y orientación correcta, y puede ser copiado / pegado en la escena original.

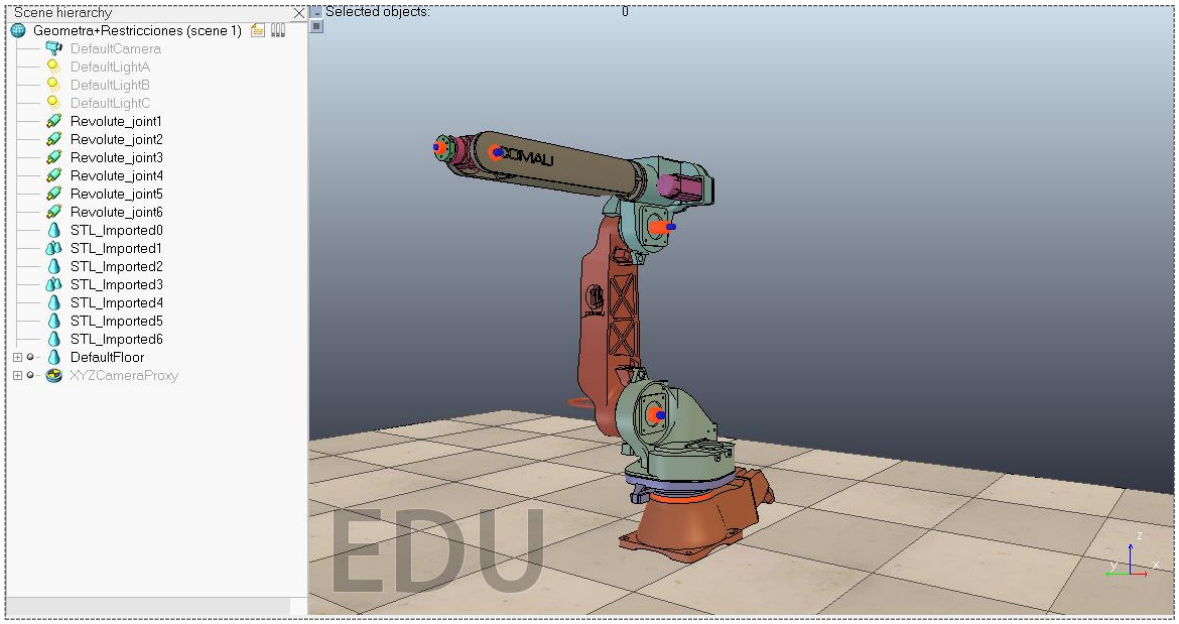


Fig. 5.12 Resultado final en escena del posicionamiento de las articulaciones.

Como se puede observar, en la jerarquía de la escena han aparecido las articulaciones añadidas. Notar que en el brazo del robot hay una articulación no visible, pues su eje coincide con el eje del brazo.

Una vez se tienen definidas las articulaciones del robot, es necesario imponer ciertas restricciones físicas, ya que habrá ciertas configuraciones que las articulaciones del robot real no podrán alcanzar. Del libro de especificaciones del robot se obtienen las siguientes restricciones referentes a limitaciones de giro de las mismas:

	Ángulo máximo	Ángulo mínimo	Rango
Eje 1	+170º	-170º	340º
Eje 2	+155º	-85º	240º
Eje 3	0º	-170º	170º
Eje 4	+210º	-210º	420º
Eje 5	+130º	-130º	260º
Eje 6	+2700º	-2700º	5400º

Tabla 5.1 Restricciones articulaciones

Para imponer estas restricciones se accede al diálogo de propiedades generales de cada articulación, se deselecciona la opción de "*Position is cyclic*" introduciéndose los valores correspondientes.

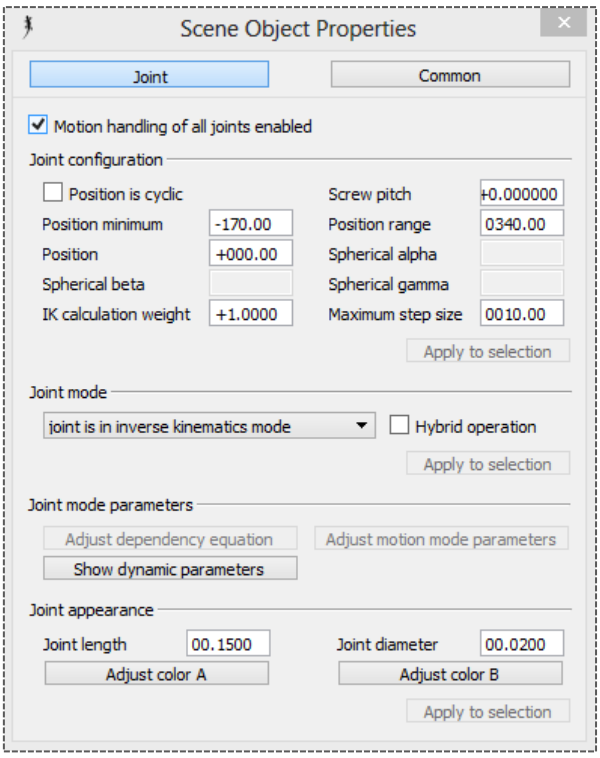


Fig. 5.12 Introducción de las limitaciones de las articulaciones

5.4 Control cinemático inverso del robot

El módulo de cálculo de V-REP cinemática inversa (IK) es muy potente y flexible. Permite manejar virtualmente cualquier tipo de mecanismo en el modo de cinemática inversa (modo de IK) o el modo de cinemática directa (modo de FK). El problema de la CI puede ser visto como el de encontrar los valores comunes correspondientes a alguna posición y/u orientación específica de un elemento de cuerpo dado. Para un robot, por ejemplo, el problema sería encontrar el valor de todas las articulaciones en el mismo dada la posición (y / u orientación) del extremo o último eslabón. El problema inverso - la búsqueda de la posición del eslabón final dados los valores conjuntos - se conoce como problema de FK ("forward kinematics") y, a menudo se percibe como una tarea más fácil que IK.

En la mayoría de los casos la solución del problema cinemático inverso no es única, existiendo diferentes soluciones que posicionan y orientan el extremo del robot del mismo modo. Se utiliza el módulo de cálculo IK de V-REP para resolver la cinemática inversa del manipulador a partir de algoritmos genéricos que resuelven el problema de forma iterativa.

Para cada articulación, ha sido activado el modo de funcionamiento de *"joint is in inverse kinematic mode"* en su diálogo de propiedades. Esto indicará a V-REP que será él quien calcule el valor de cada articulación con el fin de resolver el mecanismo del robot.

Se deben establecer las relaciones de parentesco pertinentes entre eslabones, articulaciones y el resto de objetos de la simulación a fin de construir la cadena cinemática, yendo desde la base del robot hasta el último eslabón.

Esto se consigue mediante la opción menú desplegable *Edit → Make last selected object parent* o simplemente arrastrando en la jerarquía de la escena un objeto sobre otro. Se ha empezado desde el final de la cadena haciendo al eslabón "SmartSix_link6" pariente de la articulación "Revolute_joint6" y procediendo sucesivamente hasta que la cadena cinemática este construida. Se debe obtener una jerarquía como la siguiente:

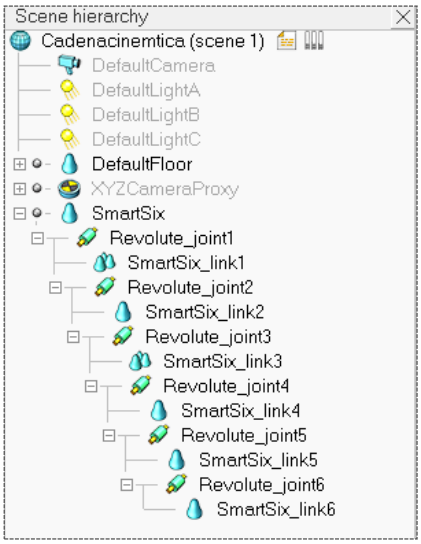


Fig. 5.13 Relaciones de parentesco entre elementos de la escena

Se procede en este momento a la definición de la tarea cinemática para el manipulador. En V-REP, una tarea IK requiere la especificación de al menos los siguientes elementos:

- Una cadena cinemática descrita con un dummy tip y un objeto "base".
- Un dummy target "objetivo" que el tip se verá obligado a seguir.

El objeto base es el primer eslabón ("SmartSix"), al cual se le ha de activar la opción de *Object is model base* en su diálogo de propiedades. Tras activarlo aparecerá justo a la izquierda en la jerarquía de la escena un icono identificativo de los objetos que son modelos base que se puede apreciar en la figura 5.13. Esta acción permitirá tratar a todo el robot como un objeto único.

Es necesario crear dos nuevos objetos tipo "dummy". Los "dummies" son objetos cuyo fin es el de posicionar o relacionar determinados objetos entre sí y que se utilizan para cerrar el mecanismo del robot. Al primer dummy se le nombrará como *"smartsix_tip"* y al segundo *"smartsix_target".* Se les asignará la posición y orientación del último eslabón de nuestra cadena cinemática.

Una vez creados se liga el dummy tip *"smartsix_tip"* al último eslabón *"SmartSix_link6"* haciéndolos parientes. Por último, para informar a V-REP de que los dummies tip y target forman un par de resolución cinemática:

- En el diálogo de propiedades del dummy tip se asigna en el apartado *dummy-dummy linking* el *dummy* target como dummy asociado.
- En el apartado Link type marcamos la opción IK, tip-target.

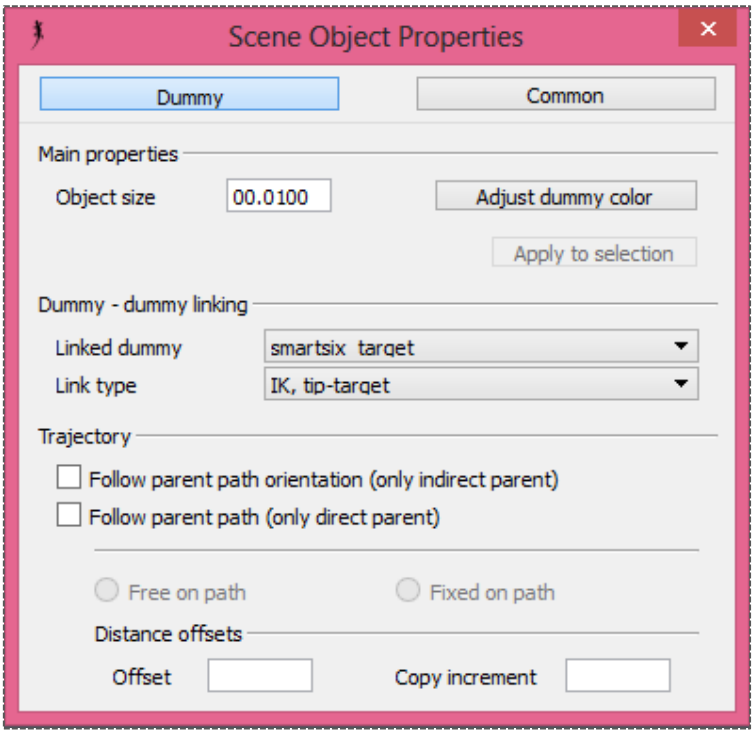


Fig. 5.14 Cuadro de propiedades de los objetos dummy

Todos los elementos para la definición de la tarea cinemática inversa están listos, y sólo tenemos que registrar la tarea como un grupo IK.

Se abre el diálogo de cinemática inversa y se hace clic en Agregar nuevo grupo IK. Un nuevo elemento aparece en la lista de grupos IK: "IK_Group". Se selecciona ese elemento. Se hace clic en *Edit IK elements* para abrir el diálogo de elementos IK. Junto a *Add new IK element*, se selecciona el dummy tip en el cuadro desplegable. A continuación, se hace clic *en Add new IK element*, lo cual hace que se agregue el elemento IK que aparece en la lista. Más abajo, se indica "SmartSix" como la base. Por último, asegurarse de que todos los ítems en la sección restricciones están marcados (marcar Alfa-Beta y Gamma) ya que se quiere que el punto ficticio "tip" siga a "target" en la posición y orientación.

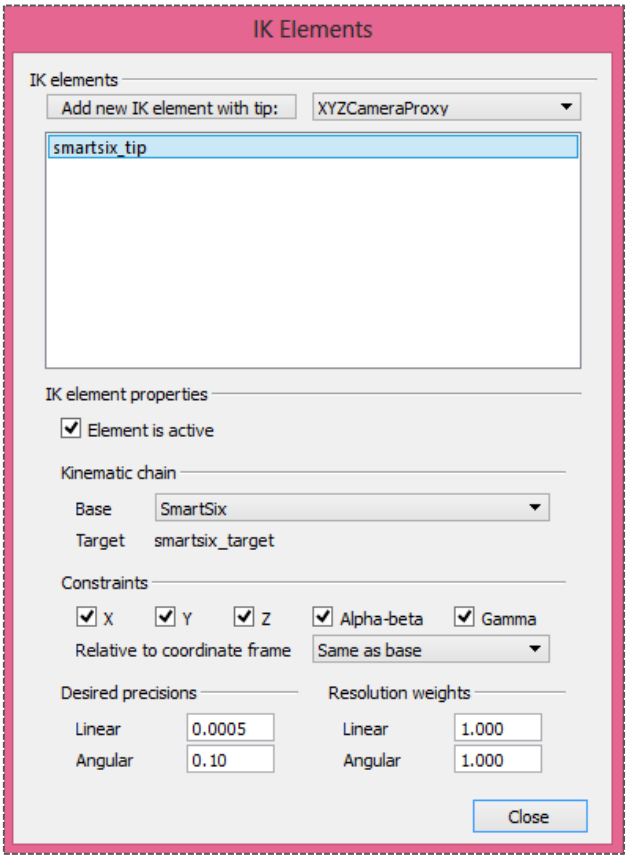


Fig. 5.15 Registro de la tarea de CI

La tarea cinemática está realizada. Para la comprobación de la misma se pulsa *Run the simulation* y se prueba a mover el objeto target. El robot entero se debe posicionar para alcanzar la posición y orientación definida por este objeto. Cuando se aleja demasiado se comprueba como la cadena cinemática se rompe. Esto ocurre cuando una configuración singular no es alcanzable.

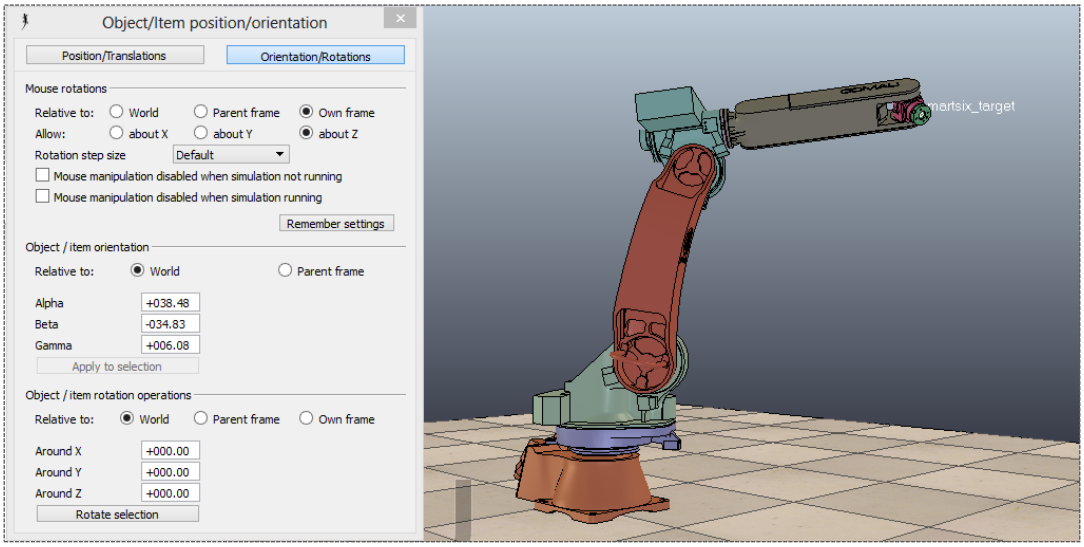


Fig. 5.16 Robot alcanzando posición y orientación definida por dummy target.

Cuando se llega a una configuración singular o no alcanzable el comportamiento del robot en la simulación se torna inestable al romperse la cadena cinemática. Hay soluciones para este comportamiento, en lugar de utilizar el método Pseudo inverso de resolución se cambia a DLS, de este modo la resolución cinemática para puntos singulares es más estable. Además podemos ajustar el coeficiente de amortiguación. Básicamente a mayor amortiguación más estable es la resolución, pero a su vez también es más lenta. Se pueden obtener las ventajas de ambos métodos de resolución definiendo dos grupos IK uno amortiguado y otro sin amortiguación, especificando una resolución condicional.

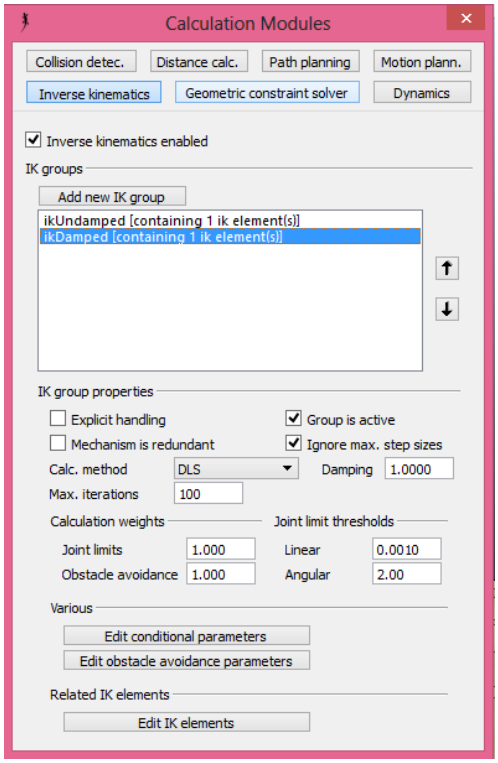


Fig. 5.17 Resolución cinemática con dos grupos IK

A continuación se procede a añadir una "esfera de manipulación" que se va a utilizar para manipular la posición / orientación del último eslabón del robot. De este modo utilizaremos la posición de la esfera como si fuese el terminal final de nuestra cadena. Haciendo clic en el menú *Add→Primitive Shape→ Sphere* añadimos la geometría de la esfera a la escena con los valores de 0.1 para el *x-size, y-size y z-size* . Se desactiva la opción *Create pure* y se hace clic en Ok. Se ajusta la posición de la esfera a la misma posición que "smartsix_target" usando el diálogo de coordenadas y transformaciones. La esfera ahora aparece en el extremo del manipulador. A continuación, se hace a la esfera padre de *"smartsix_target".* De este modo al ejecutar la simulación deberemos ser capaces de cambiar la configuración del manipulador moviendo la esfera de manipulación.

En las propiedades del objeto de la esfera se desmarcarán todos los ítems en la sección de propiedades especiales del objeto ya que la esfera manipulación realmente no pertenece al manipulador, es un elemento de interfaz de usuario. Finalmente se hace padre a *"SmartSix"* de *"manipSphere".*

5.5 Planificación de la trayectoria

Definir el movimiento de un robot implica controlar dicho robot de manera que siga un camino preplanificado. El módulo de planificación de trayectorias de V-REP permite crear todo tipo de trayectorias a partir de puntos de control y curvas de Bézier y que, posteriormente, mediante ciertos códigos de programación, el robot sea capaz de seguir estas trayectorias.

Se procede a describir el movimiento deseado del manipulador como una secuencia de puntos en el espacio (con posición y orientación). Dicha secuencia de puntos o trayectoria ha sido generada anteriormente de modo que se comprobará que el robot es capaz de alcanzar cada uno de los puntos de dicha trayectoria sin encontrar ninguna indeterminación en la configuración de sus articulaciones.

Pueden darse dos situaciones. La primera de ellas es que sea el propio robot el que transporte la herramienta y siga una trayectoria alrededor de una pieza fija para realizarle cualquier tipo de procesado. La segunda situación es que el robot contenga la pieza a procesar de modo que sea la propia pieza la que describe la trayectoria a través de la herramienta fija. En el caso de la aplicación del spray el robot transporta el zapato introducido en su horma, el cual deberá desplazarse alrededor de la pistola de spray situada fija en la celda de trabajo. Intuitivamente se trata de dos situaciones similares, pero a la hora de programar el movimiento del robot dista mucho un caso del otro.

Para una mejor comprensión de cada una de las situaciones se adjuntan dos videos que reflejan cada una de ellas, en el primer video el primer caso y en el segundo vídeo el caso que se tomará como referencia:

https://www.youtube.com/watch?v=7pKyPNpYsr8

https://www.youtube.com/watch?v=wt_cyo8si5g

Se procede en primer lugar a la definición de la trayectoria. Se cuenta con un archivo previamente generado de dicha trayectoria que consta de las coordenadas (x,y,z) y orientaciones en ángulos ZXZ de Euler de cada punto a seguir. Del archivo original se extrae la información de utilidad que se incluye en el "Anexo trayectoria" (apartado 7.1.1).

Se accede al menú Add→Path→Segment type y aparece una trayectoria rectilínea que consta de dos puntos. Se accede a la edición de la trayectoria con el fin de generar tantos puntos como tiene la trayectoria deseada y modificar la posición y orientación de dichos los mismos.

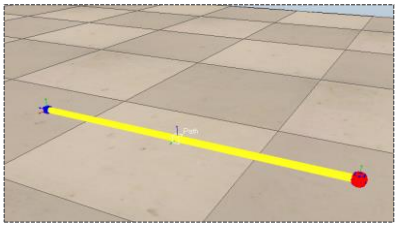


Fig. 5.18 Trayectoria Segment type inicial

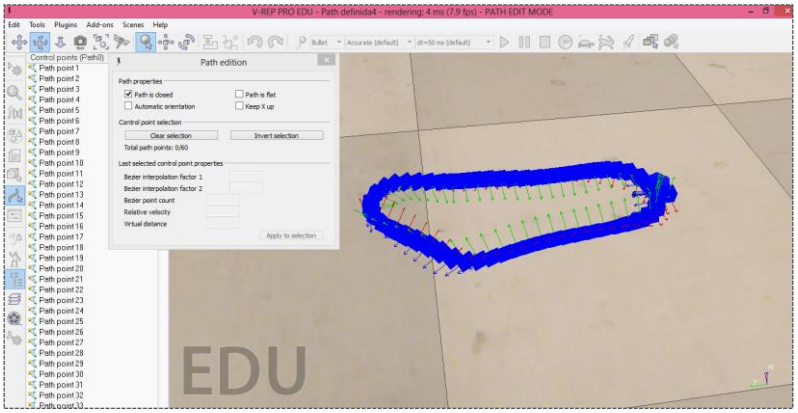


Fig. 5.20 Diálogo de edición de trayectorias con puntos introducidos

Una vez obtenida la trayectoria deseada se ha de proceder a ejecutar el movimiento del robot sobre dicha trayectoria. Para ello se empieza analizando un caso más sencillo. Se hace que el robot siga la trayectoria con la misma posición de los puntos en el espacio y a su vez la misma orientación para cada uno de ellos. Como podemos observar en la figura 5.20 cada punto varía ligeramente su orientación respecto al anterior. Luego se comienza esta tarea analizando en primer lugar el caso más sencillo.

La trayectoria a seguir en esta primera prueba seria la mostrada a continuación:

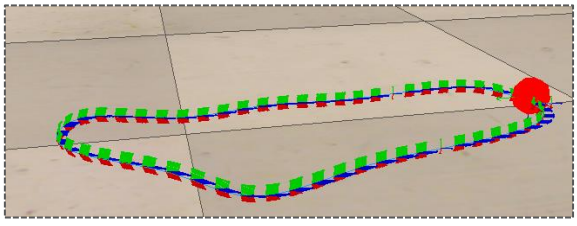


Fig. 5.21 Trayectoria con orientación de los puntos constante

Se aprecia como en este caso la orientación de cada punto de la trayectoria permanece constante. Se introduce la misma en la escena principal con nuestro robot, la horma y el zapato (pieza a procesar) ya acoplada (Fig.5.21).

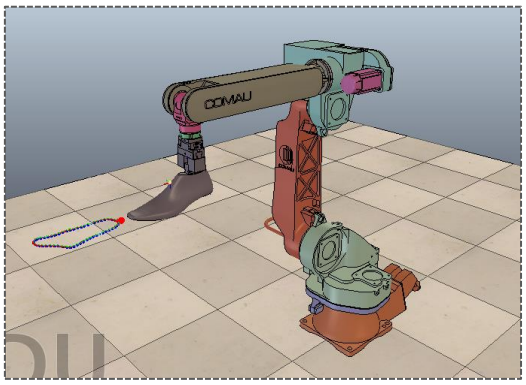


Fig. 5.21 Análisis de la primear trayectoria

Para mover el robot a lo largo de una trayectoria se debe modificar de manera acorde la localización del dummy que cierra la cadena cinemática *("smartsix_target").* Se puede hacer a través de instrucciones específicas de posicionamiento o dejar que sea el propio V-REP quien mueva el punto sobre la ruta para una velocidad dada. Para el caso estudiado deberemos recurrir a la primera opción. Nótese que, como en el caso de la resolución cinemática, vuelve a ser necesario introducir elementos dummy tanto para definir la posición y orientación con la que el robot seguirá la trayectoria como para indicar en qué punto queremos que el robot ejecute la misma.

Procedemos a programar el movimiento mediante un child script asociado a nuestro robot. Un child script representa un código o programa escrito en lenguaje de programación Lua que permite el manejo de una función particular en una simulación pequeña. En el "Anexo Trayectoria" (apartado 7.1.2) se encuentra el código de programación que se ha utilizado para hacer que el robot siga la misma. Una vez definidos todos los elementos anteriores se procede a la simulación del movimiento y vemos como nuestro robot efectivamente se mueve describiendo la trayectoria propuesta como se esperaba.

Una vez se comprueba el funcionamiento de la simulación con el código de programación propuesto, se realiza otra prueba previa con otra trayectoria. Esta vez la trayectoria cambia al llegar a la puntera del zapato la orientación del eje normal a la suela. Se puede observar la siguiente modificación en la trayectoria en la figura 5.22.

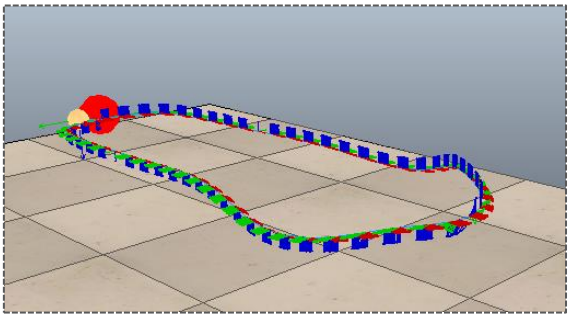


Fig. 5.22 Trayectoria con cambio de sentido del eje Z

Se procede del mismo modo anterior y se comprueba su funcionamiento. El robot es capaz de seguir la trayectoria sin encontrar ningún punto singular. En esta simulación además se pretende que la horma siga la trayectoria orientada horizontalmente, es decir, se ha girado 90º en sentido horario el plano en el que se recorrerá, la trayectoria. Esto se consigue aplicando dicho giro al *"Dummyposicion"* que es el punto de referencia donde se ejecutará la trayectoria. El resultado de esta operación de aprecia en la figura 5.23.

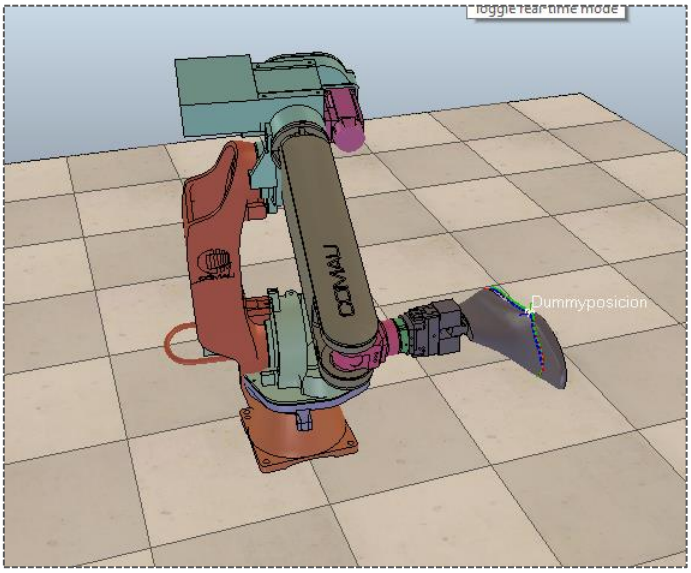


Fig. 5.23 Robot siguiendo trayectoria horizontalmente.

Una vez comprendido el funcionamiento del simulador en lo referente a seguimiento de trayectorias, así como establecidas las pertinentes relaciones de parentesco entre los nuevos elementos introducidos en la escena (dichas relaciones se encuentran adjuntas en el "Anexo Trayectoria" en el punto 7.1.3 de esta memoria) y la influencia de su posicionamiento y orientación en el resultado de la simulación, deducida del análisis de los casos más sencillos, se procede a introducir la trayectoria real descrita al principio de este apartado.

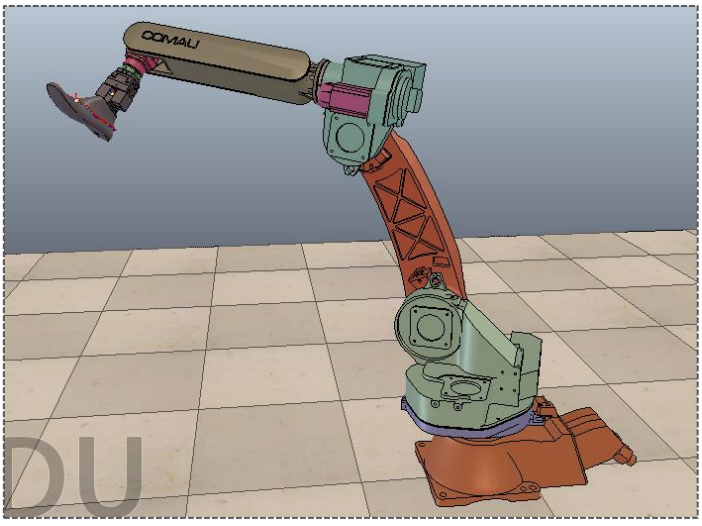


Fig. 5.24 Smart5 SiX describiendo trayectoria real en el punto de aplicación del spray.

En principio se prueba posicionando el dummy posición (punto donde se ejecutará la trayectoria) en un punto arbitrario. Posteriormente dicho punto se posicionará en el espacio donde está situado en la celda el punto de aplicación del spray. Se comprueba que en cualquiera de los dos casos la simulación continua siendo correcta, es decir, no se obliga al robot a alcanzar puntos a los que físicamente no puede acceder.

5.6 Detección de colisiones

V-REP puede detectar colisiones entre dos entidades de una manera muy flexible. El módulo de colisiones sólo permite detectarlas, sin embargo, no reacciona directamente ante ellas (para ello se debería proceder al análisis del módulo dinámico).

Como una primera aproximación a esta herramienta, se trata de detectar si existe colisión entre el robot y los elementos colindantes o entre los propios eslabones del mismo. Se describirá el procedimiento a seguir para poder realizar dicho análisis que resulta de gran utilidad para definir si en el proceso real el robot colisionara con alguna entidad.

Partiendo de la resolución cinemática se va a registrar un objeto colisión que debe detectar las colisiones entre el manipulador y su entorno. Lo que se pretende es que cada forma individual en el manipulador sea capaz de detectar una colisión con el entorno.

En primer lugar hay que definir una colección para nuestro robot. Se abre el diálogo de colección con *Menú* → *Tools* → *Collections* o haciendo clic en el botón de la barra de herramientas apropiado. Se selecciona "SmartSix" y a continuación se hace clic en Agregar nueva colección. De este modo se añade una nueva colección vacía.

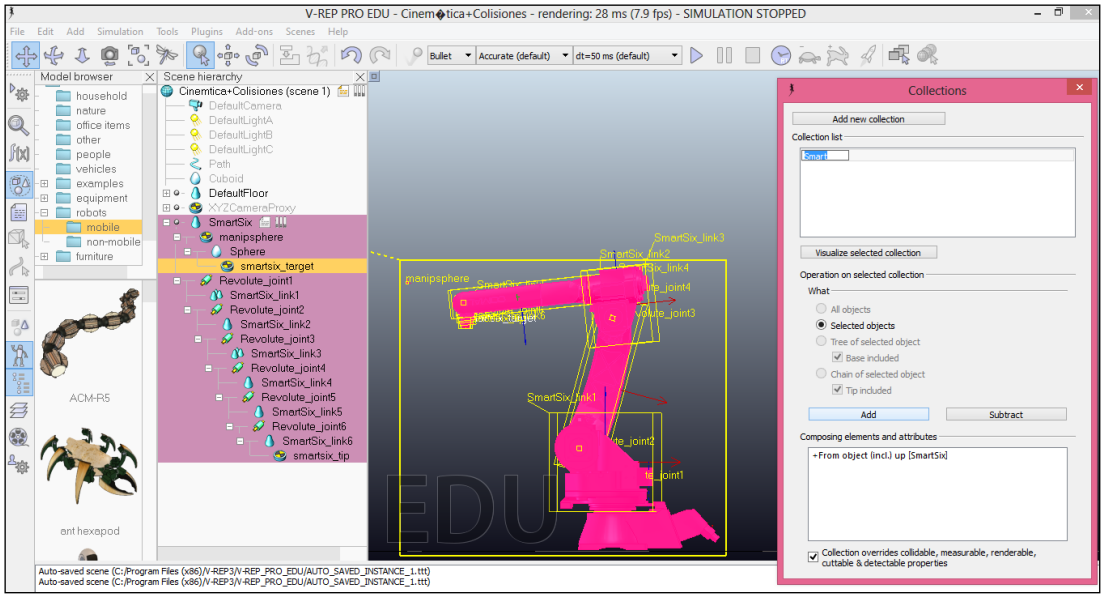


Fig. 5.25 Creación de la colección Smart con todos los elementos del robot seleccionados

Ahora se ha de definir el contenido de la colección haciendo clic en *Add* (en todo momento "SmartSix" debe estar seleccionado). Se observa cómo cambia el contenido de la colección. Ahora se selecciona el elemento de la colección que se acaba de agregar, a continuación, con un clic en *Visualize selected collection* todos los objetos que componen el manipulador se tornan en un color rosado en la escena. Se cambia el nombre de la colección a "Smart".

Una vez que la colección "Smart" está definida, se puede registrar un objeto de colisión. Abriendo el diálogo de colisiones *Calculation modules→Colision detec*. se hace clic en *"Add new collision object"* y se especifica el par de ítems siguiente: *"(Collection:Smart- All other entities) "*. Esto agrega un nuevo objeto de colisiones al que se le puede cambiar el nombre en la lista con un doble clic.

La tarea de detección de colisiones está realizada. Para comprobarlo se crea una nueva forma "Cuboid" (*Add→Primitive shape→Cuboid*) de dimensiones cualesquiera. Se crea también una trayectoria, en este caso "Circle type" y se hace el robot al seguir la trayectoria circular colisione con el cubo(los detalles de cómo hacer que el robot siga esta trayectoria se encuentran en el Anexo Colisiones, apartado 7.2).

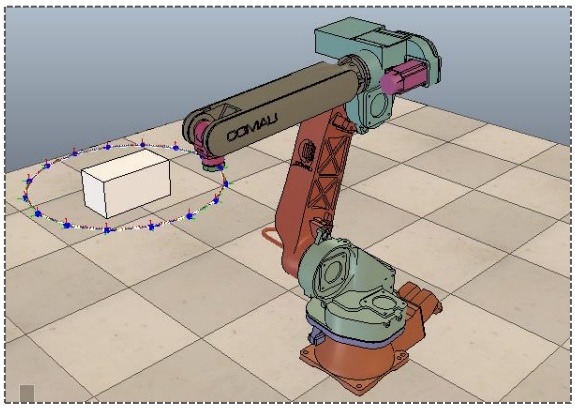


Fig. 5.25 Comprobación del módulo de colisiones creado

Ponemos la simulación en marcha y vemos como el robot cambia de color a rojo cuando detecta que ha colisionado con un objeto de la escena, en este caso, el cuboide que se había creado para ese fin.

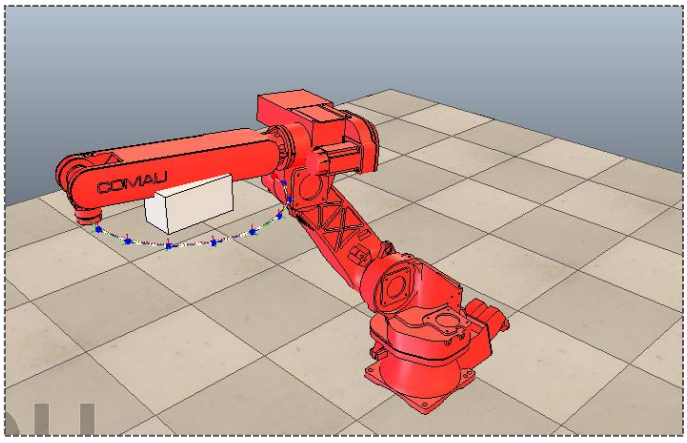


Fig. 5.26 Detección de la colisión con un objeto en la escena

5.7 Controles de usuario

V-REP ofrece una interfaz de usuario personalizable de gran alcance para las simulaciones. Esta interfaz, puede tomar la forma de diálogos que se integran en botones, editables, barras deslizantes o etiquetas. Cualquier acción en las interfaces de usuario personalizadas se informa como un mensaje que puede ser interceptado por llamadas a la API apropiadas. Esto permite la personalización de una simulación en gran medida.

Se pretende crear una interfaz que con las nociones de IK y FK adquiridas en el apartado de control cinemático de las articulaciones, permita al usuario controlar la posición de las articulaciones dentro de sus limitaciones, así como que el usuario establezca una posición y una orientación en el espacio para un objeto ("Sphere") que el manipulador debe ser capaz de alcanzar. Para ello será necesario, en primer lugar la creación de dicha interfaz y posteriormente la programación adecuada para que un evento en dicha interfaz se traduzca en movimiento en el robot.

Se comienza por la creación de la interfaz personalizada. Para crear el control de usuario, se pincha sobre el icono situado en el panel de la izquierda que pone *Toogle custom user interface edit mode.* Pulsando el botón Add new interface se crea una nueva interfaz.

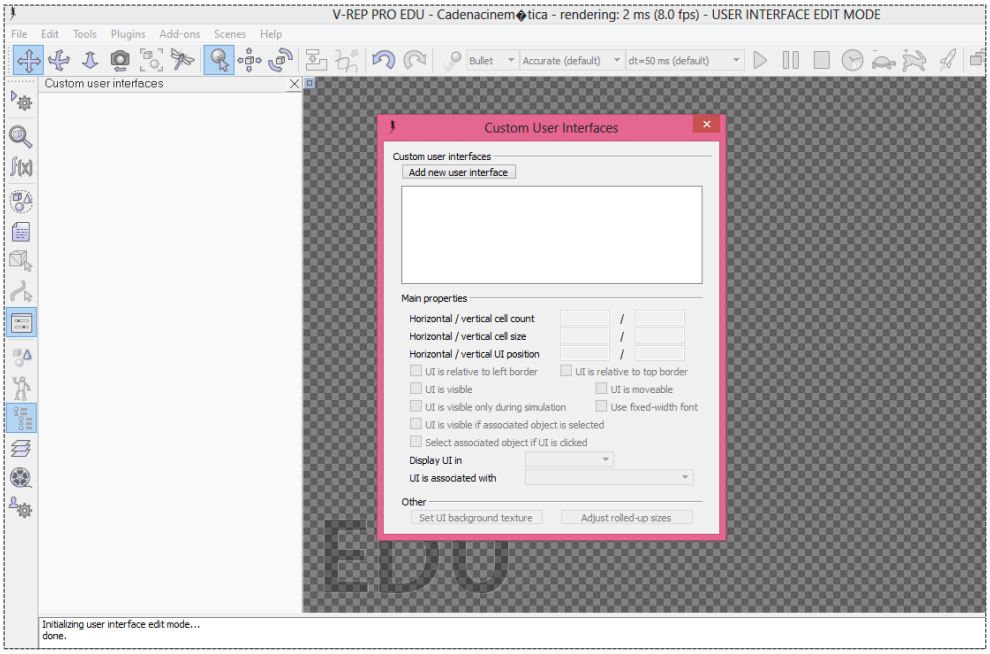


Fig. 5.27 Diálogo de creación de interfaz de usuario

Se establece el tamaño del área de trabajo *client x-size* y *client y-size* eligiendo un valor de 12 para ambas. En las propiedades principales de la interfaz se introduce en *Horizontal/vertical cell count* 28/25 respectivamente. Además se deben activar las casillas de *UI is visible, UI is visible only during simulation* y *UI is moveable.*

Una vez establecidos dichos parámetros procedemos a la creación de nuestras etiquetas y barras de deslizamiento para la resolución de la cinemática directa. Para ello seleccionamos las casillas que queremos que comprenda nuestra barra o nuestra etiqueta y posteriormente seleccionamos *Insert merged button*. En *button type* se establece qué tipo de botón se desea introducir. Se selecciona *Slider* y *Label* en cada uno de los casos respectivamente.

Se ha de tener en cuenta que posteriormente se aplicará una textura como fondo de la interfaz con una imagen del manipulador de perfil, luego es recomendable que cada barra de deslizamiento se posicione en la posición aproximada donde se situará la articulación que controlará. Además para facilitar la programación se debe introducir todas las barras de deslizamiento para el control cinemático directo en orden y seguidas (Slider correspondiente a la articulación1, Slider correspondiente a la articulación 2,…). Se procede de manera similar con las etiquetas y barras de deslizamiento correspondientes a la resolución cinemática inversa. El resultado que deberíamos obtener es el mostrado en la siguiente figura (Fig. 5.28):

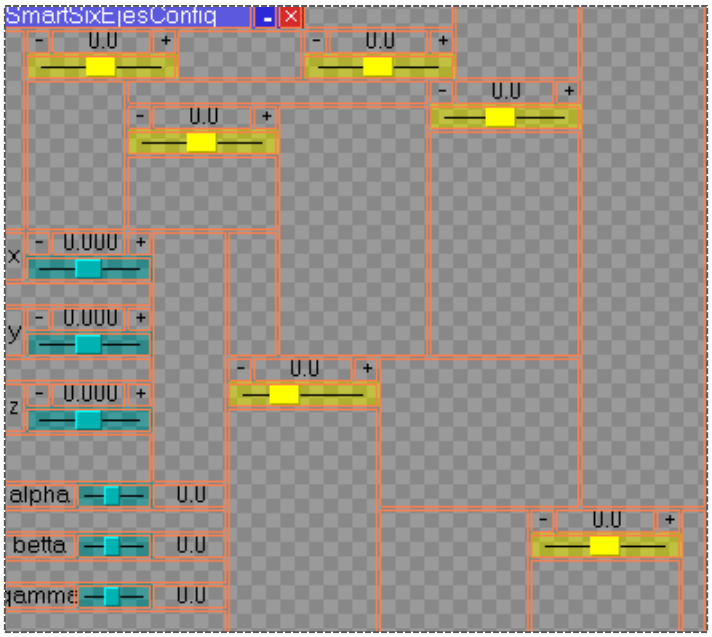


Fig. 5.28 Interfaz de usuario con etiquetas y barras deslizantes

Observando la figura anterior se advierte la presencia de texto y número en ciertas etiquetas o labels. Para agregar el texto deseado solo tenemos que introducirlo en el *apartado button label (up state)* en el dialogo de edición del botón correspondiente. De igual manera podemos asignar colores en el apartado de *button colors*. Estas diferentes opciones de edición permiten crear una interfaz más personalizada. Por último se añade la textura anteriormente mencionada (*Set UI background texture*→*Load texture*→Selección del archivo .jpge) y se obtiene la interfaz (Fig. 5.29). Para que la textura sea visible se seleccionará la opción *Transparent/show background texture* en cada botón de la interfaz.

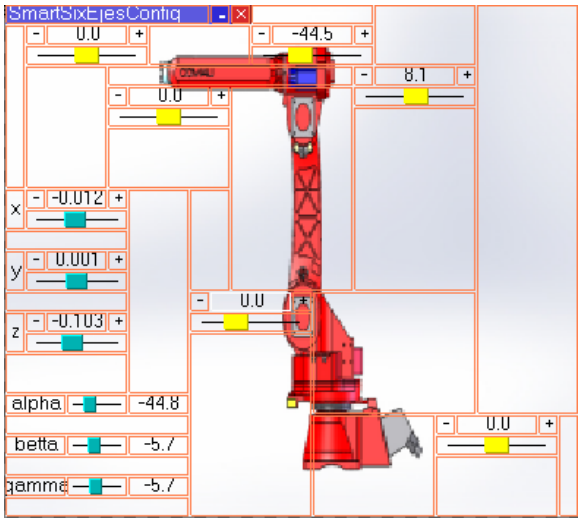


Fig. 5.29 Interfaz de usuario con textura

Llegados a este punto la tarea más ardua resulta la de la programación. Para ello se ha de tener en cuenta el manejador o *handle* que tiene cada uno de los controles de usuario introducidos y que viene dado en el dialogo de propiedades de cada botón en la sección *button handle.* De este modo, como en apartados anteriores, volvemos a asociar un child script al robot con el código correspondiente que se haya en el "Anexo Control de usuario" (apartado 7.3 de esta memoria).

Una vez terminado el código de programación se prueba su funcionamiento. En caso de haber un error en el código, al realizar la simulación aparecerá un mensaje en la escena de "script error". Podemos detectar en que línea de código detecta el error y de qué tipo de error se trata en el diálogo inferior situado debajo de la escena. Algunas de las imágenes del resultado final se adjuntan a continuación:

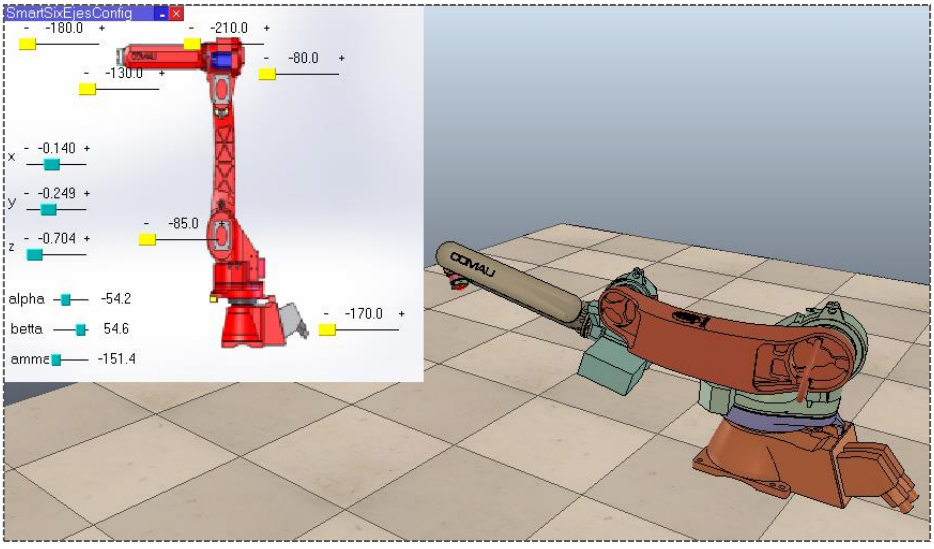


Fig. 5.30 ***Control usuario de las articulaciones para posición mínima de todas ellas mediante FK***

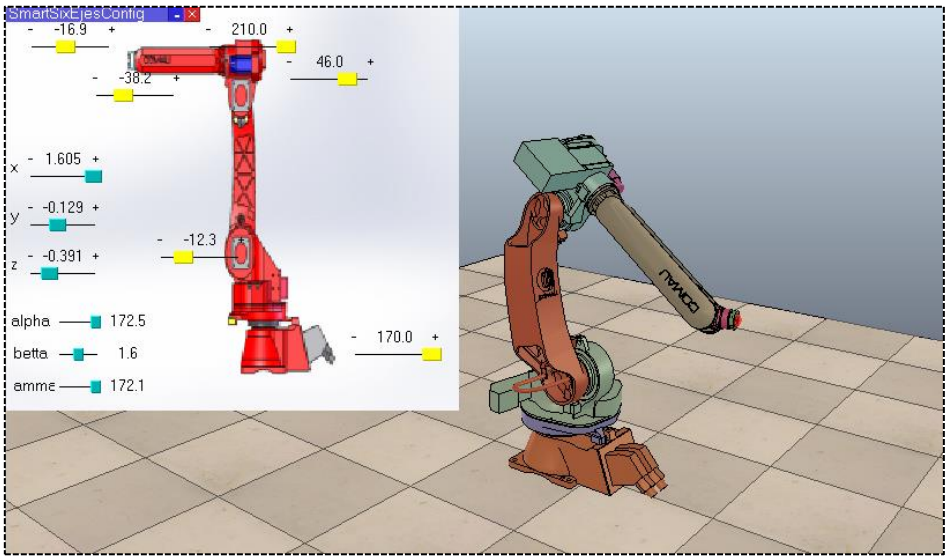


Fig. 5.31 ***Control de usuario para posición arbitraria de las articulaciones mediante FK***

Se observa que siendo el control resuelto mediante FK o IK, el valor de la posición de las articulaciones que se obtiene para alcanzar una cierta posición en el espacio de la esfera (representa la pieza que movería el robot) introducida mediante los sliders azules se muestra en las etiquetas correspondientes a cada articulación y viceversa, siendo el valor introducido por el usuario el de las articulaciones, obtendríamos la posición que alcanzaría la esfera. Todo ello es posible gracias al código de programación asociado al robot.

PRESUPUESTO

6 PRESUPUESTO

Dado que el coste de la herramienta de simulación es nulo, el presupuesto para la realización de este proyecto constará de los medios humanos que han sido requeridos para su consecución. Asignamos un coste de 20€/hora correspondiente al coste medio de un ingeniero recién titulado. Se tendrán en cuenta las diferentes actividades que se han realizado para la asignación de su correspondiente peso en horas diferenciando entre:

- Introducción en la empresa: se refiere al tiempo invertido en el primer contacto con el espacio de trabajo, asentamiento y primera presentación de los medios disponibles.
- Definición del problema: tiempo invertido en establecer objetivos tras cada tarea superada.
- Contacto con el cliente: dado que es el cliente el que establece los objetivos que se han desarrollado en este proyecto, tiempo invertido en el contacto con el mismo para su establecimiento y discusión acerca de su viabilidad.
- Aprendizaje: tiempo destinado a la formación para el manejo de las diferentes herramientas utilizadas en el proyecto.
- Desarrollo del trabajo: tiempo invertido en la realización y consecución de los resultados propuestos.
- Desplazamientos: asumiendo que este proyecto ha sido realizado en una empresa, no habiendo asistido durante todo el periodo de desarrollo del proyecto a la misma.

Ámbito	Horas	Coste/hora	Coste total
Introducción en la empresa	2	20	*40 €*
Definición del problema	8	20	*160 €*
Contacto con el cliente	2	20	*40 €*
Aprendizaje	90	20	*1.800 €*
Desarrollo del trabajo	200	20	*4.000 €*
Desplazamientos	7,5	10	*75 €*
Total	**309,5**		**6.115 €**

7 RESULTADO Y CONCLUSIONES

Afrontar este proyecto ha supuesto un gran reto para mí, dado que, pese a no haberme especializado todavía, la automática y robótica siempre me han impuesto un gran respeto. Simplemente con las asignaturas cursadas durante el grado relacionadas con este campo ya intuía que no me resultaría fácil de resolver la problemática que se plantea en este tipo de proyectos.

Aun así decidí adentrarme en el mundo de la robótica industrial y la verdad es que he sido consciente a lo largo del desarrollo de este proyecto de la gran cantidad de posibilidades que ofrece y del gran campo de investigación que supone y que sigue en continuo desarrollo.

En lo que respecta al trabajo desarrollado, gracias a la utilización de la herramienta de simulación se ha conseguido analizar un proceso robótico concreto, de forma virtual. Han sido simuladas diferentes situaciones características del sector industrial en el que se desarrolla este proyecto, comprobando su funcionamiento, definiendo las trayectorias de movimiento y evaluando de forma global el sistema robótico.

Por tanto, se puede afirmar que hemos cumplido los objetivos propuestos al inicio, ya que se ha conseguido determinar y corregir de forma eficiente las incidencias que podrían haber surgido en una implementación realizada directamente sobre un entorno real.

Se han introducido los aspectos matemáticos de representación de la posición y orientación y movimientos de traslación y rotación de un sistema de referencia, para poder comprender los fundamentos de los movimientos del manipulador en la célula de trabajo y en general los fundamentos matemáticos en los que se apoya la robótica.

Para el desarrollo de este trabajo final de carrera, se ha aprendido y analizado el funcionamiento de la aplicación de simulación V-REP, así como ciertas nociones del programa de diseño SolidWords, ambas herramientas de uso común en entornos relacionados con células robóticas industriales.

ANEXOS

8 ANEXOS

8.1 Anexo Trayectoria

8.1.1 Coordenadas de los puntos de definición de la trayectoria:

(-71.547, 9.550, 84.484, 126.135, 93.222, 60.000)
(-66.837, 3.099, 84.934, 126.135, 93.222, 60.000)
(-65.659, 1.486, 85.046, 126.135, 93.222, 60.000)
(-63.509, 11.199, 84.659, 126.135, 93.222, 60.000)
(-58.632, 19.856, 83.980, 126.135, 93.222, 60.000)
(-51.463, 26.740, 83.192, 120.963, 92.774, 60.000)
(-42.030, 30.024, 82.720, 116.867, 92.007, 60.000)
(-32.150, 31.464, 82.535, 107.277, 92.059, 60.000)
(-22.172, 31.923, 82.582, 99.788, 91.632, 60.000)
(-12.181, 31.864, 82.747, 94.184, 90.631, 60.000)
(-2.199, 31.487, 82.937, 91.348, 88.724, 60.000)
(7.785, 31.132, 83.087, 90.513, 86.583, 60.000)
(17.769, 30.754, 83.119, 91.066, 84.875, 60.000)
(27.752, 30.484, 82.941, 92.215, 83.927, 60.000)
(37.738, 30.802, 82.641, 93.635, 83.591, 60.000)
(47.701, 31.513, 82.348, 95.194, 83.728, 60.000)
(57.628, 32.751, 82.338, 96.596, 84.594, 60.000)
(67.436, 34.580, 82.847, 97.652, 85.827, 60.000)
(77.175, 36.463, 84.026, 97.997, 87.771, 60.000)
(86.863, 38.355, 85.569, 97.519, 90.559, 60.000)
(96.553, 40.406, 86.872, 96.407, 94.664, 60.000)
(106.423, 41.852, 87.275, 94.572, 100.664, 60.000)
(116.334, 42.656, 86.511, 92.543, 108.568, 60.000)
(126.166, 42.354, 84.953, 90.125, 118.074, 60.000)
(135.764, 40.708, 82.869, 86.983, 128.650, 60.000)
(144.805, 37.362, 80.236, 82.929, 139.278, 60.000)
(152.743, 31.475, 78.170, 77.151, 149.687, 60.000)
(159.222, 23.825, 77.653, 67.898, 159.035, 60.000)
(163.120, 14.713, 77.938, 49.861, 166.754, 60.000)
(162.928, 4.818, 78.348, 11.173, 171.094, 60.000)
(159.111, -4.364, 78.548, -29.894, 169.640, 60.000)
(153.074, -12.302, 78.470, -48.774, 165.961, 60.000)
(145.914, -19.268, 78.354, -59.089, 162.180, 60.000)
(138.154, -25.573, 78.487, -67.183, 157.273, 60.000)
(130.080, -31.441, 78.942, -73.921, 151.313, 60.000)
(121.956, -37.137, 80.040, -79.811, 144.444, 60.000)
(113.852, -42.478, 82.313, -84.013, 137.376, 60.000)
(105.450, -46.721, 85.634, -87.684, 130.525, 60.000)

(96.137, -48.503, 88.734, -90.513, 124.237, 60.000)
(86.272, -47.947, 89.947, -92.948, 118.421, 60.000)
(76.378, -46.616, 89.657, -94.912, 113.267, 60.000)
(66.563, -45.014, 88.760, -96.240, 108.625, 60.000)
(56.779, -43.314, 87.646, -97.043, 105.564, 60.000)
(46.965, -41.769, 86.598, -97.272, 103.456, 60.000)
(37.106, -40.314, 85.915, -97.097, 102.162, 60.000)
(27.213, -38.959, 85.601, -97.016, 100.951, 60.000)
(17.288, -37.828, 85.500, -96.940, 100.041, 60.000)
(7.362, -36.692, 85.511, -97.217, 99.601, 60.000)
(-2.562, -35.536, 85.490, -98.113, 99.431, 60.000)
(-12.498, -34.503, 85.327, -100.569, 99.632, 60.000)
(-22.415, -33.296, 85.056, -105.444, 99.596, 60.000)
(-32.281, -31.717, 84.704, -112.064, 98.608, 60.000)
(-42.039, -29.485, 84.346, -120.685, 97.395, 60.000)
(-51.265, -25.530, 84.156, -130.321, 95.977, 60.000)
(-58.648, -18.857, 84.273, -135.087, 95.696, 60.000)
(-63.480, -10.158, 84.685, -140.866, 95.097, 60.000)
(-65.648, -0.467, 85.049, -147.840, 93.610, 60.000)
(-65.673, 0.488, 85.051, -156.059, 91.407, 60.000)
(-67.500, -0.323, 85.002, -156.059, 91.407, 60.000)
(-74.810, -3.569, 84.806, -156.059, 91.407, 60.000)

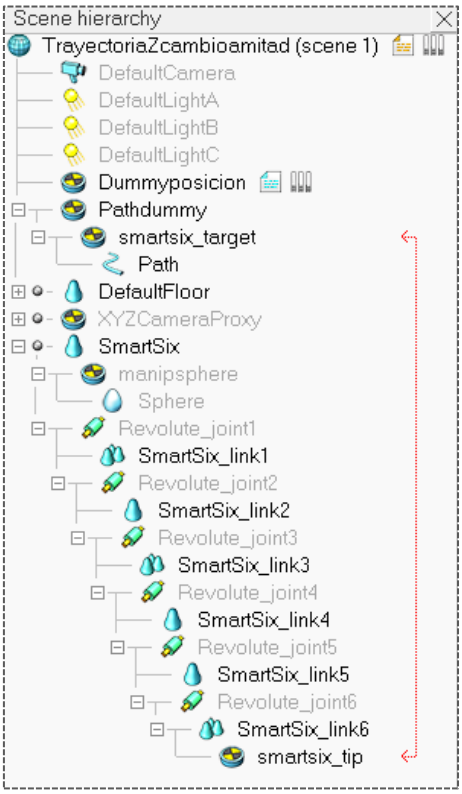
8.1.2 Código de programación del child script para movimiento del robot sobre la trayectoria.

El texto escrito en color verde corresponde a comentarios en el código para la mejor comprensión de los pasos seguidos. Podemos observar la influencia de las matrices homogéneas a la hora de realizar cualquier tipo de movimiento de traslación o rotación en el espacio.

```
simSetThreadSwitchTiming(2) -- Default timing for automatic thread switching
simDelegateChildScriptExecution()

dummy=simGetObjectHandle('Dummyposicion')
pathdummy=simGetObjectHandle('Pathdummy')
path=simGetObjectHandle('Path')

--while (simGetSimulationState()~=sim_simulation_advancing_abouttostop) do


for i=0,1,0.001 do
pos=simGetPositionOnPath(path,i) -- pos is the absolute position at distance i on the path
o=simGetOrientationOnPath(path,i) -- o is the absolute orientation at distance i on the path
matr=simBuildMatrix(pos,o) -- matr is the absolute config at distance i on the path
```

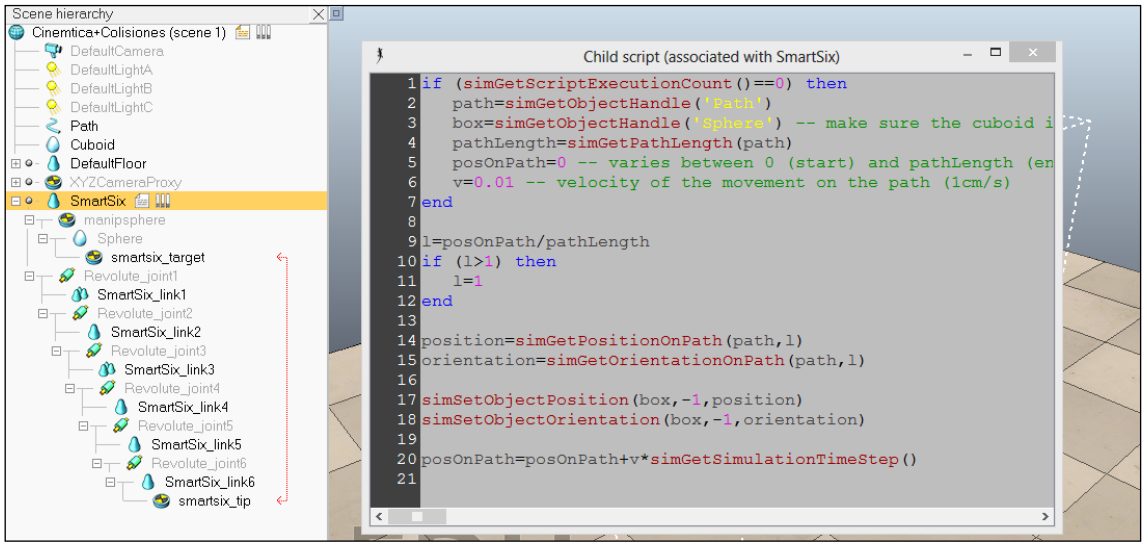
```
-- We actually desire a relative config on the path (since the path is constantly moving)
pathDummyMatr=simGetObjectMatrix(pathdummy,-1) -- pathDummyMatr is the absolute config of the path
pathDummyMatrInv=simGetInvertedMatrix(pathDummyMatr) -- pathDummyMatrInv is the absolute config of the path, inverted
matrRel=simMultiplyMatrices(pathDummyMatrInv,matr) -- matrRel is the RELATIVE config at distance i on the path

-- Finally, we need to apply the inverse of matrRel:
matrRelInv=simGetInvertedMatrix(matrRel) -- matrRelInv is the RELATIVE config, inverted, at distance i on the path
simSetObjectMatrix(pathdummy,dummy,matrRelInv)

simWait(0.5,true)

end
```

8.1.3 Imagen de la jerarquía de la escena, para nuevas relaciones de parentesco establecidas:



8.2 Anexo Colisiones

Se describe a continuación el procedimiento para conseguir que el robot siga una trayectoria simple. Existen dos métodos para conseguir este fin como se ha mencionado en el apartado de planificación de la trayectoria. De nuevo se opta por la utilización de un child script, de modo que solo se debe crear la trayectoria y programar el código asociado al robot que se muestra en la siguiente imagen:



Ejemplo de child script non-threaded para seguimiento de trayectoria "Path"

8.3 Anexo Control usuario

Se adjunta el código de programación necesario para la resolución del control de la interfaz de usuario. Se ha implementado en un script non-threaded. El texto escrito en verde corresponde a comentarios para la mejor interpretación del código.

```
if (simGetScriptExecutionCount()==0) then

smartsix=simGetObjectHandle('SmartSix')
tip=simGetObjectHandle('smartsix_tip')
sphere=simGetObjectHandle('Sphere')
manipsphere=simGetObjectHandle('manipsphere')


armJoints={-1,-1,-1,-1,-1,-1}

for i=1,6,1 do
        armJoints[i]=simGetObjectHandle('Revolute_joint'..i)
end
```

```
ui=simGetUIHandle('UI')

simSetUIButtonLabel(ui,0,simGetObjectName(smartsix)..'EjesConfig') -- Set the UI title (with the name of the current robot)



ik1=simGetIkGroupHandle('ikUndamped')
ik2=simGetIkGroupHandle('ikDamped')
ikFailedReportHandle=-1


initSizeFactor=simGetObjectSizeFactor(smartsix) -- only needed if we scale the robot up/down

-- desired joint positions, and desired cartesian positions:

desiredJ={0,0,0,0,0,0} -- when in FK mode

for i=1,6,1 do
simSetJointPosition(armJoints[i],desiredJ[i])
end

desiredConf={0,0,0,0,0,0} -- when in IK mode
currentConf={0,0,0,0,0,0} -- when in IK mode

ikMinPos={-1*initSizeFactor,-1*initSizeFactor,-1*initSizeFactor}
ikRange={2*initSizeFactor,2*initSizeFactor,2*initSizeFactor}


-- We compute the initial position and orientation of the tip RELATIVE to the robot base (because the base is moving)

initialTipPosRelative=simGetObjectPosition(tip,smartsix)

-- Before V-REP V2.5.1, "simGetObjectOrientation" contained a bug when Euler angles were queried not absolutely

-- So instead of using "initialTipOrientRelative=simGetObjectOrientation(tip,smartsix)", we make it a little bit more complicated:

m=simGetObjectMatrix(tip,smartsix)
initialTipOrientRelative=simGetEulerAnglesFromMatrix(m)


movementMode=0 -- 0=FK, 1=IK through dialog, 2=IK through manipulation sphere

maxJointVelocity=140*math.pi/180
maxPosVelocity=1.0*initSizeFactor
maxOrientVelocity=45*math.pi/180
previousS=initSizeFactor
```

```
end



simHandleChildScript(sim_handle_all_except_explicit)

s=simGetObjectSizeFactor(smartsix)
if (s~=previousS) then
        f=s/previousS

            for i=1,3,1 do

                desiredConf[i]=desiredConf[i]*f
                currentConf[i]=currentConf[i]*f
                ikMinPos[i]=ikMinPos[i]*f
                ikRange[i]=ikRange[i]*f
                initialTipPosRelative[i]=initialTipPosRelative[i]*f
             end

        maxPosVelocity=maxPosVelocity*f
        previousS=s
end

pos=simGetObjectPosition(sphere,sim_handle_parent)

m=simGetObjectMatrix(sphere,sim_handle_parent)
euler=simGetEulerAnglesFromMatrix(m)

if (math.abs(pos[1])>0.0001)or(math.abs(pos[2])>0.0001)or(math.abs(pos[3])>0.0001)or

(math.abs(euler[1])>0.0005)or(math.abs(euler[2])>0.0005)or(math.abs(euler[3])>0.0005)
then

        movementMode=2 -- IK by sphere manipulation
        m=simGetObjectMatrix(sphere,-1)
        simSetObjectMatrix(manipsphere,-1,m)
        simSetObjectPosition(sphere,sim_handle_parent,{0,0,0})
        simSetObjectOrientation(sphere,sim_handle_parent,{0,0,0})


        m=simGetObjectMatrix(sphere,smartsix)
        euler=simGetEulerAnglesFromMatrix(m)


desiredConf={m[4]-initialTipPosRelative[1],m[8]-initialTipPosRelative[2],m[12]-
initialTipPosRelative[3],euler[1]-initialTipOrientRelative[1],euler[2]-
initialTipOrientRelative[2],euler[3]-initialTipOrientRelative[3]}

            for i=1,6,1 do
                currentConf[i]=desiredConf[i]
            end
```

```
end



buttonID=simGetUIEventButton(ui)

if (buttonID>=3)and(buttonID<=8) then -- we want to control the arm in FK mode!
        cyclic,interval=simGetJointInterval(armJoints[buttonID-2])
        desiredJ[buttonID-2]=interval[1]+simGetUISlider(ui,buttonID)*0.001*interval[2]
        movementMode=0
end


if ((buttonID>=15)and(buttonID<=17)) then -- we want to control the arm in IK mode! (position only is handled here)
        desiredConf[buttonID-14]=ikMinPos[buttonID-14]+ikRange[buttonID-14]*simGetUISlider(ui,buttonID)/1000
        movementMode=1
end

if ((buttonID==18)or(buttonID==20)) then -- we want to control the arm in IK mode! (orientation 1&3 only is handled here)
        desiredConf[buttonID-14]=math.pi*(-1+2*simGetUISlider(ui,buttonID)/1000)
        movementMode=1
end

if (buttonID==19) then -- we want to control the arm in IK mode! (orientation 2 only is handled here)
        desiredConf[buttonID-14]=0.5*math.pi*(-1+2*simGetUISlider(ui,buttonID)/1000)
        movementMode=1
end


if movementMode==1 then -- We are in IK mode

        maxLinVariationAllowed=maxPosVelocity*simGetSimulationTimeStep()
        maxAngVariationAllowed=maxOrientVelocity*simGetSimulationTimeStep()
        deltaX={0,0,0,0,0,0}
        -- position:

        for i=1,3,1 do
                deltaX[i]=desiredConf[i]-currentConf[i]
                if (math.abs(deltaX[i])>maxLinVariationAllowed) then
                        deltaX[i]=maxLinVariationAllowed*deltaX[i]/math.abs(deltaX[i]) -- we limit the variation to the maximum allowed
                end
        end
        -- orientation:

        for i=1,3,1 do
                deltaX[3+i]=desiredConf[3+i]-currentConf[3+i]
```

```
-- Normalize delta to be between -pi and +pi (or -pi/2 and +pi/2 for beta):

        if (i==2) then
                rnge=math.pi
        else
                rnge=math.pi*2
        end

        deltaX[3+i]=math.fmod(deltaX[3+i],rnge)

                if (deltaX[3+i]<-rnge*0.5) then
                        deltaX[3+i]=deltaX[3+i]+rnge
                else

                        if (deltaX[3+i]>rnge*0.5) then
                                deltaX[3+i]=deltaX[3+i]-rnge
                        end
        end

if (math.abs(deltaX[3+i])>maxAngVariationAllowed) then

        deltaX[3+i]=maxAngVariationAllowed*deltaX[3+i]/math.abs(deltaX[3+i])  -- we limit the variation to the maximum allowed

                end
end


for i=1,6,1 do
        currentConf[i]=currentConf[i]+deltaX[i]
end


-- Normalize the orientation part to display normalized values:

for i=1,3,1 do
        f=1
        if i==2 then f=0.5 end
        currentConf[3+i]=math.fmod(currentConf[3+i],math.pi*2*f)
                if (currentConf[3+i]<-math.pi*f) then
                        currentConf[3+i]=currentConf[3+i]+math.pi*2*f
                else
                if (currentConf[3+i]>math.pi*f) then
                        currentConf[3+i]=currentConf[3+i]-math.pi*2*f
                end
        end
end
```

```
pos={0,0,0}
orient={0,0,0}

for i=1,3,1 do

        pos[i]=initialTipPosRelative[i]+currentConf[i]
        orient[i]=initialTipOrientRelative[i]+currentConf[3+i]
end

-- We set the desired position and orientation

simSetObjectPosition(manipsphere,smartsix,pos)
simSetObjectOrientation(manipsphere,smartsix,orient)
end

if (movementMode==1) or (movementMode==2) then

        if (simHandleIkGroup(ik1)==sim_ikresult_fail) then

        -- the position/orientation could not be reached.

        simHandleIkGroup(ik2) -- Apply a damped resolution method

        if (ikFailedReportHandle==-1) then -- We display a IK failure report message

        ikFailedReportHandle=simDisplayDialog("IK failure report","IK solver
failed.",sim_dlgstyle_message,false,"",nil,{1,0.7,0,0,0,0})

        end

        else

        if (ikFailedReportHandle>=0) then

        simEndDialog(ikFailedReportHandle) -- We close any report message about IK failure

        ikFailedReportHandle=-1

        end

        end

        -- Now update the desiredJ in case we switch back to FK mode:

        for i=1,6,1 do

                desiredJ[i]=simGetJointPosition(armJoints[i])
        end
        end



if (movementMode==0) then    -- We are in FK mode
```

```
currentJ={0,0,0,0,0,0}

for i=1,6,1 do
        currentJ[i]=simGetJointPosition(armJoints[i])
end


maxVariationAllowed=maxJointVelocity*simGetSimulationTimeStep()

for i=1,6,1 do
        delta=desiredJ[i]-currentJ[i]

        if (simGetJointInterval(armJoints[i])) then
                if (delta>math.pi) then
                        delta=delta-math.pi*2
                end

                if (delta<-math.pi) then
                        delta=delta+math.pi*2
                end
        end
-- Joint is NOT cyclic, we go the fastest direction:

        if (math.abs(delta)>maxVariationAllowed)then
        delta=maxVariationAllowed*delta/math.abs(delta)
        end


        simSetJointPosition(armJoints[i],currentJ[i]+delta)
        end

        -- Now make sure that everything is ok if we switch to IK mode:

        simSetObjectPosition(manipsphere,-1,simGetObjectPosition(tip,-1))
        simSetObjectOrientation(manipsphere,-1,simGetObjectOrientation(tip,-1))
        tipPosRel=simGetObjectPosition(tip,smartsix)


        m=simGetObjectMatrix(tip,smartsix)
        tipOrientRel=simGetEulerAnglesFromMatrix(m)


        desiredConf={tipPosRel[1]-initialTipPosRelative[1],tipPosRel[2]-
initialTipPosRelative[2],tipPosRel[3]-initialTipPosRelative[3],tipOrientRel[1]-
initialTipOrientRelative[1],tipOrientRel[2]-initialTipOrientRelative[2],tipOrientRel[3]-
initialTipOrientRelative[3]}



for i=1,6,1 do
        currentConf[i]=desiredConf[i]
```

```
end

-- Close any IK warning dialogs:

if (ikFailedReportHandle>=0) then

        simEndDialog(ikFailedReportHandle) -- We close any report message about IK failure
        ikFailedReportHandle=-1
        end

end

-- Now update the user interface:

-- First the FK part, text boxes:

for i=1,6,1 do
    simSetUIButtonLabel(ui,8+i,string.format("%.1f",simGetJointPosition(armJoints[i])*180/
math.pi))
end

-- Then the FK part, sliders, based on the target joint position if in FK mode, or based on the
current joint position if in IK mode:

for i=1,6,1 do
        cyclic,interval=simGetJointInterval(armJoints[i])

         if (movementMode~=0) then
        simSetUISlider(ui,2+i,1000*(simGetJointPosition(armJoints[i])-interval[1])/interval[2])
        else
                simSetUISlider(ui,2+i,1000*(desiredJ[i]-interval[1])/interval[2])
        end
end

-- Now the IK part:

-- First the text boxes:

-- Linear:

for i=1,3,1 do
        str=string.format("%.3f",currentConf[i])
        if (str=='-0.000') then
        str='0.000' -- avoid having the - sign appearing and disappearing when 0

        end



simSetUIButtonLabel(ui,20+i,str)
end
```

```
-- Angular:


for i=1,3,1 do
        str=string.format("%.1f",currentConf[3+i]*180/math.pi)
        if (str=='-0.0') then
                str='0.0' -- avoid having the - sign appearing and disappearing when 0
        end
        simSetUIButtonLabel(ui,23+i,str)
end


-- Now the sliders, based on the desired configuration if in IK mode, or based on the current tip configuration if in FK mode:

-- Linear:

for i=1,3,1 do

        if (movementMode~=0) then
                simSetUISlider(ui,14+i,1000*(desiredConf[i]-ikMinPos[i])/ikRange[i])
        else
                simSetUISlider(ui,14+i,1000*(currentConf[i]-ikMinPos[i])/ikRange[i])
        end
end

-- Angular:

if (movementMode~=0) then

        simSetUISlider(ui,18,1000*(desiredConf[4]+math.pi)/(2*math.pi))
        simSetUISlider(ui,19,1000*(desiredConf[5]+math.pi*0.5)/math.pi)
        simSetUISlider(ui,20,1000*(desiredConf[6]+math.pi)/(2*math.pi))
else

        simSetUISlider(ui,18,1000*(currentConf[4]+math.pi)/(2*math.pi))
        simSetUISlider(ui,19,1000*(currentConf[5]+math.pi*0.5)/math.pi)
        simSetUISlider(ui,20,1000*(currentConf[6]+math.pi)/(2*math.pi))
end

if (simGetSimulationState()==sim_simulation_advancing_lastbeforestop) then



end
```

9 BIBLIOGRAFIA

Página web del soporte software www.coppeliarobotics.com

Antonio Barrientos, Luis Felipe Peñín, Carlos Balaguer y Rafael Aracil: "**Fundamentos de Robótica Industrial**" Ed. McGraw-Hill 1997.

Manual de instrucciones Comau Robotics *"Especificaciones técnicas"* y *"Uso de la unidad de control".*

Tutoriales de introducción a V-REP proporcionados por el tutor.

Página web www.robofoot.eu