3. IMPLEMENTACIÓN SOFTWARE PARA EL PROYECTO DE AUTOMATIZACIÓN DEL CORTE DE PIEZAS.

La simulación virtual del proceso de corte de piezas por parte del robot Rx-90 se basa en todo el trabajo previo que se realiza en [4], por ello se hace completamente necesario explicar los sistemas que en él se utilizaron y comentar aquellas modificaciones implementadas para la incorporación del entorno 3D.

En este capítulo se aporta una descripción funcional y técnica del software desarrollado en el proyecto [4] explicando en los apartados iniciales las actuaciones que se han llevado a cabo para implementar la aplicación original y los cambios que se han incluido para hacer posible la simulación previa.

Se hace necesario en este apartado repetir muchas de las cosas que vienen perfectamente explicadas en [4] con el fin de entender los cambios que se han realizado.

3.1. INTRODUCCIÓN.

La aplicación informática desarrollada tiene como objetivo automatizar el proceso de corte de piezas ligeras de aviación. Para ello, su funcionalidad básica es leer el fichero de salida del software CAD/CAM que contiene la información de la trayectoria de corte a seguir, simularla haciendo un chequeo de alcanzabilidad para todos los puntos que la componen comprobando que las respectivas variables articulares no excedan del rango permitido por cada eje y, si todos ellos resultan alcanzables, se ofrecerá la posibilidad al operario de hacer una comprobación visual de la trayectoria mediante una simulación virtual del manipulador. A continuación si todo es correcto se le transmite la trayectoria al controlador del robot en un formato comprensible por éste para que pueda ejecutarla.

En cierto sentido, podría decirse que la aplicación informática que se ha programado no es más que una especie de postprocesador para el manipulador Stäubli RX-90 en concreto. Este razonamiento se debe a que, partiendo del programa en lenguaje simbólico que contiene la información acerca de la localización y la orientación del extremo de la herramienta para todos los puntos que componen la trayectoria de corte, la aplicación creada debe ejecutar las siguientes operaciones para conseguir la pieza objetivo deseada a partir de la pieza matriz o *stock*:

- Adquirir los puntos iniciales de la pieza matriz en el espacio real para poder calcular la posición y orientación iniciales donde debe situarse el extremo de la herramienta antes de realizar el corte.
- Componer los puntos de la trayectoria contenidos en el programa en lenguaje simbólico respecto a dicha localización en el espacio.
- Obtener los puntos de la trayectoria en coordenadas articulares para determinar si todos los puntos de la misma serán alcanzables por el robot.
- En caso afirmativo, se procede a la simulación virtual para poder comprobar el movimiento del robot antes de poner en funcionamiento el manipulador real.

- Si todo es correcto, se transmite la secuencia de puntos que componen la trayectoria al controlador –en un formato comprensible por éste– y se ejecuta. Si, por el contrario, alguno de los puntos de la trayectoria no fuera alcanzable, se muestra un aviso indicando que debe cambiarse bien la localización inicial del extremo de la herramienta o bien la trayectoria de corte.

Además de esta funcionalidad principal, la aplicación también puede ser utilizada como emulador de terminal. Esta característica permite teleoperar el robot desde un ordenador personal sin tener que emplear ningún software adicional.

3.2. SISTEMAS CAD.

En apartados anteriores hemos comentado que un operario puede iniciar todo el proceso de corte a partir únicamente de un archivo con las posiciones del contorno de la pieza diseñada mediante programas CAD. Así pues, el diseño de la pieza se saldría de lo que abarca el alcance de este proyecto; no obstante, se han utilizado estos programas para diseñar nuevas piezas y comprobar el correcto funcionamiento de todo el sistema.

En [4] se hace una explicación mucho más extensa sobre la programación automática en máquinas de CN, de las herramientas de diseño asistido por ordenador y, por supuesto de CATIA, el programa usado para el diseño y obtención del archivo ".APT" con los puntos del contorno de la pieza.

3.3. INTERFAZ GRÁFICA.

La consola de trabajo que ha de utilizar el operario tenía que ser simple y su vez englobar todas las tareas que permitan utilizar la aplicación de corte automatizado de piezas en comunicación con el controlador.

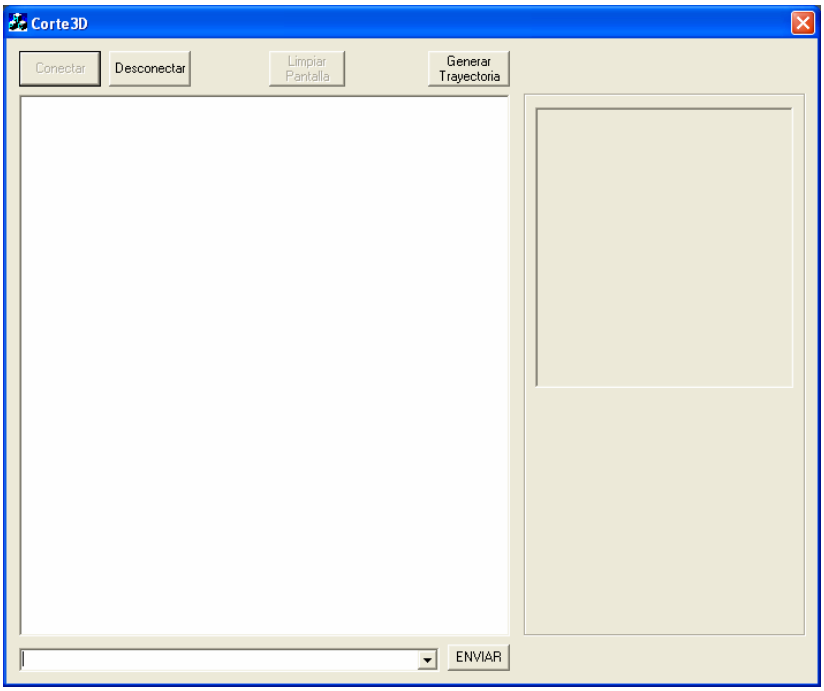

Figura 3. 1. Ventana principal de la aplicación "Corte3D".

Esta pantalla no ha sido modificada y sigue tal y como se implementó en [4]. Utilizándose igualmente como emulador de terminal permitiendo al operario introducir órdenes por línea de comandos al controlador CS7 al pulsar el botón 'Conectar'.

Para pasar al modo de funcionamiento que permite la ejecución de trayectorias basta con pulsar sobre el botón "*Generar Trayectoria*". Tras introducir los datos de entrada necesarios, la aplicación realiza de forma transparente al usuario todos los cálculos necesarios y, si todo es correcto, ordena al robot que ejecute la trayectoria de corte diseñada previamente mediante la herramienta CAD/CAM.

3.4. APLICACIÓN.

Como se ha mencionado anteriormente, al haber integrado un emulador de terminal simple dentro de la aplicación, puede decirse que existen dos modos de funcionamiento:

- Modo 'Emulador de terminal'.
- Modo 'Ejecución de trayectorias'.

En los siguientes subapartados se describirán brevemente las operaciones que se realizan en ambos, haciendo especial hincapié al complejo procesamiento matemático que es necesario llevar a cabo para completar el proceso de ejecución de trayectorias de corte.

3.4.1. Emulador de terminal.

Este modo de funcionamiento, que se encuentra disponible desde el momento en el que se inicializa la conexión entre el ordenador y el controlador, permite que el programa sirva como interfaz entre el manipulador y el usuario. De esta forma es posible tanto gestionar el controlador como teleoperar el brazo robótico (sin tener que usar el mando de control) tecleando los comandos adecuados en el campo correspondiente de la interfaz gráfica.

Para una referencia completa de los comandos del sistema operativo instalado en el armario de control se remite a la lectura de la "Guía de Usuario del sistema operativo VAL+" [11].

Como puede apreciarse en la imagen que se muestra a continuación, en la pantalla principal de la aplicación se han incluido varios controles que permiten la interacción con el controlador de modo gráfico:

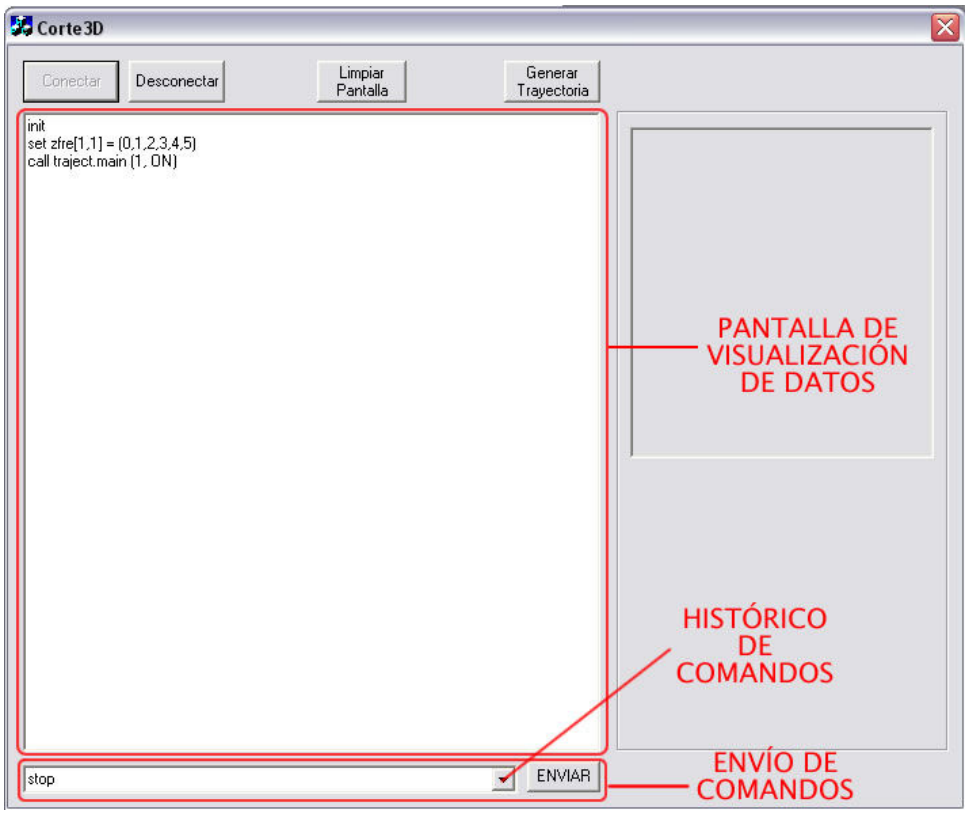

Figura 3. 2.Funcionamiento en modo emulador de Terminal.

- Campo de introducción de texto, donde deben escribirse los comandos que se desea que ejecute el controlador o el robot.
- Campo de visualización, donde puede apreciarse tanto los comandos introducidos como la respuesta enviada por el controlador. Para facilitar el uso del programa se ha diferenciado ambos tipos de contenido mediante el uso de colores distintos:
    - Las órdenes enviadas desde el equipo local hacia el controlador aparecerán en negro.
    - La respuesta del controlador (o cualquier otra información enviada por el mismo de forma asíncrona) aparecerá en color rojo.

Campo de histórico de comandos enviados, donde se puede consultar un historial de las órdenes enviadas en la sesión. Además, es posible su reenvío sin tener que reescribirlos.

3.4.2. Ejecución de trayectorias.

Este modo de funcionamiento avanzado permite, una vez completado el proceso de adquisición de datos, enviar desde la aplicación desarrollada toda la información necesaria para que el robot ejecute la trayectoria de corte programada.

Para ello, hay que realizar una serie de pasos intermedios:

- Cálculo de la posición y orientación iniciales de la herramienta:
    - Enseñanza por demostración ("*teach by showing*"). Se definen 3 puntos (A, B, C) sobre la pieza matriz en el entorno real y el virtual.
    - Se calcula un sistema de referencias {P} conocido respecto al sistema de referencias WORLD {W} del robot en el entorno real $(({}^{W}T_{P})^{-1})$, y respecto al sistema de referencias {D} asociado al entorno virtual o de diseño $(({}^{D}T_{P})^{-1})$.
- Lectura del fichero APT y adquisición de la información geométrica de los puntos que componen la trayectoria.
- Composición de los puntos de la trayectoria respecto al punto inicial.
- Realización de un chequeo de la alcanzabilidad para todos los puntos de la trayectoria calculada.

  Este paso consiste en resolver el problema cinemático inverso en cada punto de la trayectoria empleando el modelo para robot manipulador tipo PUMA simplificado con parámetros D-H del Stäubli RX-90 y comprobar que los valores de las coordenadas articulares en cada punto pertenezcan al rango alcanzable determinado por las características mecánicas del brazo robótico.
- Simulación virtual de los movimientos del robot en la pantalla del ordenador antes del inicio de la trayectoria por parte del manipulador real.
- Ejecución de la trayectoria.
    - Previamente, es necesario adaptar los datos a un formato comprensible por V_TRAJSIG.

- Luego, son transferidos por el puerto serie y almacenados en la memoria del controlador.
- Finalmente, se ejecuta el software V_TRAJSIG (programa creado en lenguaje VAL+ para el seguimiento y el control de trayectorias) en el controlador, indicando los parámetros adecuados.

A modo de resumen, en la *figura 3.3* se presenta un diagrama de flujo de las operaciones realizadas sobre los datos.

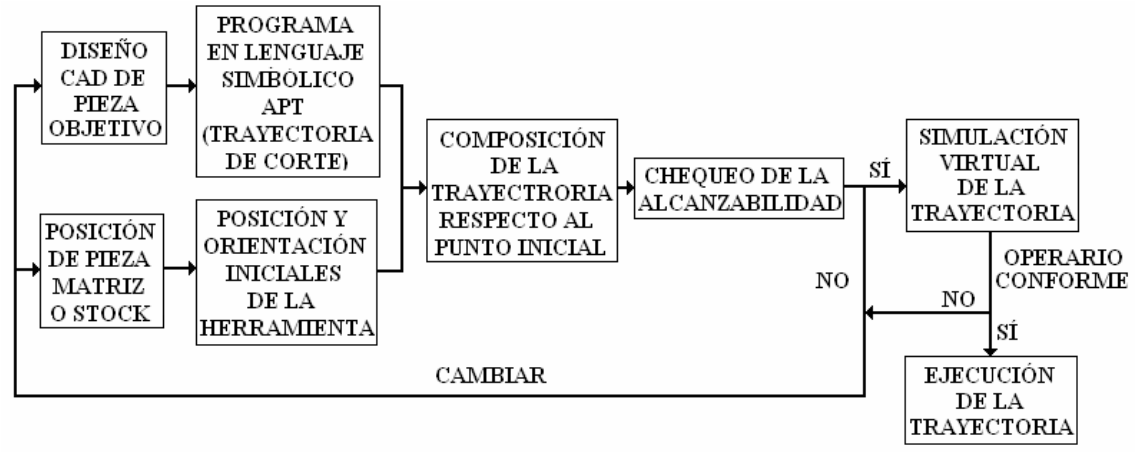

Figura 3. 3. Diagrama de ejecución del proceso de corte.

En los siguientes subapartados se comentarán las distintas operaciones implementadas en el programa: obtención de la posición y orientación iniciales de la herramienta de corte, composición de la trayectoria respecto ese punto, confirmación de que todos los puntos son alcanzables y, si procede, ejecución de la trayectoria. Además, se comenzará por describir los distintos formatos geométricos que ha sido necesario utilizar para poder ejecutar el proceso completo de automatización. Todas estas operaciones están implementadas en la clase CPuntoTrayectoria, cuyas características se describen exhaustivamente en el apartado correspondiente del siguiente capítulo.

3.4.2.1. Formatos geométricos.

Para representar la posición y orientación del extremo de la herramienta de corte en el espacio se han empleado distintas representaciones según fuese conveniente:

- Coordenadas articulares: consiste en un vector de seis componentes que refleja el valor que toman las seis variables articulares.
- Coordenadas cartesianas: consiste en un vector de seis componentes en el que las tres primeras representan la posición en el espacio en coordenadas cartesianas XYZ y las tres últimas informan sobre la orientación. Se han empleado dos tipos de representaciones de la orientación:
    - Mediante los ángulos ZYZ de Euler [1].
    - Mediante un vector unitario tridimensional que define la posición del extremo de la herramienta respecto a un sistema de referencia fijado en la muñeca del manipulador.
- Matriz de transformación homogénea: consiste en una matriz 4x4 compuesta por una matriz de rotación y un vector de translación que representan, respectivamente, la orientación y la posición.

Las coordenadas articulares son empleadas internamente en el programa para chequear la alcanzabilidad. Como ya se ha dicho, consisten en un vector de seis componentes de la forma:

$$\theta = [\theta_1 \quad \theta_2 \quad \theta_3 \quad \theta_4 \quad \theta_5 \quad \theta_6]^T \qquad (3.\ 1)$$

donde θ_i con $i=\{1,2,\ \ldots\ 6\}$ son las variables correspondientes a cada una de las seis articulaciones de las que consta el manipulador empleado.

Los dos tipos de coordenadas cartesianas empleadas se utilizan como formatos de entrada y de salida de información en la aplicación. Por una parte, la representación formada por las coordenadas XYZ y los ángulos ZYZ de Euler, que nosotros llamaremos coordenadas XYZeulerZYZ, son utilizadas por el controlador del robot. Por otra, la representación compuesta por las coordenadas XYZ y un vector, que se ha denominado coordenadas XYZvector para diferenciarlas de las anteriores, es empleada en el programa en lenguaje simbólico APT.

Como ya se ha mencionado, las coordenadas XYZeulerZYZ emplean los ángulos ZYZ de Euler para representar la orientación. Estos tres ángulos, denominados α, β y

γ, definen la orientación mediante tres giros consecutivos en dos de los tres ejes coordenados solidarios al cuerpo: primero respecto al eje Z un ángulo α, luego un ángulo β respecto al nuevo eje Y, y finalmente respecto al nuevo eje Z otra vez (en lugar de X) un ángulo γ.

$$P_{\alpha\beta\gamma} = [X \quad Y \quad Z \quad \alpha \quad \beta \quad \gamma]^T \tag{3. 2}$$

Por su parte, las coordenadas XYZvector representan la orientación mediante un vector unitario de tres componentes (i, j y k) que definen los ángulos de giro "roll" (balanceo), "pitch" (inclinación) y "yaw" (orientación) del extremo de la herramienta de corte respecto a un sistema de referencia fijo situado en la muñeca del manipulador.

$$P_{ijk} = [X \quad Y \quad Z \quad i \quad j \quad k]^T \tag{3. 3}$$

donde X, Y y Z son las coordenadas cartesianas del extremo de la herramienta respecto al sistema de referencia fijo en la muñeca del manipulador, como puede observarse en la figura siguiente:

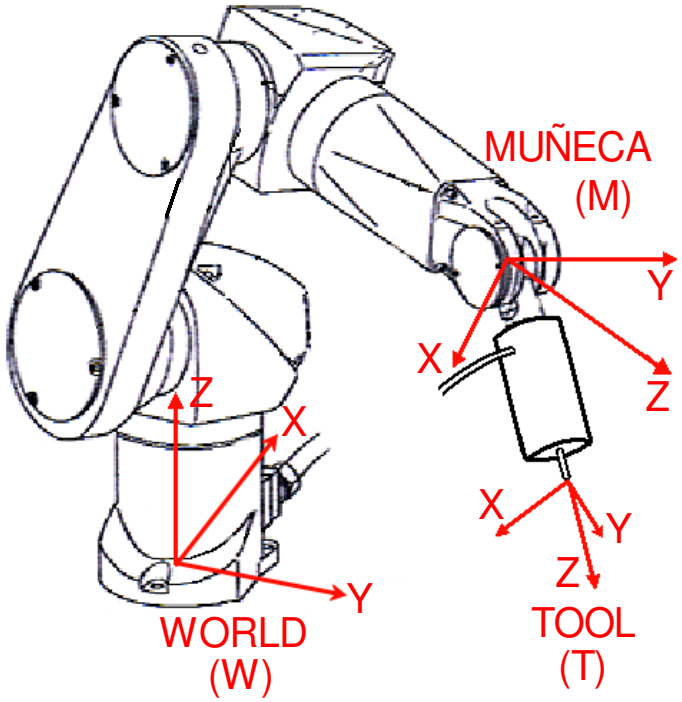

Figura 3.5. Representación de la posición y orientación mediante el formato XYZvector.

Finalmente, el último formato empleado son las matrices de transformación homogénea, que se utilizan internamente en el programa para realizar los cálculos intermedios necesarios. Estas matrices son de la forma:

$$M = \begin{bmatrix} & R & & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (3.4)$$

donde R es una matriz 3x3 que representa la orientación respecto al sistema de referencia fijado en la base del manipulador, y $T=[X\ Y\ Z]^T$ es un vector 3x1 que representa una traslación respecto al origen de coordenadas del mismo.

Obviamente, al emplear diversas representaciones ha sido necesario incorporar también los algoritmos necesarios para convertir de un formato a otro. En los siguientes subapartados se volverá sobre este tema a medida que sea necesario.

3.4.2.2. Cálculo de la posición y la orientación iniciales del extremo de la herramienta.

A partir del programa en lenguaje simbólico que se obtiene como resultado del diseño de la pieza y de la definición de la trayectoria de corte mediante el software CAD/CAM, se dispone de la información necesaria acerca de la representación geométrica de los puntos que componen la trayectoria. Sin embargo, todos estos puntos están referidos a un sistema de referencia propio del diseño, por lo que no son útiles tal cual, sino que es necesario refererirlos al sistema de referencia WORLD asociado al robot. Para ello, estos puntos deben componerse respecto al punto inicial donde estará colocada la herramienta antes de empezar a realizar el corte, que será un punto perteneciente al espacio alcanzable por el manipulador.

El proceso seguido para calcular la posición y la orientación iniciales de la herramienta se basa en la técnica conocida como enseñanza por demostración ("*teach by showing*"). Como puede apreciarse en la siguiente figura, sobre la pieza matriz o stock se definen tres puntos A, B y C fácilmente identificables (como, por ejemplo, tres de los vértices de una de las caras), que son memorizados posicionando el extremo de la herramienta sobre ellos.

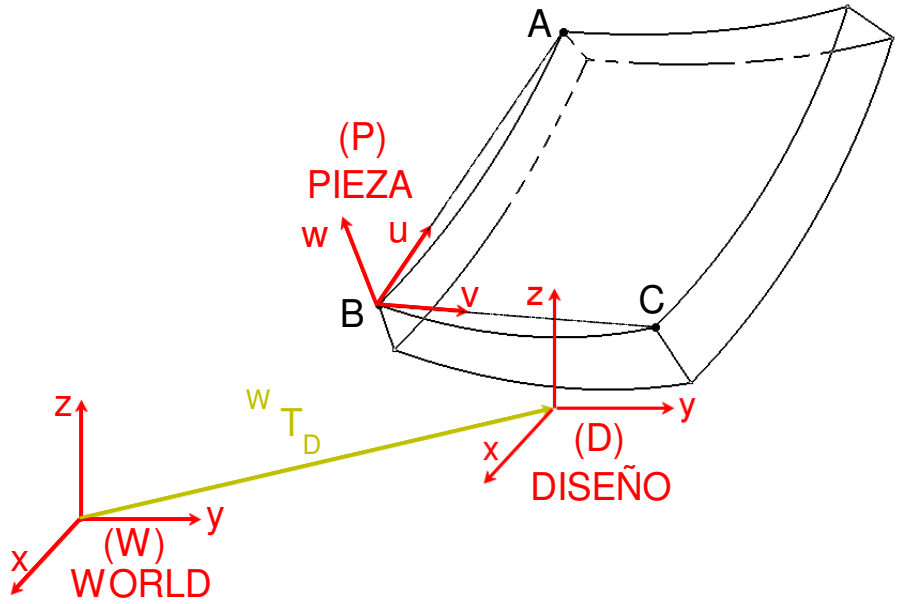

Figura 3. 4. Obtención de la posición y la orientación respecto al sistema World.

Por lo tanto, se dispone de las coordenadas cartesianas de estos tres puntos A, B y C tanto en el entorno real como en el entorno virtual. En el virtual, también son conocidas las coordenadas cartesianas respecto al sistema de referencia del diseño *{D}*, que denotaremos por el subíndice *D*. Por su parte, en el entorno real se conoce la posición de los tres puntos respecto al sistema de coordenadas *{W}* (también conocido como WORLD) asociado al robot, que denotaremos con el subíndice *W*.

A partir de estos puntos es posible definir un sistema de referencia ligado a la pieza y con origen uno de los puntos, tanto en el entorno real como en el virtual. A estos sistemas de referencia los denotaremos como $\{P_W\}$ y $\{P_D\}$ respectivamente. Para calcularlos basta con definir en cada caso tres vectores *u*, *v* y *w* que formen una base ortogonal, lo cual es posible a partir de tres puntos: dos vectores podemos calcularlos como los vectores diferencia entre los puntos, mientras que el tercero se calcula como el producto vectorial de ambos.

Una vez que se han calculado estos sistemas de referencia ligados a la pieza, queda determinada la transformación ${}^{W}T_P$ de *{W}* a $\{P_W\}$ en el entorno real, y la transformación ${}^{D}T_P$ de *{D}* a $\{P_D\}$ en el entorno virtual. Por lo tanto, conocida la posición de un punto p_i cualquiera de la pieza respecto a los sistemas de referencia

WORLD *{W}* o de diseño *{D}*, es posible determinar su posición respecto a los sistemas de referencia ligados a la pieza en sendos entornos. Es decir:

$$ {}^{P_W}p_i = \left({}^{W}T_P\right)^{-1} \cdot {}^{W}p_i \qquad (3.5) $$

$$ {}^{P_D}p_i = \left({}^{D}T_P\right)^{-1} \cdot {}^{D}p_i \qquad (3.6) $$

El último paso consiste en ligar ambas cosas: puntos en la pieza virtual y puntos en la pieza real. Para ello es necesario realizar un modelo virtual en 3D de la pieza matriz de dimensiones y forma idéntica a las de la pieza real. A partir de éste, dado cualquier punto p_i genérico por el cual deba pasar la pieza en el entorno virtual (es decir, dado ${}^{D}p_i$), podemos calcular su correspondiente ${}^{Pw}p_i$. Y como este punto coincide con el de la pieza real, es decir:

$$ {}^{P_D}p_i = {}^{P_W}p_i \qquad (3.7) $$

resulta que, igualando las ecuaciones (3.5) y (3.6) y despejando, obtenemos que:

$$ {}^{W}p_i = {}^{W}T_P \cdot \left({}^{D}T_P\right)^{-1} \cdot {}^{D}p_i \qquad (3.8) $$

En resumen, mediante esta ecuación tan simple podemos convertir los puntos por los que pasa la fresa en el entorno CAD a puntos donde posicionar la herramienta del robot en el entorno real.

Es fundamental realizar estos cálculos con mucha precisión para que exista una equivalencia total entre la posición y la orientación iniciales de la herramienta en la realidad y las definidas en el diseño, de forma que la trayectoria de corte definida en el diseño se ejecute en la realidad sobre la pieza de partida, obteniéndose como resultado final el corte deseado. Esta localización inicial virtual es determinada al trazar la trayectoria de corte en el módulo de CN del sistema CAD/CAM. Aunque es totalmente configurable, generalmente suele ser un punto separado unos 10 cm. del centro de la pieza en dirección normal a la misma.

3.4.2.3. Chequeo de la alcanzabilidad.

Una funcionalidad fundamental de la que debía disponer la aplicación informática desarrollada era la posibilidad de simular la trayectoria de corte en el PC antes de

ejecutarla, de tal forma que si alguno de los puntos de la misma no fuese alcanzable, ésta no se trazara, consiguiéndose así un ahorro de tiempo considerable y evitando posibles complicaciones con el manipulador. En este caso, si la falta de alcanzabilidad no se debiera a que la pieza diseñada es de unas dimensiones excesivas para el manipulador empleado, bastaría con cambiar la posición inicial de la pieza matriz –y por tanto el punto de partida de la herramienta– y recalcular la alcanzabilidad de la nueva trayectoria tantas veces como fuera necesario, pero sin necesidad de ejecutarla cada vez.

El criterio seguido para discriminar si un punto de la trayectoria es alcanzable o no consiste en comprobar si el valor de cada una de las seis variables articulares se encuentra dentro de su rango alcanzable. Estos datos, que se conocen de las especificaciones cinemáticas del robot, se muestran a continuación:

Coordenada articular	**Valor mínimo (º)**	**Valor máximo (º)**
θ_1	-160	160
θ_2	-200	35
θ_3	-52.5	232.5
θ_4	-270	270
θ_5	-105	120
θ_6	-270	270

Tabla 3. 1. Rango de movimiento de cada eje.

Por consiguiente, para poder comprobar si un punto de la trayectoria es alcanzable es necesario obtener el valor de las variables articulares del robot conocidas la posición y la orientación de la herramienta en el espacio cartesiano. Para ello se debe resolver el problema cinemático inverso en ese punto empleando como datos los valores de la representación en forma de matriz de transformación homogénea. En concreto, se resolverá el modelo cinemático inverso para un robot manipulador tipo PUMA, que es un caso singular debido a que posee una muñeca (articulaciones de la cadena cinemática

que unen el brazo y la herramienta) esférica que se caracteriza porque los ejes de las articulaciones de la muñeca se cortan en un punto. Esta configuración permite simplificar el análisis cinemático ya que desacopla el problema del posicionamiento de la garra del problema de la orientación de la misma.

3.4.2.4. Simulación de la trayectoria.

En [4] la "simulación" consistía en comprobar la alcanzabilidad de los puntos de la trayectoria y ahí terminaba; ahora, con la incorporación de la fase de simulación virtual en un entorno DirectX3D la comprobación es, además, visual. Por supuesto es importante comprobar en primer lugar que las coordenadas articulares de cada punto no excedan del rango de su respectivo eje, ya que el Rx-90 virtual no tiene limitaciones mecánicas y podría igualmente seguir una trayectoria errónea.

Dicha simulación la podemos repetir cuantas veces sea necesario antes de poner en funcionamiento el robot real e incluso se puede analizar desde todos los puntos de vista posibles para asegurar la fidelidad de la trayectoria obtenida.

En la imagen virtual no sólo aparece el robot moviéndose, sino que también se carga la pieza diseñada mediante programas CAD/CAM y se ubica en las posiciones donde se encuentra la pieza matriz o *stock*; de esta forma se puede ver en detalle cómo el extremo de la fresa sigue el contorno de la pieza (ya tenga un contorno en dos o tres dimensiones) y ataca perpendicularmente a su superficie.

Como se ha dicho en numerosas ocasiones, el formato de las coordenadas de los puntos que se pasa al controlador es el denominado XYXeulerZYZ, es decir, las coordenadas cartesianas *X*, *Y* y *Z* para la posición, y los ángulos de Euler α, β y γ para la orientación. En cambio, la aplicación "Rx90.exe" encargada de implementar el interfaz gráfico, maneja los puntos de la trayectoria mediante variables articulares correspondientes a los seis ejes que posee el robot.

El programa de simulación permite ser cargado directamente sin estar conectado al manipulador real y cargar trayectorias (en archivos .apt) y realizar todo el proceso de composición de puntos del entorno de diseño al espacio de trabajo del robot, para

convertirlo después mediante el modelo cinemático inverso en coordenadas articulares para realizar la simulación.

Con esta aplicación se pueden visualizar movimientos simples del manipulador virtual para realizar comprobaciones por parte del operario, ya que se pueden mover las articulaciones por separado, obteniendo en todo momento la información necesaria para conocer con exactitud la posición y la orientación del efector final, tanto en variables articulares como en coordenadas cartesianas con los ángulos 'yaw', 'pitch' y 'roll'. Por otro lado, se puede desplazar el extremo del robot siguiendo la dirección de los ejes de referencia correspondientes al sistema ubicado en la base del Rx-90, en este caso, la fresa no cambia su orientación, con lo que resulta más fácil que el movimiento se detenga por razones de inalcanzabilidad del punto destino, ya sea por sobrepasar el rango de alguno de los ejes o porque el punto se salga del espacio de trabajo.

3.4.2.5. Ejecución de la trayectoria.

Antes de transmitir la trayectoria desde el PC hasta el controlador del robot a través del puerto serie es necesario adaptar los puntos desde su representación mediante matrices de transformación homogénea al formato XYZeulerZYZ, que es el único que entiende el controlador. Para ello es necesario resolver las siguientes ecuaciones:

$$Si\ 0 < \beta < 180^{\circ} \Rightarrow \begin{cases} \alpha = A\tan 2(r_{23}, r_{13}) \\ \beta = A\tan 2\left(\sqrt{r_{31}^2 + r_{32}^2}, r_{33}\right) \\ \gamma = A\tan 2(r_{32}, -r_{31}) \end{cases} \quad (3.\ 9)$$

En caso de que $\beta = 0$ ó $\beta = 180^{\circ}$ la solución es degenerada, y sólo puede calcularse la suma o la diferencia de α y γ. Eligiendo $\alpha = 0$ se obtienen las soluciones:

$$\alpha = 0, \quad \beta = 0, \quad \gamma = A\tan 2(-r_{12}, r_{11}) \quad (3.\ 10)$$

$$\alpha = 0, \quad \beta = 180^{\circ}, \quad \gamma = A\tan 2(r_{12}, -r_{11}) \quad (3.\ 11)$$

Finalmente, después de transmitir la trayectoria hacia el armario de control es necesario ejecutarla. De esto se encargará un software propio del controlador llamado V_TRAJSIG. El operador sólo debe encargarse de configurarlo de forma adecuada y de pasarle datos válidos.

V_TRAJSIG es un programa para los robots Stäubli de la serie RX creado en el lenguaje de programación VAL+. La función de este programa consiste básicamente en la generación y el seguimiento de trayectorias a partir de puntos de referencia en coordenadas cartesianas. Además, permite controlar gran cantidad de parámetros relacionados con dichas trayectorias, como por ejemplo parámetros estáticos (tales como definiciones geométricas, posiciones...), parámetros dinámicos (velocidades, aceleraciones, paradas programadas...) e incluso las comunicaciones y periféricos asociados (principalmente constituidos por entradas y salidas tanto digitales como analógicas, control de ejes adicionales...).

3.5. DESCRIPCIÓN TÉCNICA. CLASES.

En este apartado vamos a hablar de los detalles técnicos de la programación de la aplicación desarrollada en [4] para el corte automatizado de piezas; al igual que en apartados anteriores, habrá que repetir muchas de las explicaciones que se dan en [4] para poder contextualizar las modificaciones realizadas sobre el código.

A continuación se expone una tabla con la lista de las distintas clases programadas para implementar una aplicación MFC con Visual C++; y seguidamente se detallarán únicamente aquellas clases en las cuales se han realizado cambios para la incorporación de la fase de la simulación gráfica.

3.5.1. Diagrama de clases.

Clase	Ficheros	Funcionalidades
CCorte3DApp	Corte3D.h Corte3D.cpp	Clase principal de la aplicación.
CCorte3DDlg	Corte3DDlg.h Corte3DDlg.cpp	Clase encargada de mostrar la pantalla principal de la aplicación y gestionar los eventos que ocurren en la misma.
CRichEditEx	RichEditEx.h RichEditEx.cpp	Clase derivada del control CRichEditCtrl que tiene la particularidad de que hace autoscroll vertical incluso después de que el contenido se actualice, bien con UpdateData o bien de otra forma. Además, contiene un método para mostrar objetos CString de un color

		determinado en el control y otro para borrar su contenido.
CSelectConfigCOMDlg	SelectConfigCOMDlg.h SelectConfigCOMDlg.cpp	Clase encargada de mostrar el cuadro de diálogo que permite configurar los parámetros de la conexión a través del puerto serie RS-232 y gestionar los eventos que ocurren en el mismo.
CMundoVirtualDlg	MundoVirtualDlg.h MundoVirtualDlg.cpp	Clase encargada de mostrar el cuadro de diálogo que permite introducir la información geométrica sobre el diseño de la pieza necesaria para calcular la transformación a aplicar entre el entorno virtual y el real. También sirve para gestionar los eventos que ocurren en el mismo.
CAboutDlg	Corte3DDlg.h Corte3DDlg.cpp	Clase encargada de mostrar el cuadro de diálogo “Sobre…” y gestionar los eventos que ocurren en el mismo.
CPuertoSerie	PuertoSerie.h PuertoSerie.cpp	Clase para usar el puerto serie para comunicarse con el controlador del robot Staübli RX-90.
CExcepcionPuertoSerie	PuertoSerie.h PuertoSerie.cpp	Clase para gestionar las excepciones debidas a la

		comunicación por el puerto serie.
CPuntoTrayectoria	PuntoTrayectoria.h PuntoTrayectoria.cpp	Esta clase contiene las estructuras y operaciones necesarias para trabajar con los puntos de las trayectorias: permite almacenar los puntos en diversos formatos (XYZvector, XYZeulerZYZ, theta[i] {i=1..6}, matriz de transformación homogénea), realizar conversiones entre ellos y ejecutar las operaciones necesarias para cada formato. Escribe en ficheros '.txt' los puntos de la trayectoria para su posterior uso en la simulación virtual y en MatLab como método de depuración.
CDlgSimulacion	DlgSimulacion.h DlgSimulación.cpp	Clase nueva implementada con el trabajo actual (no viene de [4]) cuya finalidad es la de crear un diálogo para preguntar al operario si desea iniciar la aplicación de simulación virtual. Si la respuesta es sí, se llama a la aplicación 'Rx90.exe'

Tabla 3. 2. Clases desarrolladas en la aplicación 'Corte3D'.

3.5.2. CCorte3DApp.

Ésta es la clase principal, que se genera automáticamente por el entorno de desarrollo al crear un nuevo proyecto MFC.

Los atributos miembro (variables) de esta clase no han sido modificados, siguen siendo los mismos, pero sí se han producido cambios en algunas de las funciones (métodos miembro) y se han añadido otras nuevas. A continuación sólo comentamos aquellas que han sido modificadas o creadas para este proyecto.

- **BOOL AbrirYLeerFicheroAPT(CString& strContFicheroAPT)**
  Esta función se utiliza para abrir y leer un fichero '.apt'. Internamente, se abre un manejador de ficheros al archivo con formato APT indicado, se lee su contenido y se almacena en un *buffer* temporal. Posteriormente, se cierra el manejador del fichero, se guarda la información en un objeto de tipo CString y se borra el *buffer* temporal de lectura. De esta forma se le pregunta al operario la pieza que desea obtener de la plancha matriz. El nombre de dicha pieza es importante para poder cargarlo *a posteriori* en la aplicación de simulación, por ello lo guardamos en un fichero denominado "nombrepieza.txt", de forma que no haya que volver a preguntar lo mismo dos veces al operario.

- **BOOL ProcesarContenidoFicheroAPT(CString& strContFicheroAPT)**
  Una vez que tenemos el contenido del fichero '.apt' almacenado en una variable tipo CString, esta función procesa dicho contenido y constituye el array de objetos **CPuntoTrayectoria** (según el número de puntos del contorno que se hubiesen calculados en el programa CAD/CAM) que contiene la información geométrica sobre la trayectoria de corte. A partir de éste, también realiza todas las operaciones necesarias para determinar si la trayectoria es alcanzable o no. Si lo es, se pregunta al operario si desea iniciar la aplicación de simulación virtual y envía la información necesaria

de la trayectoria por el puerto serie, ordenando al controlador del robot que la ejecute.

- **void CalculaPuntoCentral(int NPuntos)**

  Función que permite calcular el punto central de la pieza en la que se va a trabajar. Las coordenadas de este punto se refieren al sistema de referencia ubicado en la base del robot. Este dato será utilizado más adelante por la aplicación virtual para localizar la pieza diseñada en el entorno del manipulador en el sitio exacto donde se encuentra la pieza matriz real.

  Como parámetros sólo se le pasa una variable tipo 'int' que contiene el número de puntos de la trayectoria.

  No devuelve nada. Es un método tipo 'void'

- **void PuntosDeAproximacion()**

  Nueva función que crea nuevos puntos en la trayectoria únicamente para el seguimiento virtual del robot. Estos puntos implementan una maniobra de aproximación desde una distancia de 10 cm. Al punto inicial de la trayectoria en la dirección del eje Z del sistema de referencias asociado al efector final. Como hemos dicho, sólo afecta a la trayectoria que sigue el robot virtual en la nueva aplicación gráfica.

  No recibe nada como parámetro ni devuelve nada.

- **void PuntosDeSeparacion(int nNumeroPuntosAPT)**

  Método muy parecido al anterior (PuntosDeAproximación) que crea también nuevos puntos pero en esta ocasión para separarse del último punto de la trayectoria en la dirección del extremo de la fresa. Al igual que antes, estos puntos sólo se utilizan para el seguimiento de la trayectoria por parte del robot virtual en la aplicación de simulación visual.

  En este caso sí recibe como parámetro el número de puntos de contorno de la pieza obtenidos del fichero '.apt'.

  No devuelve nada.

- **void CCorte3DApp::ComprobarPosiblesSaltos(int PuntosAPT)**

  Esta función pretende emular al programa V_TRAJSIG del controlador en el caso de que en el transcurso de la trayectoria se produzca un salto considerable en alguna de las variables articulares del robot. Cuando esto sucede, el robot real no mueve el extremo de la fresa mientras adapta las articulaciones a sus nuevos valores, de forma que el resultado sea perfecto. Sin esta función, el robot virtual movía bruscamente la articulación pertinente sin preocuparse de las demás produciendo un movimiento totalmente indeseado. En este método se introducen nuevos puntos para que a la vez que la articulación correspondiente cambia su valor (recordamos que se trata de un cambio considerable) las demás articulaciones cambien paulatinamente para que la fresa no modifique su posición de corte. Esta modificación sólo afecta a la trayectoria a seguir por el robot virtual, no trastoca la trayectoria calculada y que se le envía al controlador, ya que el programa V_TRAJSIG ya solucionaba estos problemas eficientemente.

  Se le pasa como parámetro el número de puntos de la trayectoria en una variable tipo 'int'.

3.5.3. CCorte3DDlg.

Clase encargada de mostrar la pantalla principal de la aplicación y gestionar los eventos que ocurren en la mismo. Es generada automáticamente por el entorno de desarrollo cuando se crea el cuadro de diálogo.

En esta clase únicamente se ha modificado respecto al proyecto original [4] una de sus funciones y de forma muy leve. Lo comentamos a continuación.

- **void OnBtApt()**

  Método manejador del mensaje lanzado cuando se pulsa el botón "*Generar Trayectoria*". Su objetivo final es realizar todas las operaciones necesarias para que el robot ejecute la trayectoria contenida en el fichero APT que se le indique.

En primer lugar, solicita la información necesaria para calcular los sistemas de referencia asociados a la pieza matriz tanto en el entorno real como en el virtual. Posteriormente, llama a los métodos adecuados de la clase **CCorte3Dapp** (‘**CCorte3DApp::AbrirYLeerFicheroAPT**’ y ‘**CCorte3DApp::ProcesarContenidoFicheroAPT**’). Finalmente, si todo es correcto, se envían al controlador los comandos necesarios para que se ejecute la trayectoria.

Lo único que se ha modificado aquí es la llamada al cuadro de diálogo que pregunta al operario si desea simular previamente la trayectoria de corte mediante la aplicación del Rx-90 virtual y una instrucción que se envía al controlador por línea de comandos para aproximarse al punto inicial del corte.

3.5.4. CPuntoTrayectoria.

Esta clase contiene las estructuras y operaciones necesarias para trabajar con los puntos de las trayectorias. Las características principales de esta clase son las siguientes:

- Permite almacenar la información geométrica de los puntos que componen la trayectoria de corte en diversos formatos, según sea necesario:
  - XYZvector, que es el que contienen los ficheros en formato APT.
  - XYZeulerZYZ, theta[i] {i=1..6}, , que es el que utiliza V_TRAJSIG.
  - Coordenadas articulares, utilizado para el test de alcanzabilidad y utilizado por la aplicación de simulación para el movimiento del robot virtual.
  - Matriz de transformación homogénea, utilizado para cálculos intermedios.
- Permite realizar conversiones entre estos formatos.

Incorpora métodos que implementan algoritmos para realizar las operaciones necesarias para cada formato (composición de matrices, inversión de matrices, resolución del modelo cinemático inverso, etc…).

No se han modificado los atributos de esta clase, así que sólo enumeramos y comentamos aquellos métodos en los que sí se han efectuado cambios o se han creado específicamente para este proyecto.

- **void ResolverModeloCinematicoInverso(int nBrazo, int nCodo, int nMuneca)**

  Método que sirve para realizar la resolución del problema cinemático inverso para el robot Stäubli RX90 en el punto almacenado. Para ello utiliza los valores almacenados en la matriz de transformación homogénea (atributo '**m_dMatriz**') y devuelve el resultado en coordenadas articulares (atributo '**m_sCoordTheta**'). Las ecuaciones del modelo inverso son distintas a las del proyecto [4] como se explica extensamente en el apartado 2 de esta memoria.

  Los parámetros son los siguientes:

  - **nBrazo:** Variable de tipo entero que especifica la posibilidad entre situar el robot en un punto con configuración en brazo derecho o en brazo izquierdo.
  - **nCodo:** Variable de tipo entero para configuración en codo arriba o codo abajo.
  - **nMuneca:** Variable de tipo entero que diferencia el sentido de giro de la muñeca entre sentido horario o antihorario.

- **void CrearMatrizDesplazada(Vector *vPuntos)**

  Esta función es muy parecida a la de **CrearMatriz(Vector *vPuntos)** de [4], cuyo cometido es el de crear a partir tres puntos (es el vector que se le pasa como parámetro) en coordenadas cartesianas la matriz de transformación homogénea de un sistema de referencia, calculado con dichos puntos, y referido al sistema de referencia ubicado en la base del manipulador.

  La nueva función **CrearMatrizDesplazada** tiene en cuenta el desplazamiento en el eje Z del sistema de referencia del extremo debido al cambio de dimensiones del propio efector final. Este desplazamiento se debe

a que el controlador lee la posición actual del extremo del manipulador teniendo en cuenta que éste mide 85mm, y manda dicho dato (en forma de punto en coordenadas cartesianas) a la aplicación que se ejecuta en el PC. Pero en realidad lo que a nosotros nos interesa es el punto dónde se encuentra la punta de la fresa, que serán las esquinas del stock matriz. Para ello hay que desplazar esta matriz 275mm. (longitud de la fresa) en la dirección del eje Z.
El parámetro que se le pasa es un array de tres variables tipo 'Vector'.

Como resultado se crea la matriz **m_dMatriz** del atributo **m_MundoReal**.

3.5.5. CDlgSimulacion.

Ésta es un clase creada especialmente en este proyecto para conectar las dos aplicaciones en las que se sustenta el corte automatizado de piezas, 'Corte3D.exe' y 'Rx90.exe'.

La función de esta clase es simplemente preguntar al operario mediante un cuadro de diálogo si desea simular visualmente la trayectoria del robot, y si la respuesta es afirmativa, se llama a la aplicación 'Rx90.exe'.

Esta clase no tiene atributos y sus métodos (funciones) son los siguientes:

- **CDlgSimulacion(CWnd* pParent /*=NULL*/)**
  Constructor estándar.

- **void OnOK()**

  Si el operario pulsa 'OK' se llama a la aplicación 'Rx90.exe'.