UNIVERSIDAD DE GUANAJUATO

CAMPUS IRAPUATO - SALAMANCA
DIVISIÓN DE INGENIERÍAS

"Diseño de un Manipulador Articulado (RRR) para seguimiento de trayectoria"

TESIS PROFESIONAL

QUE PARA OBTENER EL TÍTULO DE:

Ingeniero en Mecatrónica

PRESENTA:

Fernando González Caudillo

DIRECTORES:

Dr. Maximino Antonio González Palacios
Dr. Luz Antonio Aguilera Cortés

SALAMANCA, GTO. Noviembre, 2016

UNIVERSIDAD DE GUANAJUATO
Campus Irapuato - Salamanca
División de Ingenierías

Asunto: *Notificación de Dictamen de Propuesta de Tesis*

C. Fernando González Caudillo
Estudiante de la Licenciatura en Ingeniería Mecatrónica
PRESENTE.

Una vez concluido el análisis y discusión sobre la propuesta del proyecto de tesis titulada: **"Diseño de un Manipulador Articulado (RRR) para seguimiento de trayectoria "**, bajo la revisión del *Asesor(es):* González Palacios Maximino Antonio, Dr. y Aguilera Cortés Luz Antonio, Dr.; en su reunión ordinaria del semestre Enero-Junio 2015, la Comisión de Titulación del Consejo Divisional acordó la siguiente resolución:

> Con fundamento en las fracciones primera a cuarta del Artículo 67 y el Artículo 68 del Estatuto Académico, esta comisión acordó **POR UNANIMIDAD DE VOTOS ACEPTAR SU PROPUESTA DE TESIS** designando el siguiente jurado:
>
> ***Presidente:*** *Cervantes Sánchez Jesús, Dr.*
> ***Secretario:*** *Aguilera Cortés Luz Antonio, Dr.*
> ***Vocal:*** *Cerda Villafaña Gustavo, Dr.*

Se extiende la presente notificación de la resolución el 11 de Marzo del 2015

A T E N T A M E N T E
"LA VERDAD OS HARÁ LIBRES"
EL DIRECTOR

Dr. Roberto Rojas Laguna

UNIVERSIDAD DE GUANAJUATO
CAMPUS IRAPUATO SALAMANCA
DIVISION DE INGENIERIAS

c.c.p. Dr. Raúl Enrique Sánchez Yáñez – Secretario Académico de la División de Ingenierías
González Palacios Maximino Antonio, Dr. y Aguilera Cortés Luz Antonio, Dr. – Asesor(es)
Archivo

Agradecimientos

Primeramente doy gracias a mi **Dios**, ya que sin Él no habría podido lograr todo lo que soy y lo que tengo. Permitiéndome vivir y ver su gran amor que tiene por mí.

A mis padres **Fernando González** y **Edith Caudillo**, a mi hermana **Sarah**, que en todo momento me han cuidado, apoyado incondicionalmente en mis estudios, me han dado gran alegría a mi corazón; me han formado y educado como persona y me han acompañado a lo largo de mi vida. Pero principalmente por su amor para conmigo.

A la familia **Toscano del Ángel**, que desde muy chico los conozco, les doy las gracias porque a pesar de no ser de su familia directa, me han cuidado y amado como si fuera parte de ella. A Sabas y Celia, a su hija Lucia y a las pequeñas sobrinas Diana y Estephania que han sido de gran alegría.

A mis asesores **Dr. Max Antonio González** y **Dr. Luz Antonio Aguilera**, que me han mostrado interés y gran apoyo en el desarrollo de este trabajo, han sido muy pacientes y generosos con su tiempo y conocimientos en la ayuda, revisión y corrección de esta tesis. A todos mis maestros y doctores de la **Universidad de Guanajuato** que me han impartido clases, enseñanzas y sus conocimientos. Agradezco por su apoyo y dedicación en la formación académica que me dieron como estudiante, como ingeniero y como persona a lo largo de toda mi carrera.

A todos mis **compañeros** y **amigos** que han formado parte de mi desarrollo como estudiante, han sido de gran ayuda y amistad. Gracias por ser de tan agradable compañía.

Son muchas personas las cuales merecen mis agradecimientos, puesto que sin ellas no podría terminar esta etapa de mi vida, sin embargo, me faltaría espacio para todos. Por lo cual, agradezco a todos aquellos que participaron directa e indirectamente en la elaboración de esta tesis. ¡Muchas gracias a todos ustedes y que Dios los bendiga!

Resumen

Esta tesis presenta diversas simulaciones de trayectorias lineales y curvilíneas trazadas por un brazo robótico industrial, implementadas en una plataforma de desarrollo de software con aplicaciones de diseño y simulación de manipuladores seriales. El mayor desafío que se desea realizar con el manipulador articulado (6R), es que éste pueda desplazarse fácilmente en una trayectoria curvilínea en el espacio tridimensional, con una planificación adecuada de dicha trayectoria es posible realizar tareas industriales con este requerimiento. Debido a esta necesidad se aplica un método matemático para la planeación de trayectorias curvilíneas en un espacio tridimensional conocidas como Splines.

Las simulaciones gráficas de aplicaciones industriales en un entorno virtual permiten prever problemas y optimizar acciones adicionales a las ya planeadas. El entorno virtual utilizado para la simulación de las diversas aplicaciones es la plataforma `ADEFID`, dicha plataforma muestra el funcionamiento y ejecución de múltiples algoritmos de control para que el brazo robótico siga la trayectoria definida en una tarea industrial asignada.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Contenido

1.1 Robótica en la Industria.

La robótica está involucrada en el estudio de aquellas máquinas que pueden asistir a los seres humanos en la ejecución de alguna tarea, en lo que respecta tanto a la actividad física y la toma de decisiones. Dichas máquinas han generado gran aceptación en el medio industrial gracias a su capacidad de llevar a cabo gran número de tareas con rapidez y precisión, además de adaptarse a diversas situaciones con sólo modificar el algoritmo que comanda los movimientos de la máquina [1].

También la **Robótica** la definen como: "La ciencia que estudia los robots como sistemas que operan en algún entorno real, estableciendo algún tipo de conexión inteligente entre percepción y acción".

De acuerdo al *Robot Institute of America*, que se transformó después en la *Robot Industries Association (RIA*), define **robot** como:

> Un manipulador reprogramable multifuncional, diseñado para mover material, partes, herramientas o dispositivos especializados mediante movimientos programados variables para la ejecución de tareas diversas [2].

Las características importantes que aparecen en esta definición y que distinguen a un robot de otros dispositivos manipuladores, como máquinas automatizadas o similares, son la multifuncionalidad, es decir, el robot debe ser lo suficientemente versátil como para ejecutar tareas diversas, no previstas a priori por su diseñador, y la programabilidad, esto es, la capacidad de cambiar de una tarea a otra sin más que cambiar el programa (la secuencia de instrucciones) que debe ejecutar.

En la actualidad se está viviendo una etapa en el que casi cualquier forma de trabajo físico que el ser humano desarrolla se puede replicar de forma más rápida, con mayor precisión, más económica y más consistente por medio del uso de robots y mecanismos controlados por un sistema computarizado. Muchos puestos de trabajo altamente capacitados están completamente automatizados. Los trabajos de manufactura tales como fresado de metales, torneado, soldadura (ver Fig. 1.1) y ensamble ahora se llevan a cabo con más facilidad, más económica y más rápido utilizando máquinas

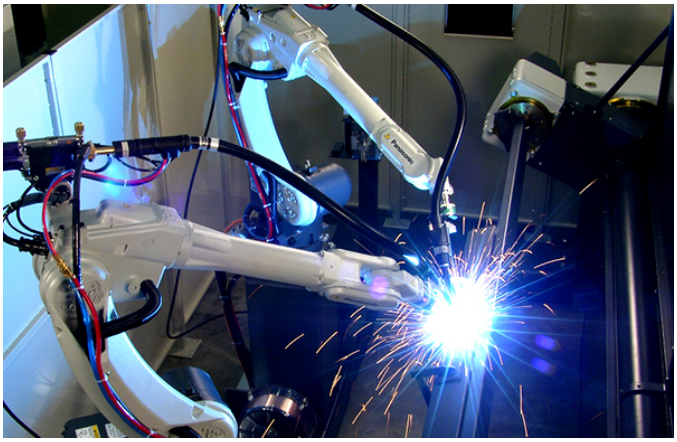


Figura 1.1: Robot industrial aplicando soldadura por arco [3].

de control numérico y robots industriales controlados por el uso fácil de software de CAD / CAM 3D. Los diseños para los componentes mecánicos se pueden crear de forma rápida en una pantalla de ordenador y convertirlos en un mundo real a prototipos de material sólido en menos de una hora, ahorrando así una gran cantidad de tiempo y costoso material que normalmente se desperdicia debido a un error humano.

Los robots industriales y máquinas están siendo utilizados para ensamblar, fabricar o pintar la mayoría de los productos de uso diario tales como placas madre para computadoras y periféricos de hardware, automóviles y todo tipo de electrodomésticos útiles que se encuentran en un hogar moderno. En el siglo XXI, los ingenieros han dominado casi todas las formas de control de movimiento y han demostrado que los robots y las máquinas pueden realizar casi cualquier trabajo que se considera demasiado pesado, agotador, aburrido o peligroso y dañino para los seres humanos.

1.2 Aplicaciones industriales en la robótica.

La versatilidad de mejora que diferencia entre un robot y una máquina-herramienta de control numérico llega a tener mucho peso en la industria, ya que con el efector final (EF) del robot se pueden integrar diferentes tipos de herramientas para realizar múltiples tareas en algún proceso de producción, así como manipular dentro de un gran espacio de trabajo. Los robots in-

dustriales presentan tres capacidades fundamentales que los hacen útiles en un proceso de fabricación: manipulación de materiales, manufactura, y medición [2].

En un proceso de fabricación, cada objeto tiene que ser transferido de un lugar de la fábrica a otro con el fin de ser almacenado, maquinado, ensamblado y empaquetado. Durante la transferencia, las características físicas del objeto no sufren ninguna alteración.

La capacidad del robot para recoger un objeto, moverlo en un espacio con rutas predefinidas y dejarlo en algún contenedor hace que el robot en sí mismo sea candidato ideal para las aplicaciones de manejo de materiales o **manipulación de materiales** (ver Fig. 1.2). Las aplicaciones típicas de la manipulación de materiales incluyen:

- Paletización (Colocación de las mercancías sobre paletas de forma automática y ordenada).
- Carga y descarga de bodegas y/o almacenes.
- Clasificación de materiales.
- Recoger y Colocar (Pick-and-Place).
- Empaquetado.

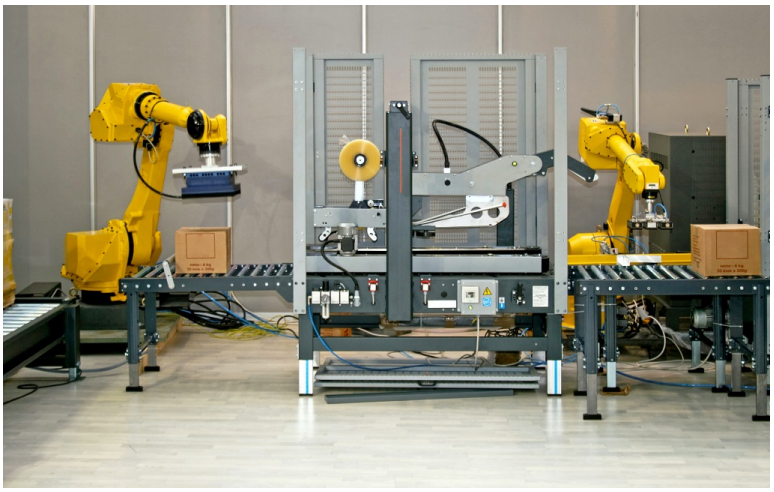


Figura 1.2: Robots con aplicaciones de manipulación de materiales [4].

El proceso de **manufactura** consiste en transformar materiales a partir de una materia prima en productos terminados; durante el proceso, el material cambia sus propias características físicas como resultado del maquinado o su diseño como resultado del ensamble con otras piezas, como se muestra en la Fig. 1.3. La capacidad del robot para manipular los objetos y herramientas lo hacen apto para ser empleado en muchas áreas de fabricación. Las aplicaciones comunes en el área de manufactura son:

- Soldadura por arco, soldadura por puntos.
- Recubrimiento de material y pintura.
- Sellado y adición de pegamento.
- Corte por láser y chorro de agua a presiones altas.
- Maquinado y Fresado.
- Acabados con pulido y desbardado.
- Atornillado, cableado y sujeción de equipos especializados.
- Ensamble de material mecánico y eléctrico.
- Ensamble de placas electrónicas.

Figura 1.3: Complejo industrial con producción automatizada [5].

Además del manejo y manipulado de material, en un proceso de manufactura es necesario realizar mediciones para comprobar la calidad del producto. La capacidad robótica de moverse y explorar en un espacio tridimensional, aunado a la capacidad de **medición**, hace un equipo robótico muy útil en la industria, como el que se muestra en la Fig. 1.4. Las aplicaciones típicas incluyen:

- Inspección de superficies.
- Detección de contorno del objeto.
- Detección de imperfecciones de fabricación.

Figura 1.4: Robot cartesiano con aplicaciones de medición [6].

Las aplicaciones enlistadas anteriormente describen el empleo actual de los robots como componentes de sistemas de automatización industrial. Y dada la necesidad de cubrir ciertas áreas en la industria manufacturera y la automotriz, se presenta un sistema virtual en el que el robot pueda simular aplicaciones y tareas industriales, en las que estén involucradas trayectorias curvilíneas y continuas.

1.3 Justificación.

Hoy en día las marcas líderes en manipuladores industriales, como es el caso de Fanuc, ABB, Mitsubishi, Kuka, Kawasaki, por mencionar algunas, proveen de software de simulación como se muestra en las figuras 1.5 y 1.6. Sin embargo, éstos están enfocados a sus propios productos, además los costos y mantenimiento de las licencias son elevados y sólo se justifica su compra cuando la empresa que los adquiere tiene grandes volúmenes de producción. Gracias a la flexibilidad de la plataforma desarrollada en `ADEFID`, se cuenta con un producto en el que es posible simular cualquier arquitectura de manipuladores seriales.

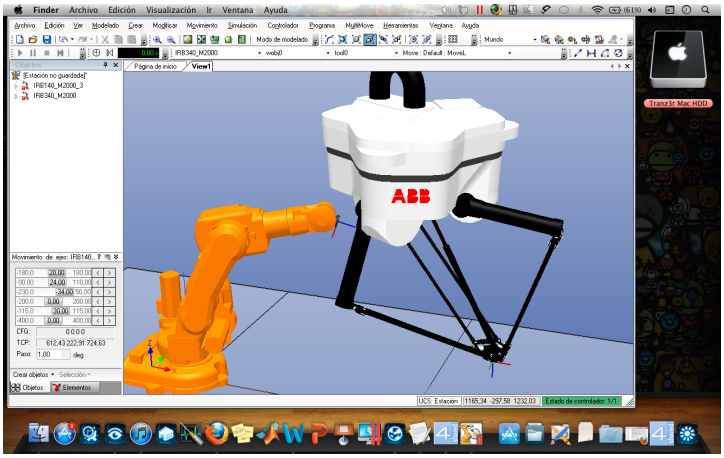


Figura 1.5: Simulador de ABB RobotStudio [7].

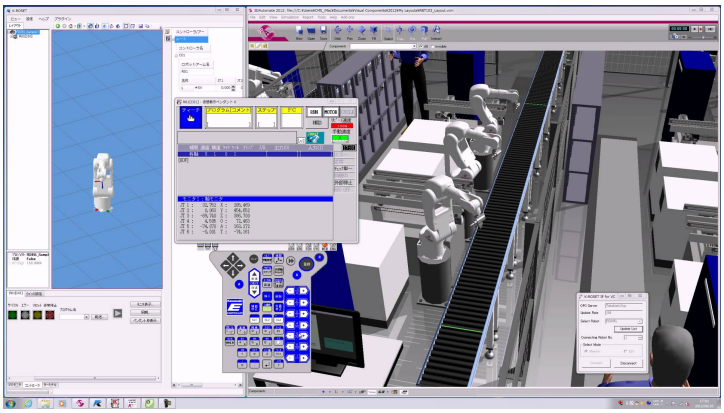


Figura 1.6: Simulador Kawasaki K-ROSET [8].

En la industria existen varias aplicaciones con el fin de lograr producción rápida, de calidad y de alta precisión. En las operaciones de pick-and-place, tales como la manipulación de pieza, montaje, paletización, etc., el efector final del manipulador tiene que viajar entre dos puntos específicos en el área de trabajo y el camino que se necesita en el medio no es una preocupación. En aplicaciones de seguimiento de trayectoria tales como soldadura, corte, pintura, etc. [9], el efector final tiene que seguir una trayectoria específica en un espacio tridimensional, ya sea por una función paramétrica curva o una secuencia de puntos establecidos para que generen una trayectoria continua, manteniendo al mismo tiempo la velocidad nominal tanto como sea posible. En este último caso, la planificación de la trayectoria puede ser compleja cuando hay limitaciones en la velocidad del efector final, la aceleración de articulación, y el error de trayectoria.

Una de las industrias que ha crecido grandemente gracias a la robótica, es la industria automotriz (ver Fig. 1.7). La gran producción en serie de los autos ha requerido la automatización. A mayor demanda de producción, es necesaria la simulación de diversas aplicaciones industriales con algoritmos de control en la manipulación de robots industriales que hagan tareas de planificación de trayectorias continuas de alta precisión, calidad y rapidez.

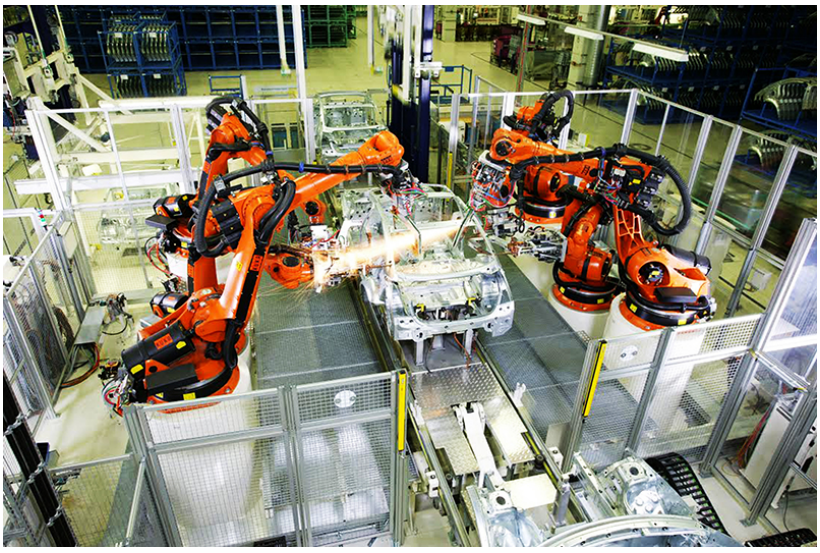


Figura 1.7: Industria automotriz en la actualidad [10].

Dada la necesidad de cubrir ciertas áreas en la industria automotriz y manufacturera, la justificación de este proyecto es desarrollar un sistema en el que el robot pueda simular aplicaciones y tareas industriales (ver Fig. 1.8), en las que estén involucradas trayectorias lineales y trayectorias curvilíneas continuas [11].

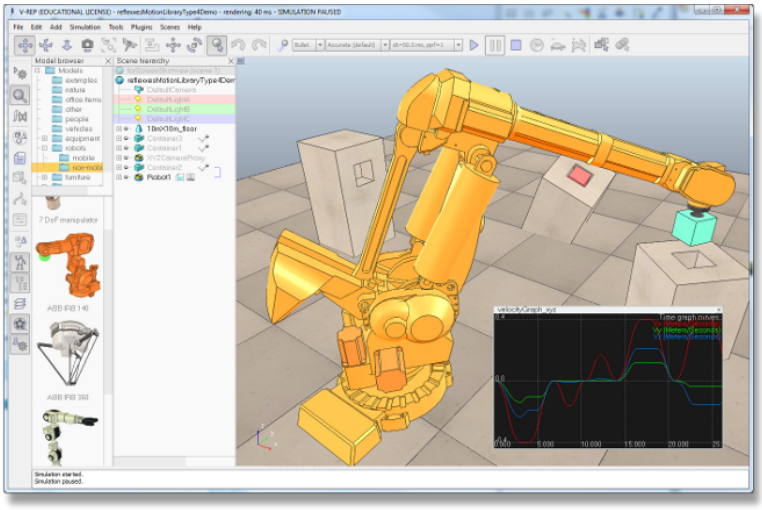


Figura 1.8: Software de simulación de V-REP [12].

1.4 Objetivos.

1.4.1 Objetivo general.

Crear un algoritmo y simulaciones de los Manipuladores Antropomórficos o Articulados (6R), que puedan desplazarse fácilmente en una trayectoria continua en el espacio tridimensional cartesiano, con aplicaciones en la manipulación industrial, tales como: soldadura, pintura, maquinado, corte, entre otros; desarrollados en la plataforma `ADEFID` [13].

1.4.2 Objetivos particulares.

La aplicación industrial propuesta es el corte por láser y/o soldadura por arco en las partes de la carrocería de un automóvil (ver Fig. 1.9), proceso importante en el sector automotriz. De manera que es necesario crear interpolaciones lineales y curvilíneas. Estas trayectorias continuas pueden ser

funciones paramétricas ya definidas geométricamente o trayectos que se forman con gran cantidad de puntos guía en el espacio. Para planificar dichas trayectorias en el espacio, las funciones Splines son las que se analizarán para dicha interpolación. Con el fin de llevar a cabo este proceso de simulación, es necesario especificar varios aspectos previos.

- Buscar algún manipulador industrial que cumpla con especificaciones cinemáticas, útiles para trazar trayectorias curvilíneas en grandes espacios de trabajo.
- Integrar los diseños CAD de los eslabones del robot y del automóvil a la plataforma `ADEFID`, esto servirá para darle una vista más real a la simulación.
- Analizar el funcionamiento de las funciones Splines, para la interpolación de trayectorias curvilíneas en un espacio tridimensional. Seleccionando los cálculos matemáticos necesarios para crear los algoritmos de control y movimiento en el lenguaje de programación que utiliza `MS Visual Studio` ®.
- Finalmente hacer las simulaciones de las trayectorias curvilíneas con el robot industrial, haciendo ver un proceso industrial en la plataforma `ADEFID`.

Figura 1.9: Robots soldando vehículos [14].

Capítulo 2

Modelo matemático

Contenido

En el presente capítulo se analizarán las diversas clasificaciones geométricas de los manipuladores seriales. Particularmente se desarrolla la cinemática de los manipuladores articulados, los cuales son los que se utilizarán para el presente trabajo.

2.1 Representación simbólica de los robots.

Los manipuladores robóticos están compuestos por una serie de **eslabones**, unidos entre ellos mediante articulaciones para formar una cadena cinemática. Las articulaciones son pares cinemáticos, las **articulaciones** de rotación son conocidas como **revolutas** y las articulaciones lineales son **prismáticas**. Una articulación de revoluta funciona como una bisagra, pues permite rotación relativa entre dos eslabones. Una articulación prismática permite el movimiento lineal relativo entre dos eslabones. Es así que se representan las articulaciones de revoluta mediante una R y las prismáticas mediante una P. Gráficamente, la representación de las articulaciones se pueden visualizar de la forma como se muestra en la Fig. 2.1. De este modo se puede representar a un robot de acuerdo a sus primeras tres articulaciones, de modo que si las tres primeras articulaciones de un robot son revolutas se puede decir que es un manipulador articulado RRR [15].

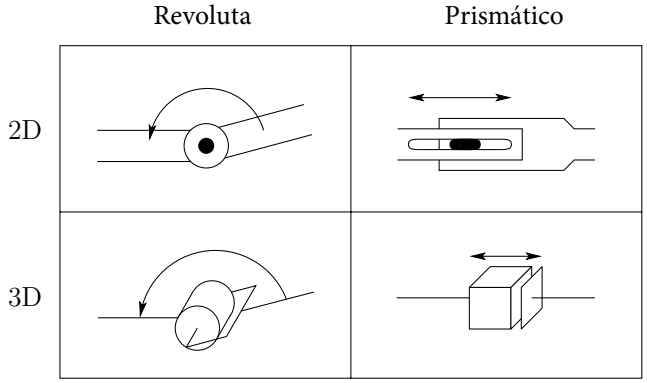


Figura 2.1: Representación simbólica de las articulaciones del robot [15].

Cada articulación representa la unión entre dos eslabones. Al eje de rotación de una articulación de giro, o el eje a lo largo de una articulación prismática, según sea el caso se define por z_i entre la unión de los eslabones i e $i+1$. Al desplazamiento relativo entre dos eslabones se le llama **variable de articulación**, y se representa mediante θ si es una articulación de revoluta o b si se trata de una articulación prismática como se muestra en la Fig. 2.2.

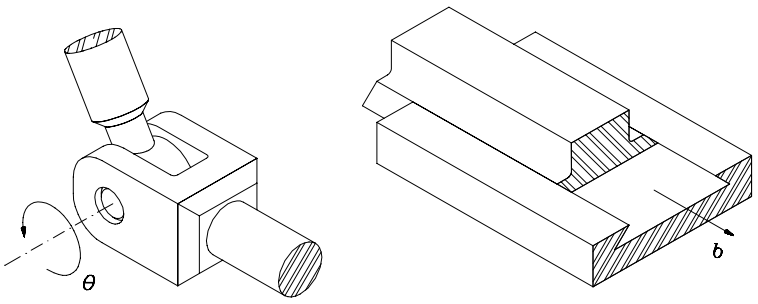


Figura 2.2: Articulaciones tipo Revoluta y Prismática [20].

2.2 Espacio de configuración del manipulador.

La configuración de un manipulador es la completa especificación de la localización de cada punto del manipulador. El conjunto de todas las configuraciones posibles se denomina espacio de configuración. La representación de una configuración del robot se dará como un conjunto de variables de articulación. Este conjunto se representa mediante la letra q y las variables de articulación para la configuración descrita estarán dadas por $q_1, q_2, \ldots, q_n$, de modo que para una articulación de revoluta $q_i = \theta_i$, mientras que para una articulación prismática $q_i = b_i$ [15].

Se dice que un cuerpo tiene n **grados de libertad** (GDL), si la forma mínima de representar completamente su configuración está dada en función de n parámetros. Entonces, el número de GDL es igual a la dimensión del espacio de configuración del manipulador. En un manipulador robótico el número de GDL está determinado por el número de articulaciones que lo componen. Un cuerpo rígido como el mostrado en la figura 2.3 en el espacio

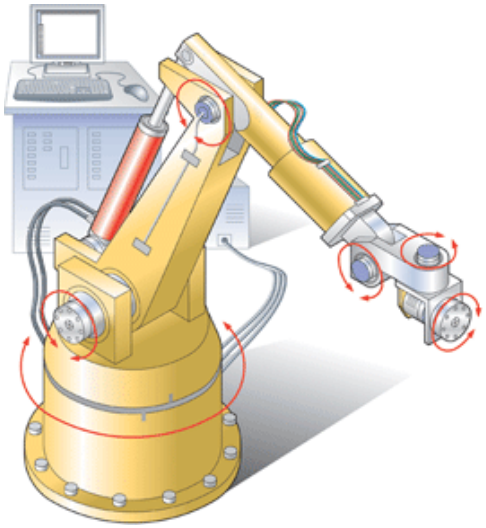


Figura 2.3: Manipulador robótico con 6 GDL [16].

tridimensional posee 6 GDL: tres para especificar su **posición** y tres para para su **orientación**. Por tal motivo, un manipulador industrial debería poseer por lo menos seis grados de libertad. Ya que con menos de seis GDL el manipulador no puede llegar a cualquier punto de su entorno de trabajo con orientación arbitraria. Ciertas aplicaciones tales como alcanzar alrededor o detrás de obstáculos pueden requerir más de seis GDL. Sin embargo, la dificultad de controlar un manipulador aumenta rápidamente con el número de articulaciones.

2.3 Espacio de trabajo del manipulador.

El espacio de trabajo de un manipulador es el volumen total que puede alcanzar el efector final cuando el robot realiza todos los posibles movimientos permitidos. Este espacio está limitado por la geometría del manipulador así como las limitaciones mecánicas de las articulaciones (ver Fig. 2.4). En general se divide en dos tipos. El espacio de trabajo es accesible cuando todo el conjunto de puntos es alcanzado por el manipulador, mientras que el espacio de trabajo diestro consiste en aquellos puntos que el manipulador puede alcanzar con una orientación arbitraria con el efector final.

Considerando aplicaciones automotrices, se optó por un manipulador in-

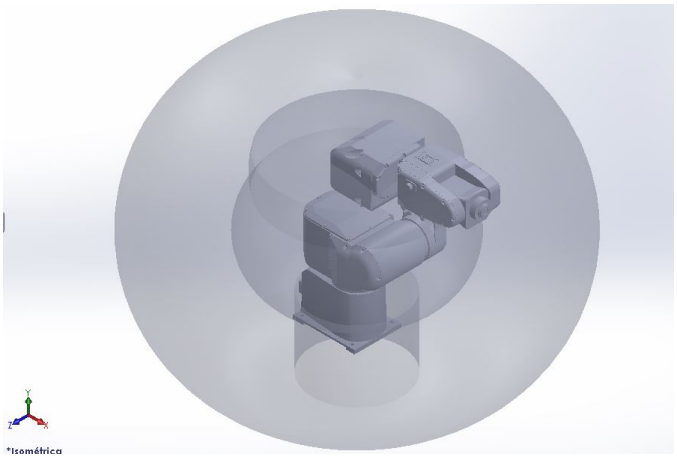


Figura 2.4: Espacio de trabajo del robot FanucLR-Mate200IB.

dustrial articulado de gran alcance en su espacio de trabajo, es así que el robot Kawasaki BX100N fue uno de los seleccionados, ya que tiene dimensiones ideales para tales aplicaciones industriales, como un alcance máximo de 2.2 m (ver Fig. 2.5) y puede utilizarse en tareas de corte o soldadura para realizar múltiples trayectorias en la fabricación de un automóvil.

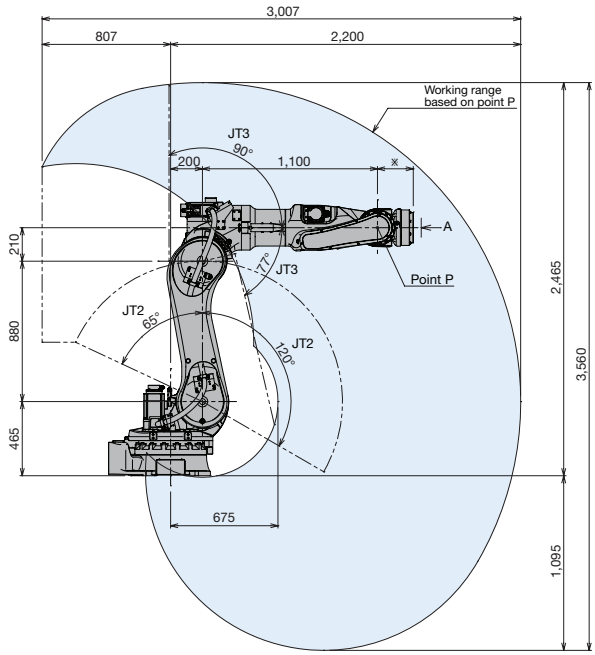


Figura 2.5: Manipulador Kawasaki BX100N [17].

2.4 Clasificación de los manipuladores robóticos.

Los manipuladores robóticos pueden ser clasificados de acuerdo a varios criterios, tales como su fuente de energía, la forma en que se unen sus articulaciones, su geometría, su estructura cinemática, su área de aplicación o el método de control que utilizan [15]. Esta clasificación es útil principalmente con el fin de determinar qué robot es el adecuado para una tarea determinada. Por ejemplo, un robot hidráulico no sería adecuado para la manipulación de alimentos o aplicaciones donde la limpieza es primordial, muchas aplicaciones y tareas descritas por los manipuladores se muestran en el capítulo anterior. En esta sección se hablará un poco sobre la clasificación de acuerdo a la geometría.

2.5 Clasificación geométrica de los manipuladores.

Muchos manipuladores industriales en la actualidad tienen seis o menos GDL. Estos manipuladores son usualmente clasificados cinemáticamente con base en sus tres primeras articulaciones del brazo, donde su muñeca de articulación se describe por separado. Se conocen cinco clasificaciones básicas dentro de las cuales recaen, casi en su totalidad, los robots industriales de hoy en día: **articulado (RRR)**, **esférico (RRP)**, **SCARA (RRP)**, **cilíndrico (RPP)** y **cartesiano (PPP)** [15].

Cada uno de estos cinco brazos manipuladores son robots seriales. Una sexta clase distinta de manipuladores consiste en los llamados robots **paralelos** (ver Fig. 2.6). En un manipulador paralelo los eslabones están arreglados en una cadena cinemática cerrada o no serial.

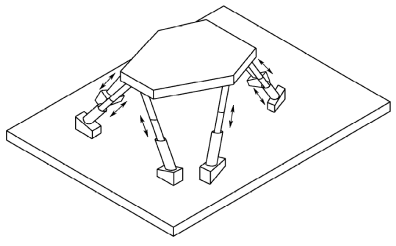
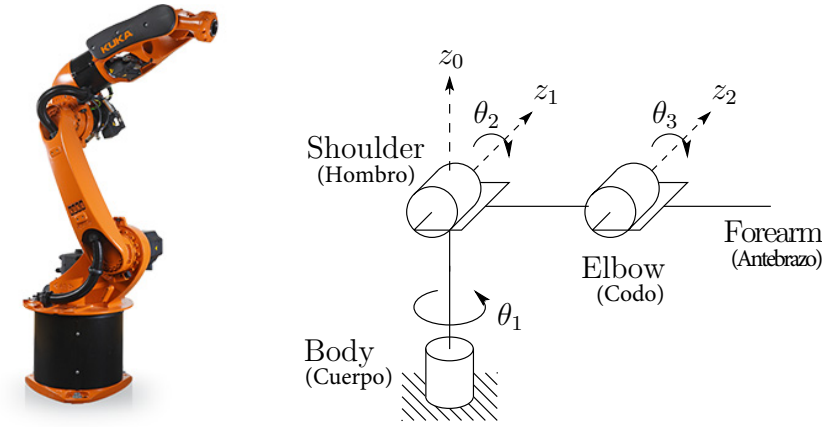


Figura 2.6: Manipulador paralelo [2].

2.5.1 Manipulador articulado (RRR).

El manipulador articulado también conocido como revoluta, o manipulador antropomórfico se muestra en la figura 2.7a. Este tipo de manipuladores es de los más utilizados para el diseño de robots industriales. En los manipuladores antropomórficos las primeras tres articulaciones se denominan como cuerpo, brazo superior, y antebrazo, como se muestra en la Fig. 2.7b. Los ejes de las articulaciones se designan como cintura (z_0), hombro (z_1), y codo (z_2). Normalmente, el eje de articulación z_2 es paralelo a z_1 y ambos ejes, z_1 y z_2 son perpendiculares a z_0.



(a) KUKA KR16 arc HW [18].

(b) Representación simbólica de un manipulador articulado RRR [15].

Figura 2.7: Representación de un manipulador articulado.

El espacio de trabajo de un manipulador articulado se muestra en la figura 2.8. El manipulador provee relativamente gran libertad de movimiento en un espacio compacto.

2.5.2 Manipulador esférico (RRP).

Mediante la sustitución de la tercer articulación o del codo en el manipulador articulado por una articulación prismática se obtiene el manipulador esférico, haciendo ahora que el eje z_2 sea perpendicular a z_1, como se muestra en la figura 2.9a. El término de manipulador esférico deriva del hecho de que las coordenadas de articulación coinciden con las coordenadas esféricas

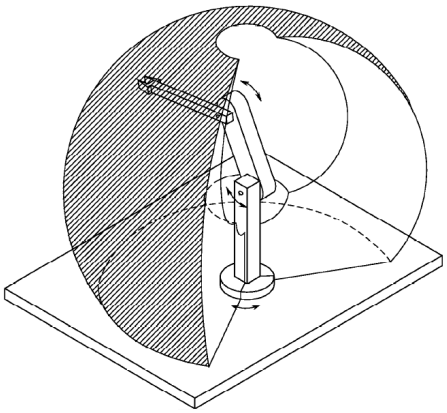
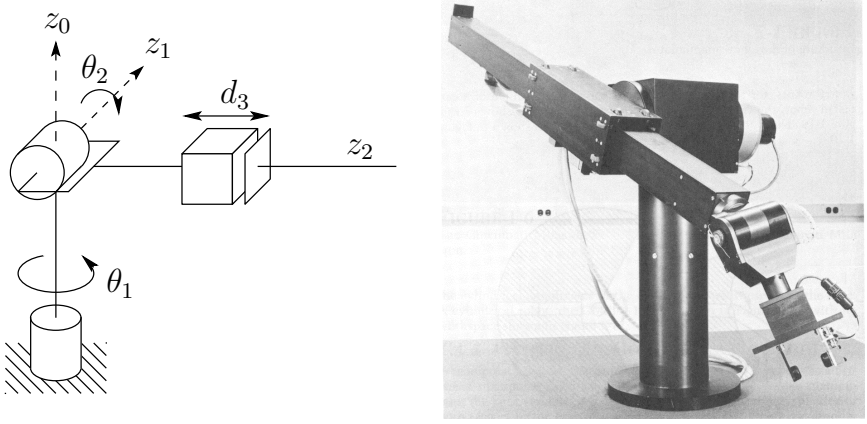


Figura 2.8: Espacio de trabajo de un manipulador articulado [2].

del efector final con relación a un sistema de coordenadas situadas en la articulación del hombro. La figura 2.9b muestra el brazo Stanford, uno de los robots esféricos más conocidos. El espacio de trabajo de un manipulador esférico se muestra en la figura 2.10.



(a) Representación simbólica de un manipulador RRP.

(b) Robot Stanford.

Figura 2.9: Representación de un manipulador esférico [15].

2.5.3 Manipulador SCARA (RRP).

El manipulador **SCARA** (del inglés **S**elective **C**ompliant **A**rticulated **R**obot for **A**ssembly), que se muestra en la figura 2.11a, es una configuración que

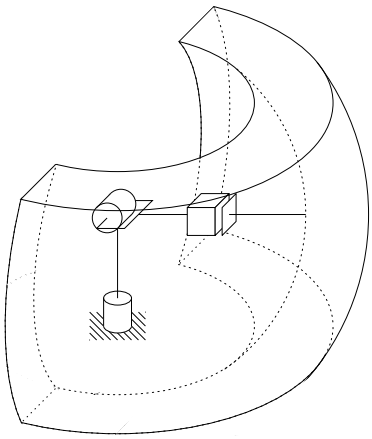
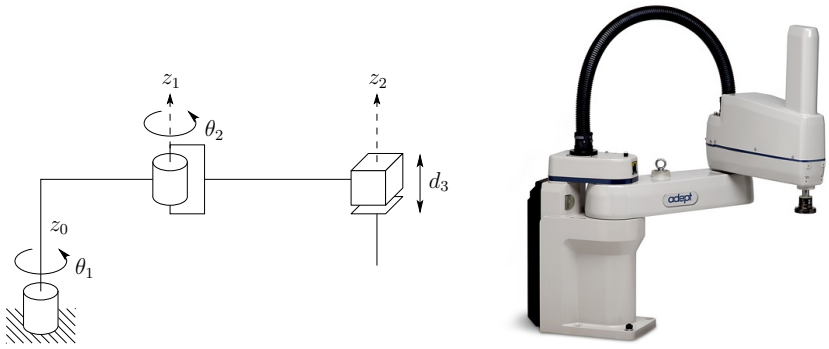


Figura 2.10: Espacio de trabajo de un manipulador esférico [15].

ha adquirido popularidad en la industria y como su nombre lo describe, fue pensado para ser utilizado en tareas de ensamble. A pesar de que esta configuración es de la forma RRP, presenta diferencias notables con respecto al manipulador esférico que también es RRP. La diferencia más notable radica en el hecho de que para un manipulador SCARA los ejes z_1, z_2 y z_3 son paralelos entre sí. La Fig. 2.11b muestra un manipulador SCARA Adept. El espacio de trabajo de un manipulador SCARA se muestra en la figura 2.12.



(a) Representación simbólica de un manipulador SCARA.

(b) Robot Adept Cobra 600.

Figura 2.11: Representación de un manipulador SCARA [15].

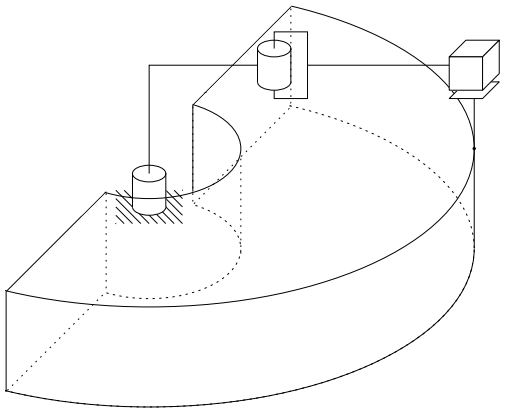
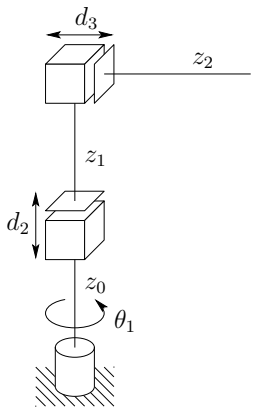


Figura 2.12: Espacio de trabajo de un manipulador SCARA [15].

2.5.4 Manipulador cilíndrico (RPP).

La primera articulación de este manipulador es una articulación de revoluta, lo que produce una rotación sobre la base, mientras que las dos articulaciones siguientes son prismáticas. Su nombre se debe a que las variables de articulación serán el valor de la posición del efector final en coordenadas cilíndricas. Un ejemplo de esta configuración se muestra en la figura 2.13a.



(a) Representación simbólica de un robot cilíndrico.

(b) Seiko RT3300.

Figura 2.13: Representación de un manipulador cilíndrico [15].

Los manipuladores cilíndricos son útiles para aplicaciones de montaje sobre una mesa y, como los robots gantry, para la transferencia de materiales

y carga. Un ejemplo de un robot cilíndrico, se muestra en la figura 2.13b. El espacio de trabajo de un manipulador cilíndrico se ilustra en la figura 2.14.

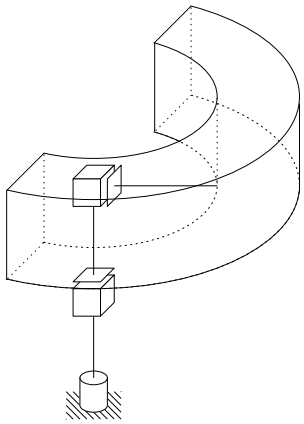


Figura 2.14: Espacio de trabajo de un manipulador cilíndrico [15].

2.5.5 Manipulador cartesiano (PPP).

Un manipulador cuyas primeras tres articulaciones son prismáticas se conoce como un manipulador cartesiano. Este tipo de manipulador es el de análisis más sencillo pues las variables de articulación serán las coordenadas del efector final. Son utilizados principalmente en tareas de ensamble y medición. La figura 2.15 muestra de manera gráfica como está conformado un manipulador cartesiano. Y el espacio de trabajo de un manipulador cartesiano se muestra en la figura 2.16.

2.6 Muñeca y efector final del manipulador.

Las articulaciones de la cadena cinemática entre el brazo y el efector final se conocen como la **muñeca**. Las articulaciones de la muñeca son casi siempre revolutas. Cada vez es más común diseñar manipuladores con las **muñecas esféricas**, lo que se refiere con este tipo de muñecas es que los tres ejes de articulación se intersectan en un punto común [15]. La muñeca esférica está representada simbólicamente en la figura 2.17.

La muñeca esférica simplifica enormemente el análisis cinemático, lo que permite separar el posicionamiento y la orientación del efector final. Normalmente, el manipulador posee tres grados de libertad para la posición, que

(a) Representación simbólica de un manipulador cartesiano.

(b) Robot Epson.

Figura 2.15: Representación de un manipulador cartesiano [15].

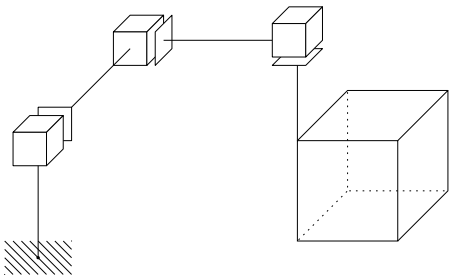
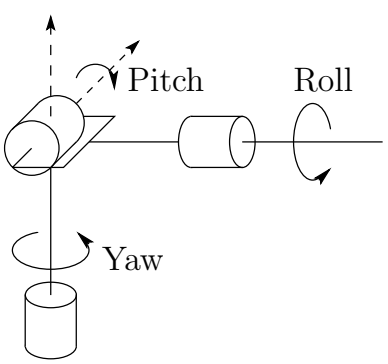


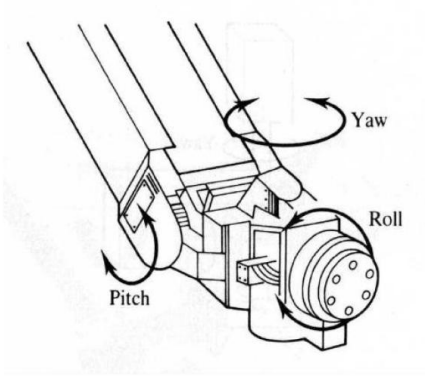
Figura 2.16: Espacio de trabajo de un manipulador cartesiano [15].

son producidos por tres o más articulaciones en el brazo. El número de grados de libertad para la orientación dependerá entonces de las articulaciones de la muñeca.

Los conjuntos de brazo y de la muñeca de un robot se utilizan principalmente para colocar en el extremo del efector final cualquier herramienta que pueda llevar. El efector final o la herramienta es quien realiza realmente el trabajo. El tipo más simple de efectores finales son las pinzas, las cuales por lo general son capaces de sólo dos acciones, abrir y cerrar. Se utilizan principalmente para la transferencia de materiales, algunos manipuladores requieren de herramientas más complejas para realizar tareas, tales como soldadura, corte por láser, montaje, pulido, etc. Para la aplicación del presente proyecto se requiere una herramienta que pueda hacer la función de corte, ya sea por láser o chorro de agua a grandes presiones, como se muestra



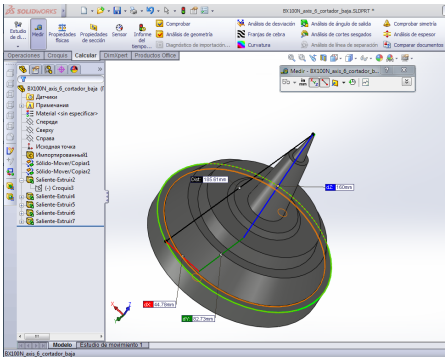
(a) Representación simbólica de muñeca esférica [15].



(b) Muñeca de un robot.

Figura 2.17: Representación de la muñeca de un robot.

en la figura 2.18a. Para este propósito se diseñó un efector final en el que el manipulador pueda cortar algún material con el trazo de una trayectoria predeterminada, es así, que se basó en un diseño de algunos cortadores como se muestra en la figura 2.18b.



(a) Diseño CAD de cortadora.

(b) Cortadora láser [19].

Figura 2.18: Herramienta del EF como cortador láser.

2.7 Cinemática del manipulador.

El problema de la cinemática consiste en describir el movimiento del manipulador sin la consideración de las fuerzas y pares de torsión que provocan el movimiento. Por consiguiente, la descripción cinemática es de forma geométrica. Considerando en primer lugar el problema de la cinemática directa, que consiste en determinar la posición y orientación del efector final dados los valores de las variables conjuntas del robot. El problema de la cinemática inversa es determinar los valores de articulación dados la posición y orientación del efector final [15].

Como ya se ha mencionado, los pares cinemáticos más utilizados en los robots industriales, son los pares prismáticos y de revoluta. Estos pares cinemáticos cuentan con un solo grado de libertad de movimiento: el ángulo de rotación en el caso de un par de revoluta, y la magnitud del desplazamiento lineal en caso del par prismático. Con la suposición de que cada articulación tiene un solo grado de libertad, la acción de cada articulación puede ser descrita por un número real. El objetivo de la cinemática directa es determinar el efecto acumulativo del conjunto que forman las variables de articulación, esto es, determinar la posición y orientación del efector final dados los valores de dichas variables. En contraste, la cinemática inversa busca determinar el valor de las variables de articulación conociendo la posición y orientación del efector final.

Un manipulador robótico de n articulaciones cuenta con $n+1$ eslabones, pues cada articulación conecta 2 eslabones. Por consiguiente las articulaciones se numeran de 1 a n, mientas que los eslabones se numeran de 0 a n, comenzando con la base. Con esta convención, la articulación i conecta al eslabón $i-1$ con el eslabón i. Con la $i-$ésima articulación, se asocia una variable de articulación, representada como q_i. En el caso de una articulación de revoluta q_i es el ángulo de rotación, en el caso de una articulación prismática q_i es el desplazamiento lineal de la articulación como se muestra en la ecuación (2.1).

$$q_i = \begin{cases} \theta_i & \text{si la articulación } i \text{ es revoluta} \\ b_i & \text{si la articulación } i \text{ es prismática} \end{cases} \tag{2.1}$$

Al llevar a cabo el análisis cinemático se fija un sistema de referencia a cada eslabón. En general el sistema coordenado $o_i x_i y_i z_i$ es fijo al eslabón i. El sistema coordenado $o_i x_i y_i z_i$, que está unido a la base del robot se conoce como el sistema de referencia inercial. La figura 2.19 ilustra la idea de unir los sistemas de coordenadas de cada eslabón de manera rígida a los eslabones de un manipulador articulado.

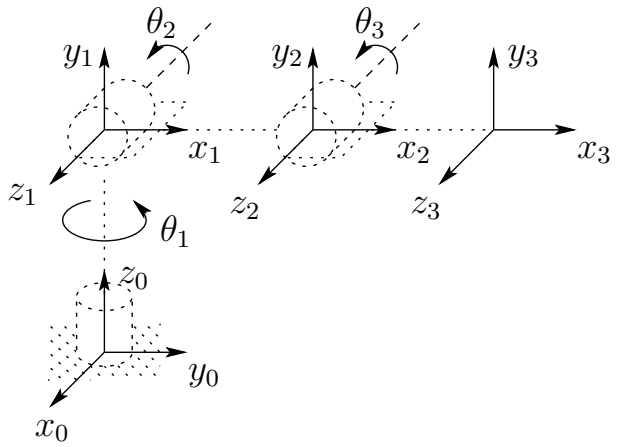


Figura 2.19: Cuadro de coordenadas de un manipulador articulado [15].

Por otra parte, se supone que A_i es la matriz de transformación homogénea que expresa la posición y orientación de $o_i x_i y_i z_i$ con respecto a $o_{i-1} x_{i-1} y_{i-1} z_{i-1}$. La matriz A_i no es constante, conforme cambia la configuración del robot. Sin embargo, como en estos robots se manejan únicamente articulaciones prismáticas y/o de revoluta, A_i es función de una sola variable, la variable de articulación q_i, como se muestra en la ecuación (2.2).

$$A_i = A_i(q_i) \tag{2.2}$$

La matriz de transformación homogénea que expresa la posición y orientación de $o_j x_j y_j z_j$ con respecto a $o_i x_i y_i z_i$ se llama, por convención, una matriz de transformación, y se denota por T_j^i (Ec. 2.3):

$$T_j^i = \begin{cases} A_{i+1} A_{i+2} \dots A_{j-1} A_j & \text{si } i < j \\ I & \text{si } i = j \\ (T_i^j)^{-1} & \text{si } j > i \end{cases} \tag{2.3}$$

De manera que se unen rígidamente los diversos marcos para las articulaciones correspondientes, se deduce la posición de cualquier punto del efector final, expresado por el parámetro n, que es una constante independiente de la configuración del robot. Se denota la posición y orientación del efector final con respecto al sistema de referencia inercial o base por o_0 representado por tres vectores (en los que se dan las coordenadas del origen o base del robot hasta el marco del efector final del mismo) y la matriz de rotación R_n^0 de 3×3, que definen la matriz de transformación homogénea por la ecuación (2.4).

$$H = \begin{bmatrix} R_n^0 & o_n^0 \\ 0 & 1 \end{bmatrix} \tag{2.4}$$

Entonces la posición y orientación del efector final en el marco inercial es:

$$H = T_n^0 = A_1(q_1) \cdots A_n(q_n) \tag{2.5}$$

Cada matriz de transformación A_i está formada de la siguiente manera:

$$A_i = \begin{bmatrix} R_i^{i-1} & o_i^{i-1} \\ 0 & 1 \end{bmatrix} \tag{2.6}$$

Por lo tanto

$$T_j^i = A_{i+1} \cdots A_j = \begin{bmatrix} R_j^i & o_j^i \\ 0 & 1 \end{bmatrix} \tag{2.7}$$

La matriz R_j^i mostrada, en la ecuación (2.8), expresa la orientación del sistema de coordenadas $o_j x_j y_j z_j$ respecto a $o_i x_i y_i z_i$ y está dado por las partes de rotación de todas las matrices A (ecuaciones (2.5) a (2.7)), como se muestra a continuación:

$$R_j^i = R_{i+1}^i \cdots R_j^{j-1} \tag{2.8}$$

Las coordenadas vectores o_j^i se dan de forma recursiva por la ecuación (2.8).

$$o_j^i = o_{j-1}^i + R_{j-1}^i o_j^{j-1} \tag{2.9}$$

2.8 Convención de Denavit-Hartenberg, DH.

La convención Denavit-Hartenberg, DH, es una nomenclatura utilizada para describir la arquitectura de un manipulador robótico. Desde que se dio a conocer esta nomenclatura a la fecha, diversas variantes de la misma han ido surgiendo, siendo tres las más utilizadas: la convención de Denavit-Hartenberg original, la variante distal y la variante proximal. Para el desarrollo del presente trabajo se utilizó la variante distal de la convención de Denavit-Hartenberg.

Principalmente la convención de Denavit-Hartenberg se realiza para seleccionar la correcta ubicación de los marcos de sistemas coordenados que están unidos a cada eslabón. Además permite obtener una descripción de la geometría del manipulador y generar las matrices de transformación homogénea para especificar cualquier punto dentro de la geometría del manipulador.

En esta convención cada transformación homogénea A_i es representada como el producto de cuatro transformaciones básicas [15]:

$$A_i = \text{Rotación}_{z,\theta_i}, \text{Traslación}_{z,b_i}, \text{Traslación}_{x,a_i}, \text{Rotación}_{x,\alpha_i} \tag{2.10}$$

Para simplificar un poco la notación, se utilizará $sin\theta = s\theta$, $cos\theta = c\theta$, con el motivo de reducir las expresiones de funciones trigonométricas. Por lo tanto, la ecuación (2.10), se desarrolla de la siguiente forma:

$$\begin{aligned} A_i &= \begin{bmatrix} c\theta_i & -s\theta_i & 0 & 0 \\ s\theta_i & c\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & b_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &\times \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_i & -s\alpha i & 0 \\ 0 & s\alpha i & c\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{2.11}$$

Y llevando a cabo la multiplicación de matrices se obtiene:

$$A_i = \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & b_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.12}$$

Donde las cuatro cantidades θ_i, a_i, b_i y α_i son los parámetros asociados al eslabón i y la articulación i [15], y serán llamados **parámetros DH**. También el parámetro θ_i es conocido como ángulo de rotación, el parámetro b_i se conoce como desviación, el parámetro a_i es llamado distancia y el parámetro α_i se conoce como ángulo de torsión. De la ecuación (2.2) se sabe que la matriz de transformación homogénea A_i, es función de una sola variable, esto implica que tres parámetros de la ecuación (2.12) son constantes y el restante es la variable de articulación. En el caso de una articulación de revoluta la variable de articulación es θ_i, de modo que a_i, b_i y α_i son constantes, mientras que en el caso de una articulación prismática la variable de articulación es b_i y los otros tres parámetros son constantes.

2.9 Asignación de ejes coordenados.

Es necesario mencionar que no se puede representar todo movimiento de un cuerpo rígido mediante el uso de sólo cuatro parámetros. Para que se pueda representar un movimiento de un cuerpo rígido de la forma que lo expresa la ecuación (2.10) es necesario que exista una sola matriz de transformación homogénea para ir del marco de referencia $o_i x_i y_i z_i$ al $o_{i-1} x_{i-1} y_{i-1} z_{i-1}$, la única forma de que esto pase es que exista una relación bien definida entre los marcos sobre los cuales se llevará a cabo la transformación. Esta relación está dada por las siguientes dos condiciones [15]:

Características DH de los ejes coordenados

(DH1) El eje x_i es perpendicular al eje z_{i-1}.

(DH2) El eje x_i intersecta al eje z_{i-1}.

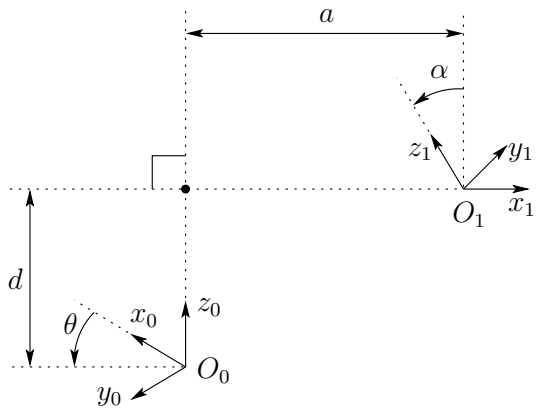


Figura 2.20: Cuadro de ejes coordenados con los parámeros DH1 y DH2 [15].

Estas dos propiedades se muestran en la figura 2.20. Bajo estas condiciones, se afirma que existen números únicos de a, b, θ y α. El parámetro a es la distancia entre los ejes z_0 y z_1, y se mide a lo largo del eje x_1. El ángulo α es el ángulo entre los ejes z_0 y z_1, medido en un plano normal a x_1. El sentido positivo de α se determina a partir de z_0 a z_1 por la regla de la mano derecha como se muestra en la figura 2.21. El parámetro d es la distancia del origen o_0 a la intersección de los ejes x_1 y z_0 medida a lo largo del eje z_0. Finalmente, θ es el ángulo entre los ejes x_0 a x_1 medido en un plano normal a z_0.

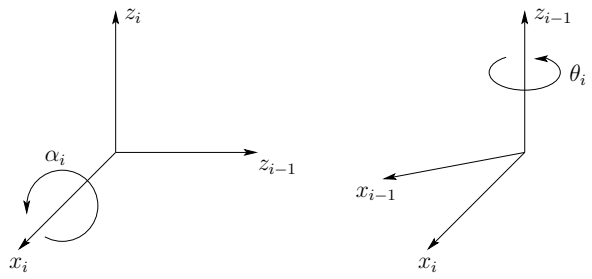


Figura 2.21: Sentido positivo para parámetros α_i y θ_i [15].

Para cualquier robot manipulador dado, se pueden escoger los sistemas coordenados $0, 1, 2, \ldots, n$ de tal manera que se satisfagan las dos condiciones. La convención Denavit-Hartenberg también define una serie de pasos para

asignar los marcos de referencia de modo que las condiciones necesarias sean satisfechas. En primer lugar se deben asignar los ejes $z_0, z_1, z_2, \ldots, z_{n-1}$. La asignación de estos ejes es arbitraria, sin embargo, siempre se busca que z_i sea el eje de acción de la articulación $i+1$. De esta forma z_0 es el eje de acción de la articulación 1, z_1 es el eje de acción de la articulación 2, etc.

En este punto hay dos casos a considerar: (i) La articulación $i+1$ es una revoluta, z_i es el eje de revolución de la articulación $i+1$; (ii) La articulación $i+1$ es prismática, z_i es el eje de traslación de la articulación $i+1$ [15].

Una vez que se han establecido todos los ejes z, se establece el marco base. El origen o_0 puede estar situado en cualquier punto sobre el eje z_0. Por lo general, se establece que el origen del marco base esté ubicado en la base del manipulador. Una vez que se ha seleccionado el lugar donde estará el origen o_0, se escoge la ubicación de x_0, y_0 de la manera que sea más conveniente y que se obtenga un sistema que siga la regla de la mano derecha. Con esto queda completamente establecido el marco de la base.

A continuación se definirán los sistemas coordenados restantes. El sistema i deberá ser establecido con base en el sistema $i-1$, y deberá comenzarse con el sistema 1, (ver Fig. 2.22). Para comenzar a ubicar el marco i es necesario considerar tres casos:

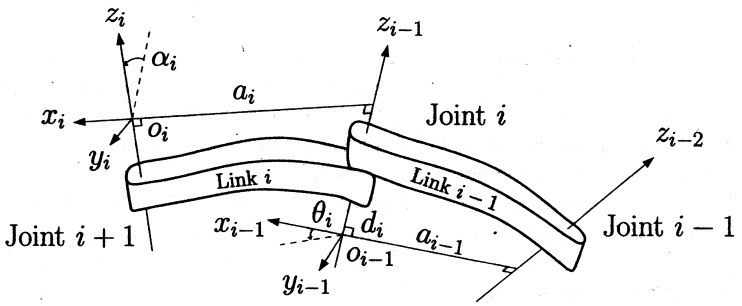


Figura 2.22: Asignación de parámetros en el marco DH [15].

1. **Los ejes z_{i-1} y z_i no son coplanares.**
   Si z_{i-1} y z_i no son coplanares, entonces existe un solo segmento de recta que es perpendicular a ambos z_{i-1} y z_i. La línea que contiene esta normal común a z_{i-1} y z_i define a x_i, y el punto donde esta recta

intersecta a z_i es el origen o_i como se muestra en la figura 2.23. Por último el eje y_i se coloca siguiendo la regla de la mano derecha.

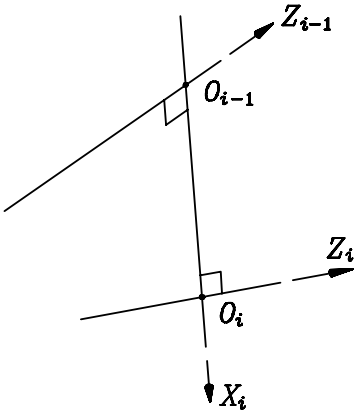


Figura 2.23: Los ejes z_{i-1} y z_i no son coplanares [20].

2. **Los ejes z_{i-1} y z_i se intersectan.**
   Si los ejes z_{i-1} y z_i se intersectan, en este caso x_i se escoge normal al plano formado por z_{i-1} y z_i. Esta situación se ejemplifica en la figura 2.24. La dirección positiva del eje es arbitraria. El eje y_i se asigna siguiendo la regla de la mano derecha.

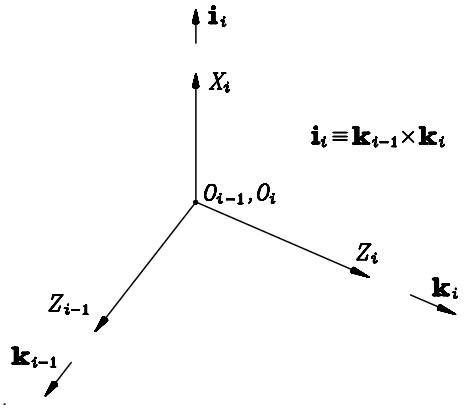


Figura 2.24: Los ejes z_{i-1} y z_i se intersectan [20].

3. **Los eje z_{i-1} y z_i son paralelos.**
   Si los ejes z_{i-1} y z_i son paralelos, existen un número infinito de segmentos de recta que son normales a z_{i-1} y z_i. En este caso el origen o_i se coloca en cualquier punto a lo largo de z_i. El eje x_i se coloca a partir del origen o_i en dirección perpendicular hacia z_{i-1}. Por lo general se selecciona la normal que pasa por o_{i-1} como x_i, de modo que o_i está en el punto donde esta normal intersecta a z_i. Este caso se muestra en la figura 2.25. Nuevamente y_i vuelve a ser seleccionado de acuerdo a la regla de la mano derecha.

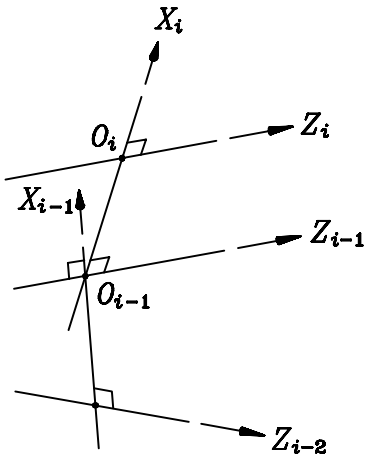


Figura 2.25: Los eje z_{i-1} y z_i son paralelos [20].

Cabe mencionar que estos métodos para ubicar los marcos de referencia de cada eslabón, fueron diseñados para cumplir con las condiciones necesarias para que todas las transformaciones homogéneas entre cada marco i con el marco $i-1$ sea de la forma de la ecuación (2.10).

2.10 Parámetros DH del robot Kawasaki BX100N.

Para llevar a cabo el análisis de la cinemática del manipulador articulado de 6 grados de libertad, tipo Kawasaki BX100N es necesario utilizar la tabla DH 2.1 que se muestra a continuación.

Mediante la observación del sistema coordenado mostrado en las figuras 2.26 y 2.27, y las medidas reales del manipulador Kawasaki BX100N

Tabla 2.1: Tabla de parámetros DH para un manipulador de 6 GDL

Eslabón	θ_i	a_i	b_i	α_i
1	θ_1^*	a_1	b_1	α_1
2	θ_2^*	a_2	b_2	α_2
3	θ_3^*	a_3	b_3	α_3
4	θ_4^*	a_4	b_4	α_4
5	θ_5^*	a_5	b_5	α_5
6	θ_6^*	a_6	b_6	α_5

(mostradas en el apéndice **A**) se pueden obtener los siguientes datos para generar la tabla 2.2 de parámetros DH.

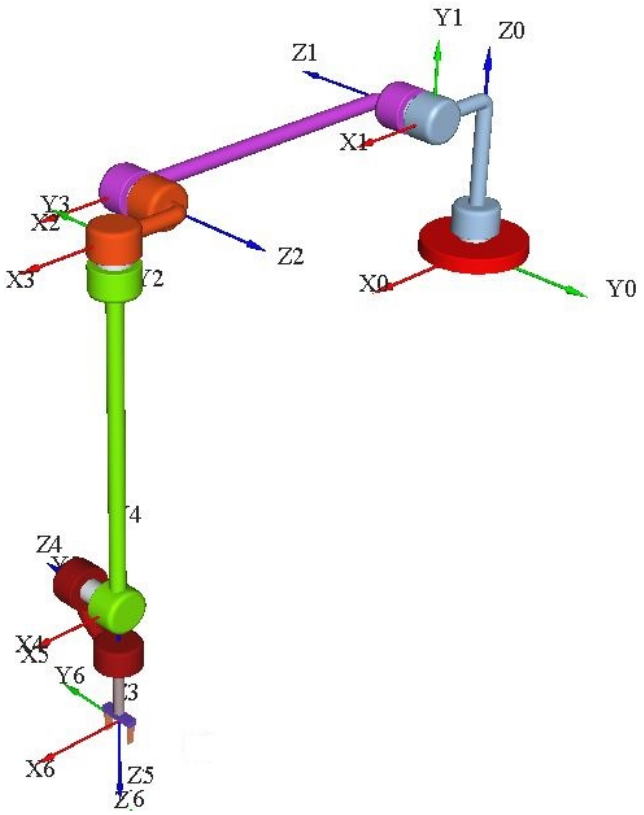


Figura 2.26: Sistema coordenado del robot Kawasaki BX100N.

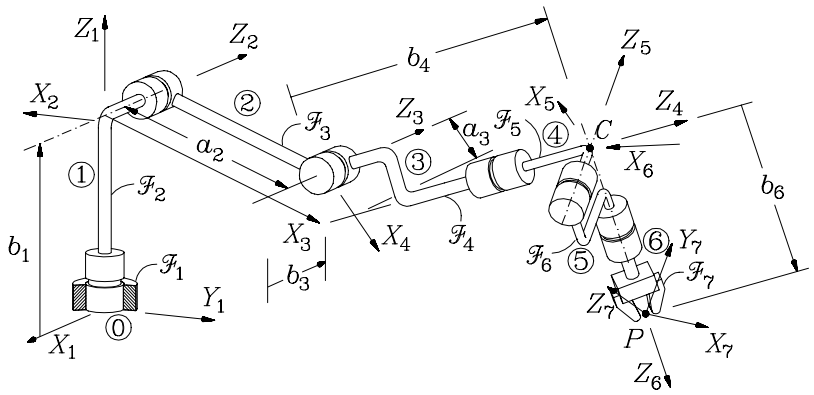


Figura 2.27: Marco de ejes coordenados de un robot articulado [20].

Tabla 2.2: Parámetros DH de Kawasaki BX100N

Eslabón	θ_i	a_i	b_i	α_i
1	θ_1^*	200	465	$\pi/2$
2	θ_2^*	880	187	π
3	θ_3^*	210	187	$-\pi/2$
4	θ_4^*	0	1100	$-\pi/2$
5	θ_5^*	0	0	$\pi/2$
6	θ_6^*	0	$225 + 160$	0

2.11 Cinemática directa.

Como se ha mencionado anteriormente, la tarea de la cinemática directa consiste en conocer la posición y orientación del efector final a partir de valores dados para las variables de articulación. Para llevar a cabo esta tarea es necesario encontrar n matrices de transformación homogénea, donde n es el número de eslabones del robot.

El manipulador que se estará utilizando en el presente trabajo cuenta con 6 GDL, donde las variables de cada eslabón se encuentran en la tabla DH (tabla 2.2), los valores de cada uno de los renglones de la tabla DH se utilizan para generar la matriz A_i, que es una matriz de transformación homogénea como en la ecuación (2.12). Cada uno de los parámetros DH:

θ_i, a_i, b_i y α_i en la ecuación (2.12) será reemplazado por su correspondiente valor tomando el renglón de la tabla DH (tabla 2.1).

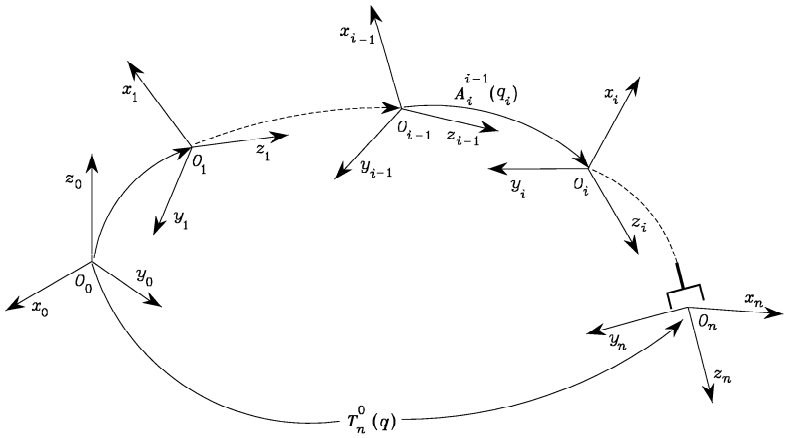


Figura 2.28: Transformación de coordenadas en cadena cinemática abierta [2].

Lo que finalmente se busca es encontrar T_n^0 como se muestra en la figura 2.28, que es la matriz de transformación homogénea (ecuación (2.5)) que describirá la posición y orientación del efector final con respecto a la base.

$$H = T_n^0 = A_1(q_1) \cdots A_n(q_n)$$

La matriz H está representada principalmente por la matriz de orientación y un vector de posición, que son los que representan a la pose del efector final, adicionalmente un vector de ceros y un escalar 1 en el último renglón de la matriz H (ecuación (2.6)), son necesarios para representar tal matriz como cuadrada.

Cada matriz de transformación A_i está formada de la siguiente manera:

$$A_i = \begin{bmatrix} R_i^{i-1} & o_i^{i-1} \\ 0 & 1 \end{bmatrix}$$

Una vez obtenida la tabla de parámetros DH, es posible obtener cada matriz de transformación homogénea de cada eslabón con la ecuación (2.12).

Para la matriz A_1 se tiene que:

$$cos(\alpha_1) \equiv 0$$
$$sin(\alpha_1) \equiv 1$$

$$A_1 \equiv \begin{bmatrix} c\theta_1 & 0 & s\theta_1 & 200\ c\theta_1 \\ s\theta_1 & 0 & -c\theta_1 & 200\ s\theta_1 \\ 0 & 1 & 0 & 465 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.13}$$

Para la matriz A_2 se tiene que:

$$cos(\alpha_2) \equiv -1$$
$$sin(\alpha_2) \equiv 0$$

$$A_2 \equiv \begin{bmatrix} c\theta_2 & s\theta_2 & 0 & 880\ c\theta_2 \\ s\theta_2 & -c\theta_2 & 0 & 880\ s\theta_2 \\ 0 & 0 & -1 & 187 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

Para la matriz A_3 se tiene que:

$$cos(\alpha_3) \equiv 0$$
$$sin(\alpha_3) \equiv -1$$

$$A_3 \equiv \begin{bmatrix} c\theta_3 & 0 & -s\theta_3 & 210\ c\theta_3 \\ s\theta_3 & 0 & c\theta_3 & 210\ s\theta_3 \\ 0 & -1 & 0 & 187 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.15}$$

Para la matriz A_4 se tiene que:

$$cos(\alpha_4) \equiv 0$$
$$sin(\alpha_4) \equiv -1$$

$$A_4 \equiv \begin{bmatrix} c\theta_4 & 0 & -s\theta_4 & 0 \\ s\theta_4 & 0 & c\theta_4 & 0 \\ 0 & -1 & 0 & 1100 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.16}$$

Para la matriz A_5 se tiene que:

$$cos(\alpha_5) \equiv 0$$
$$sin(\alpha_5) \equiv 1$$

$$A_5 \equiv \begin{bmatrix} c\theta_5 & 0 & s\theta_5 & 0 \\ s\theta_5 & 0 & -c\theta_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.17}$$

Para la matriz A_6 se tiene que:

$$cos(\alpha_6) \equiv 1$$
$$sin(\alpha_6) \equiv 0$$

$$A_6 \equiv \begin{bmatrix} c\theta_6 & -s\theta_6 & 0 & 0 \\ s\theta_6 & c\theta_6 & 0 & 0 \\ 0 & 0 & 1 & 385 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.18}$$

Teniendo las matrices de transformación homogénea anteriores, se puede calcular la cinemática directa del manipulador de la siguiente forma:

$$T_1^0 = A_1 \tag{2.19}$$
$$T_2^0 = T_1^0 T_2^1 = A_1 A_2 \tag{2.20}$$
$$T_3^0 = T_2^0 T_3^2 = A_1 A_2 A_3 = T_2^0 A_3 \tag{2.21}$$
$$T_4^0 = T_3^0 T_4^3 = A_1 A_2 A_3 A_4 = T_3^0 A_4 \tag{2.22}$$
$$T_5^0 = T_4^0 T_5^4 = A_1 A_2 A_3 A_4 A_5 = T_4^0 A_5 \tag{2.23}$$
$$T_6^0 = T_5^0 T_6^5 = A_1 A_2 A_3 A_4 A_5 A_6 = T_5^0 A_6 \tag{2.24}$$

Para mayor simplicidad en los cálculos, con algún software matemático es posible calcular rápidamente la matriz de transformación homogénea final, haciendo las operaciones matriciales.

Y con los valores de $\theta_1 = \theta_3 = \theta_4 = \theta_6 = 0°, \theta_2 = 90°$ *y* $\theta_5 = -90°$ en la estructura cinemática del robot como se muestra en la figura 2.29, se

obtiene la matriz de transformación homogénea siguiente:

$$H = T_6^0 \equiv \begin{bmatrix} 1.0 & 0.0 & 0.0 & 1300.0 \\ 0.0 & -1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -1.0 & 1170.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{2.25}$$

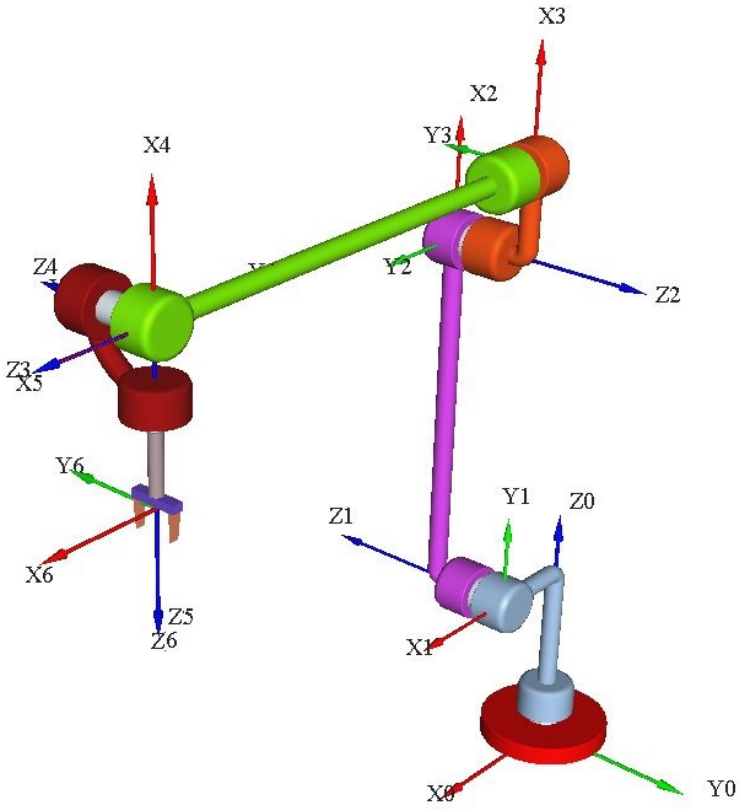


Figura 2.29: Esqueleto del manipulador con los ángulos $\theta_1 = \theta_3 = \theta_4 = \theta_6 = 0°$; $\theta_2 = 90°$ *y* $\theta_5 = -90°$.

2.12 Cinemática inversa.

Los manipuladores industriales se utilizan cada vez más con una arquitectura especial que permite el desacoplamiento de la posición y orientación por separado. En este caso se analizará un manipulador articulado de 6 grados de libertad con una muñeca esférica, la cual permite analizar los primeros tres

ejes de articulación θ_1, θ_2 y θ_3, que determinan la posición del manipulador y los últimas tres articulaciones θ_4, θ_5 y θ_6, que constituyen a la muñeca del manipulador, y se dice que es esférica ya que el punto de intersección de estos tres ejes se encuentra en un punto C (como se muestra en la figura 2.30) y todos los movimientos de la muñeca esférica se encuentran centradas en C.

En términos de los parámetros DH de un manipulador articulado desacoplado, las variables $a_4 = a_5 = b_5 = 0$, por lo tanto, los orígenes de los ejes 5 y 6 son coincidentes. Todos los demás parámetros DH pueden tomar valores arbitrarios.

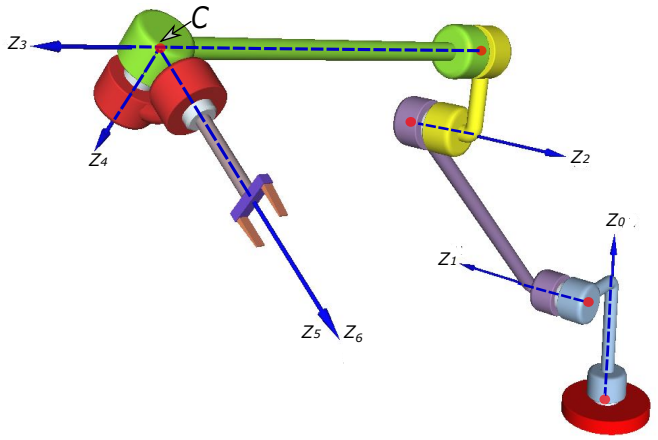


Figura 2.30: Configuración geométrica de un robot articulado con muñeca esférica.

2.12.1 Problema de posicionamiento.

Como se mencionó anteriormente, en un manipulador desacoplado, las tres primeras articulaciones determinan la posición del centro de la muñeca, mientras las tres últimas articulaciones de ésta, definen la orientación del efector final.

Sea C el punto donde los ejes 4, 5 y 6 se intersectan, que representa el centro de la muñeca esférica y entonces la distancia del punto C al origen o base del manipulador sea el vector posición del manipulador. Claramente la posición en C es independiente de los ángulos de articulación θ_4, θ_5 y θ_6, por

lo tanto, solamente las tres primeras articulaciones deben ser consideradas con la geometría del manipulador que se muestra en la figura 2.31.

Teniendo como entrada la matriz de transformación homogénea como dato inicial.

$$H \equiv T_6^0 = A_1 \cdots A_6 = \begin{bmatrix} R & \mathbf{d} \\ \mathbf{0} & 1 \end{bmatrix} \tag{2.26}$$

$$H \equiv \begin{bmatrix} r_{11} & r_{12} & r_{13} & d_x \\ r_{21} & r_{22} & r_{23} & d_y \\ r_{31} & r_{32} & r_{33} & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.27}$$

De la matriz de transformación homogénea (2.26), sea $\mathbf{d}$ el vector de posición que tiene el punto P de referencia del efector final del robot y la matriz R la que determina la orientación del efector final del robot. Sea también $\mathbf{o}_C^0$ el vector posición que va desde el origen hasta el punto C del manipulador.

Se puede conocer la posición del punto C en el manipulador, con las siguientes relaciones [15].

$$\mathbf{d} = \mathbf{o}_C^0 + b_6 R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.28}$$

$$\mathbf{o}_C^0 = \mathbf{d} - b_6 R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.29}$$

Donde los elementos de los vectores de $\mathbf{d}$ y $\mathbf{o}_C^0$ son:

$$\mathbf{d} \equiv \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix}, \quad \mathbf{o}_C^0 \equiv \begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix} \tag{2.30}$$

Por lo tanto, la ecuación (2.29) se puede representar de la siguiente manera:

$$\begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix} = \begin{bmatrix} d_x - b_6 r_{13} \\ d_y - b_6 r_{23} \\ d_z - b_6 r_{33} \end{bmatrix} \tag{2.31}$$

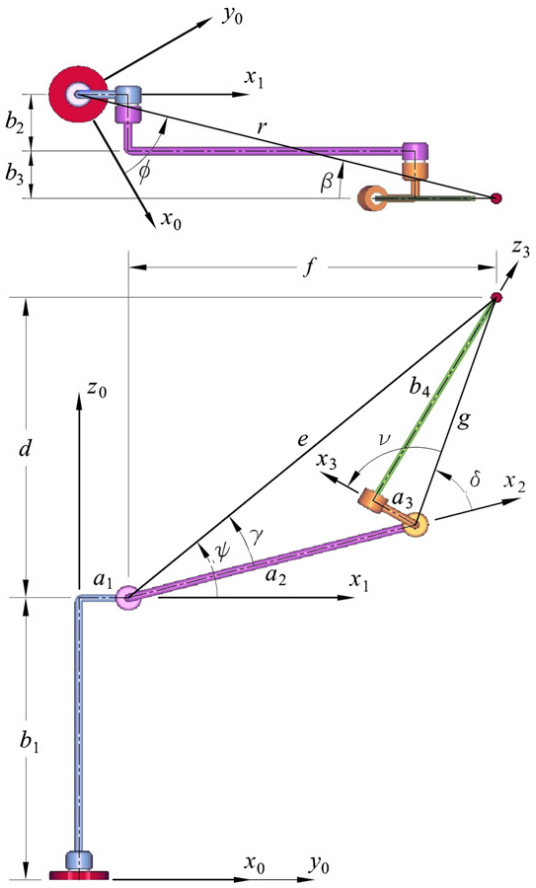


Figura 2.31: Geometría de soporte en la solución del problema de la cinemática inversa [24].

Ahora que se conoce la posición del punto C de la ecuación (2.31) en el manipulador, es posible calcular las primeras tres variables de articulación con la geometría del robot como se muestra en la figura 2.31. Para ilustrar un ejemplo de las configuraciones posibles en las que el manipulador puede posicionarse para obtener la misma posición en el punto C, se ilustra la figura 2.32 con las primeras configuraciones posibles en un robot PUMA.

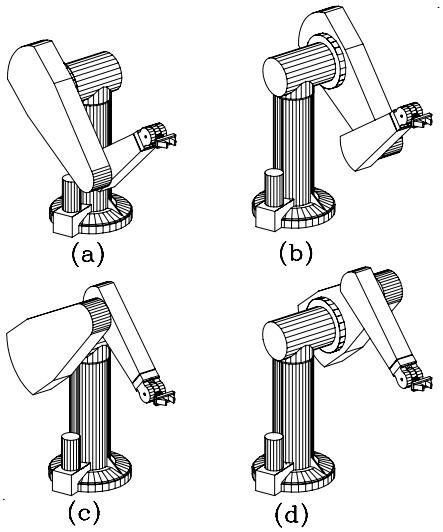


Figura 2.32: Cuatro soluciones básicas de la cinemática inversa del robot PUMA. (a) Brazo derecho codo abajo, (b) Brazo izquierdo codo abajo, (c) Brazo derecho codo arriba y (d) Brazo izquierdo codo arriba [20].

Tomando en cuenta la formulación presentada en [24], se tiene,

$$\lambda_i \equiv \cos(\alpha_i)$$
$$\mu_i \equiv \sin(\alpha_i)$$

$$h = (b_2 + (b_3 + b_4\lambda_3)\lambda_2)\mu_1 \tag{2.32}$$

$$\phi = \arctan\frac{p_y}{p_x} \tag{2.33}$$

$$r = \sqrt{p_x^2 + p_y^2} \tag{2.34}$$

$$\beta = \arctan\frac{h}{\sqrt{r^2 - h^2}} \tag{2.35}$$

$$v = \arctan\frac{b_4\mu_3}{a_3} \tag{2.36}$$

$$g = \sqrt{a_3^2 + (b_4\mu_3)^2} \tag{2.37}$$

Si el brazo está posicionado hacia adelante, entonces:

$$e = \sqrt{(p_z - b_1)^2 + (\sqrt{r^2 - h^2} - a_1)^2} \tag{2.38}$$

$$\psi = \arctan \frac{\mu_1(p_z - b_1)}{\sqrt{r^2 - h^2} - a_1} \tag{2.39}$$

$$c_\delta = \frac{e^2 - a_2^2 - g^2}{2ga_2} \tag{2.40}$$

Si el codo está posicionado en la parte superior, entonces

$$\delta = \arctan \frac{\sqrt{1 - c_\delta^2}\mu_1}{c_\delta} \tag{2.41}$$

Si el codo está posicionado en la parte inferior, entonces

$$\delta = \arctan \frac{-\sqrt{1 - c_\delta^2}\mu_1}{c_\delta} \tag{2.42}$$

Se tiene que

$$\gamma = \arctan \frac{g \sin \delta}{a_2 + g \cos \delta} \tag{2.43}$$

$$\theta_1 = \phi + \beta \tag{2.44}$$

$$\theta_2 = \psi - \gamma \tag{2.45}$$

$$\theta_3 = v + \delta\lambda_2 \tag{2.46}$$

Si el brazo está posicionado hacia atrás, entonces:

$$e = \sqrt{(p_z - b_1)^2 + (\sqrt{r^2 - h^2} + a_1)^2} \tag{2.47}$$

$$\psi = \arctan \frac{\mu_1(p_z - b_1)}{\sqrt{r^2 - h^2} + a_1} \tag{2.48}$$

$$c_\delta = \frac{e^2 - a_2^2 - g^2}{2ga_2} \tag{2.49}$$

Si el codo está posicionado en la parte superior, entonces

$$\delta = \arctan \frac{\sqrt{1 - c_\delta^2}\mu_1}{c_\delta} \tag{2.50}$$

Si el codo está posicionado en la parte inferior, entonces

$$\delta = \arctan -\frac{\sqrt{1 - c_\delta^2}\mu_1}{c_\delta} \tag{2.51}$$

Se tiene que

$$\gamma = \arctan \frac{g \sin \delta}{a_2 + g \cos \delta} \tag{2.52}$$

$$\theta_1 = \phi + (\pi - \beta) \tag{2.53}$$

$$\theta_2 = \pi - (\psi - \gamma) \tag{2.54}$$

$$\theta_3 = v - \delta\lambda_2 \tag{2.55}$$

2.12.2 Problema de orientación.

De manera simultánea, es posible calcular las siguientes tres variables de articulación θ_4, θ_5 y θ_6 en el manipulador. Para esto se requiere conocer la matriz de orientación en las tres últimas variables de articulación R_6^3. Para la configuración de muñeca esférica que se está usando, es posible resolver el problema relacionando los ángulos de Euler, que usan la misma configuración de rotaciones que una muñeca esférica.De manera que se puede calcular R_6^3 de la siguiente forma, [15]:

$$R = R_3^0 R_6^3 \tag{2.56}$$

$$R_6^3 = (R_3^0)^{-1} R = (R_3^0)^T R \tag{2.57}$$

Y se calcula la misma matriz de rotación con los ángulos de Euler $R_{ZYZ} = R_{z,\phi}R_{y,\theta}R_{z,\psi}$, como se muestra a continuación. Para simplificar un poco la notación, se utilizará el siguiente ejemplo: $s_4c_5c_6$ es equivalente a $\sin\theta_4\ \cos\theta_5\ \cos\theta_6$. Y si $\phi = \theta_4$, $\theta = \theta_5$ y $\psi = \theta_6$ entonces,

$$R_6^3 = \begin{bmatrix} c_4c_5c_6 - s_4s_6 & -c_4c_5s_6 - s_4c_6 & c_4s_5 \\ s_4c_5c_6 + c_4s_6 & -s_4c_5s_6 + c_4c_6 & s_4s_5 \\ -s_5c_6 & s_5s_6 & c_5 \end{bmatrix} \tag{2.58}$$

La solución a este problema se divide en dos casos como se muestra en la figura 2.33. En primer lugar tanto r_{13} y r_{23} de la ecuación (2.27) son cero. Luego de la ecuación (2.58) se deduce que $s_5 \neq 0$, y por lo tanto r_{31} y r_{32} de la ecuación (2.27) no son cero. Si este par r_{13} y r_{23} son cero, entonces $r_{33} \neq \pm 1$, y se tiene $c_5 = r_{33}$, $\quad s_5 = \pm\sqrt{1 - r_{33}^2}$

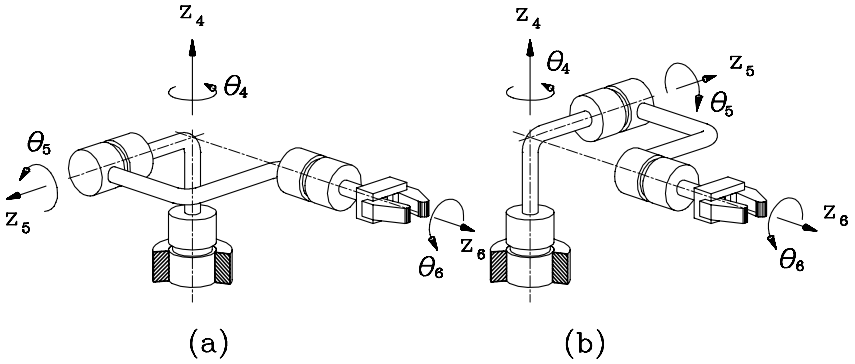


Figura 2.33: Dos configuraciones posibles en una muñeca esférica [20].

De forma que

$$\theta_5 = \arctan\left(r_{33}, \sqrt{1 - r_{33}^2}\right) \tag{2.59}$$

o

$$\theta_5 = \arctan\left(r_{33}, -\sqrt{1 - r_{33}^2}\right) \tag{2.60}$$

Si se toma el valor de θ_5 de la ecuación (2.59), entonces $s_5 > 0$, y

$$\theta_4 = \arctan\left(r_{13}, r_{23}\right) \tag{2.61}$$

$$\theta_6 = \arctan\left(-r_{31}, r_{32}\right) \tag{2.62}$$

Si se toma el valor de θ_5 de la ecuación (2.60), entonces $s_5 < 0$, y

$$\theta_4 = \arctan\left(-r_{13}, -r_{23}\right) \tag{2.63}$$

$$\theta_6 = \arctan\left(r_{31}, -r_{32}\right) \tag{2.64}$$

En este capítulo se mostraron algunos conceptos básicos de los robots seriales, específicamente de los manipuladores articulados, el análisis de la cinemática directa e inversa para la generación de su movimiento. El siguiente capítulo comenzará con la planificación de trayectorias lineales y curvilíneas para aplicaciones en la robótica.

Capítulo 3

Planificación de trayectoria

Contenido

En el presente capítulo se describe la teoría y planificación de diversas trayectorias en un espacio tridimensional, que el efector final de un manipulador pueda trazar. Entre estas trayectorias se encuentran las lineales y las curvilíneas.

3.1 Introducción.

Los movimientos que experimenta un sistema mecánico robótico deben de ser, en regla, lo más suaves posibles; esto es, se deben evitar cambios abruptos en posición, velocidad, y aceleración. Por otra parte, los cambios abruptos surgen cuando el robot colisiona con algún objeto, una situación que debe ser evitada en la tarea programada para el robot. Aunque se planeen movimientos suaves con técnicas simples, como se describirá más adelante, no hay garantía que no ocurran movimientos abruptos. De hecho, si el ambiente de trabajo está lleno de objetos, ya sea que estén estacionarios o en movimiento, las colisiones podrían surgir.

Sin embargo, existen aplicaciones en los manipuladores que no solo son ambientes bien estructurados o planeados en la producción automatizada, sino que hay ambientes mal planeados donde existen demasiados obstáculos, e inclusive llega a intervenir actividad humana. Y si en el espacio de trabajo del manipulador hay intervención humana, se debe ver la forma en la que haya menos situaciones de choque o de algún accidente, si es que no hay una buena planeación de su trayectoria. Es por eso que la planeación de trayectorias en un manipulador llega a ser un gran reto.

Por lo tanto, en la planificación de trayectorias y movimientos del robot en ambientes no estructurados se requieren técnicas más avanzadas que están fuera del alcance de este proyecto, que involucran áreas tales como el reconocimiento de imágenes e inteligencia artificial. Por esta razón, en el presente proyecto solamente se llega a la planificación de los movimientos del robot en entornos estructurados.

Dos tipos de planificación de trayectorias se analizarán,

- Operaciones con trayectorias lineales, tales como pick-and-place y

- Operaciones con trayectorias continuas, tales como funciones en el espacio, curvas, etc.

3.2 Operaciones con trayectorias lineales.

3.2.1 Pick and place.

En operaciones de pick and place, el manipulador articulado tiene la tarea de tomar una pieza de trabajo desde una pose inicial, determinada por la posición de uno de sus puntos y su orientación con respecto a un sistema de coordenadas definido por el efector final del robot, hasta una pose final, determinada de la misma manera.

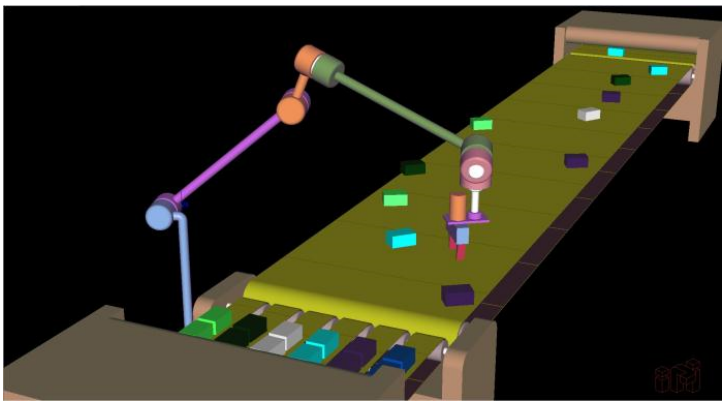


Figura 3.1: ADRS: Operación de pick & place [22].

Las operaciones de pick and place se ejecutan en procesos de fabricación y producción industrial tales como en la carga y descarga de las cintas transportadoras (conveyors), como se muestra en la Fig. 3.1, y las operaciones de ensamble de manera simple, como poner los rodamientos de rodillos sobre un eje. El común denominador de estas tareas es la manipulación de materiales, que por lo general requiere la presencia de las máquinas convencionales cuyo movimiento es muy simple y por lo general se caracteriza por velocidad uniforme. En algunos casos, como en las operaciones de ensamblaje, un conjunto de piezas de trabajo, por ejemplo, en un magazine, deben ser recolectadas en un orden prescrito a un recipiente, que constituye una operación conocida como paletización.

Cabe señalar que tanto la pose inicial como la pose final en un pick and place se prescriben en el espacio cartesiano, los movimientos del robot se implementan en el espacio de articulación. Por lo tanto, la planificación del pick and place se lleva a cabo en este último espacio, lo que provoca la necesidad del mapeo del movimiento previsto en el espacio cartesiano, a fin de garantizar que el robot no choque con algún objeto en su entorno.

Sin embargo, para movimientos generales en 3D y geometrías arbitrarias, los requerimientos computacionales hacen que el procedimiento sea muy impráctico. Un enfoque más práctico consistiría en dos etapas, [20]; (i) la planificación de una trayectoria preliminar en el espacio de la articulación, sin tener en cuenta los obstáculos, y (ii) verificar visualmente si las colisiones se producen con la ayuda de un sistema de procesamiento de gráficos de la animación del movimiento del robot en la presencia de obstáculos. La disponibilidad de hardware de gráficos de gran capacidad permite la animación rápida de los movimientos del robot dentro de un entorno en tiempo real. La figura 3.2 muestra una captura de la animación de una aplicación de recolección y de reconocimiento de bloques de color, donde sus trayectorias son lineales, definidas de un punto a otro, desarrolladas en la plataforma `ADEFID`, la cual está haciendo una aplicación tipo pick-and-place.

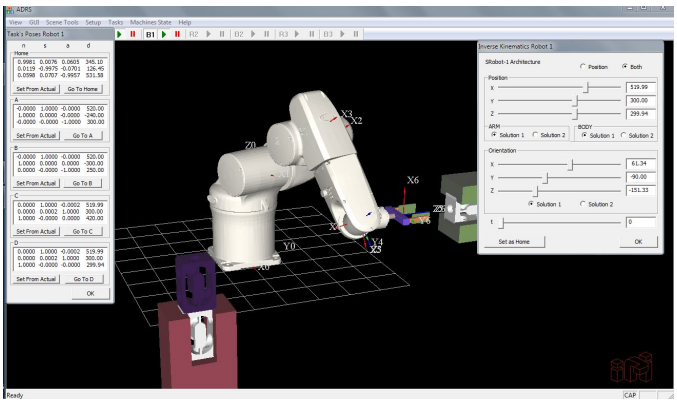


Figura 3.2: Operación pick and place en un entorno gráfico.

Con el fin de proceder a la síntesis de la trayectoria de articulación, se debe mapear la pose inicial y final de donde se encuentra la pieza de trabajo y en la pose de su contenedor, que se supone está firmemente ligado con el efector final del manipulador, dentro de las configuraciones descritas

del manipulador en el espacio de trabajo. Sea $\boldsymbol{\theta}_I$ la variable del vector de articulación inicial y $\boldsymbol{\theta}_F$ la variable del vector de articulación final del robot. Por otra parte, la pose inicial en el espacio cartesiano se define por el vector de posición $\mathbf{p}_I$ del punto de operación P del efector final y la matriz de rotación Q_I. Del mismo modo, la pose final en el espacio cartesiano se define por el vector de posición $\mathbf{p}_F$ de P y la matriz de rotación Q_F. Por otra parte, sea $\dot{\mathbf{p}}_I$ y $\ddot{\mathbf{p}}_I$ el vector velocidad y el vector aceleración de P, respectivamente, mientras que $\boldsymbol{\omega}_I$ y $\dot{\boldsymbol{\omega}}_I$ el vector de velocidad angular y el vector de aceleración angular, respectivamente, de la pieza de trabajo, todos éstos en la pose inicial. Estas variables para la pose final se denotan del mismo modo, donde el subíndice I se cambia por F. Asimismo, se supone que el tiempo se cuenta desde la pose inicial, es decir, en esta pose, $t = 0$. Si la orientación se lleva a cabo en el tiempo T, entonces en la pose final $t = T$. Por lo tanto, se tiene el conjunto de condiciones que definen un movimiento suave entre la pose inicial y la final, esto es,

$$\mathbf{p}(0) = \mathbf{p}_I \qquad \dot{\mathbf{p}}(0) = \mathbf{0} \qquad \ddot{\mathbf{p}}(0) = \mathbf{0} \tag{3.1}$$

$$Q(0) = Q_I \qquad \boldsymbol{\omega}(0) = \mathbf{0} \qquad \dot{\boldsymbol{\omega}}(0) = \mathbf{0} \tag{3.2}$$

$$\mathbf{p}(T) = \mathbf{p}_F \qquad \dot{\mathbf{p}}(T) = \mathbf{0} \qquad \ddot{\mathbf{p}}(T) = \mathbf{0} \tag{3.3}$$

$$Q(T) = Q_F \qquad \boldsymbol{\omega}(T) = \mathbf{0} \qquad \dot{\boldsymbol{\omega}}(T) = \mathbf{0} \tag{3.4}$$

En ausencia de singularidades, entonces, las condiciones de velocidad y aceleración implican velocidad y aceleración cero, esto es,

$$\boldsymbol{\theta}(0) = \boldsymbol{\theta}_I \qquad \dot{\boldsymbol{\theta}}(0) = \mathbf{0} \qquad \ddot{\boldsymbol{\theta}}(0) = \mathbf{0} \tag{3.5}$$

$$\boldsymbol{\theta}(T) = \boldsymbol{\theta}_F \qquad \dot{\boldsymbol{\theta}}(T) = \mathbf{0} \qquad \ddot{\boldsymbol{\theta}}(T) = \mathbf{0} \tag{3.6}$$

3.2.2 Interpolación polinomial 3-4-5.

Una simple inspección en las condiciones (3.5) y (3.6) revela que una interpolación lineal entre configuraciones iniciales y finales no funciona aquí, por lo que es necesario una interpolación de grado superior. Por otra parte estas condiciones implican a su vez, seis condiciones para cada trayectoria de articulación, lo que significa que si un polinomio se va a emplear para

representar el movimiento de todas las articulaciones, a continuación, este polinomio deberá ser al menos del quinto grado, por lo tanto, se comienza por el estudio de planificación de trayectorias con la ayuda de un polinomio de quinto grado.

Con el fin de representar a cada movimiento de articulación, se usa un polinomio de quinto grado $s(\tau)$, [20],

$$s(\tau) = a\tau^5 + b\tau^4 + c\tau^3 + d\tau^2 + e\tau + f \tag{3.7}$$

de manera que

$$0 \leq s \leq 1, \qquad 0 \leq \tau \leq 1 \tag{3.8}$$

y

$$\tau = \frac{t}{T} \tag{3.9}$$

Por lo tanto, se considera un polinomio normal, que nos permitirá representar a cada variable de articulación θ_j a lo largo de su rango de movimiento, de modo que, [20];

$$\theta_j(t) = \theta_j^I + (\theta_j^F - \theta_j^I)s(\tau) \tag{3.10}$$

Donde θ_j^I y θ_j^F son el valor inicial y final de cada variable de articulación representada con el subíndice j.

En forma vectorial, la equación (3.10) se re-escribe como;

$$\boldsymbol{\theta}(t) = \boldsymbol{\theta}_I + (\boldsymbol{\theta}_F - \boldsymbol{\theta}_I)s(\tau) \tag{3.11}$$

Y por lo tanto,

$$\dot{\boldsymbol{\theta}}(t) = (\boldsymbol{\theta}_F - \boldsymbol{\theta}_I)s'(\tau)\dot{\tau}(t) = (\boldsymbol{\theta}_F - \boldsymbol{\theta}_I)\frac{1}{T}s'(\tau) \tag{3.12}$$

Del mismo modo,

$$\ddot{\boldsymbol{\theta}}(t) = \frac{1}{T^2}(\boldsymbol{\theta}_F - \boldsymbol{\theta}_I)s''(\tau) \tag{3.13}$$

Y

$$\dddot{\boldsymbol{\theta}}(t) = \frac{1}{T^3}(\boldsymbol{\theta}_F - \boldsymbol{\theta}_I)s'''(\tau) \tag{3.14}$$

Lo que se necesita ahora son los valores de los coeficientes de $s(\tau)$ que aparecen en la ecuación (3.7). Estos se encuentran fácilmente por las condiciones (3.5) y (3.6), en conjunto con las ecuaciones (3.10) - (3.14). De esta manera se obtienen condiciones finales para $s(\tau)$, es decir,

$$s(0) = 0,\ s'(0) = 0,\ s''(0) = 0,\ s(1) = 1,\ s'(1) = 0,\ s''(1) = 0 \tag{3.15}$$

Las derivadas de $s(\tau)$ que se muestran abajo se obtienen fácilmente a partir de la ecuación (3.7),

$$s'(\tau) = 5a\tau^4 + 4b\tau^3 + 3c\tau^2 + 2d\tau + e \tag{3.16}$$

y

$$s''(\tau) = 20a\tau^3 + 12b\tau^2 + 6c\tau + 2d\tau \tag{3.17}$$

Así, los primeras tres condiciones de la ecuación (3.15) conducen a

$$f = e = d = 0 \tag{3.18}$$

mientras que las últimas tres condiciones producen tres ecuaciones lineales en a, b, y c, es decir

$$a + b + c = 1 \tag{3.19}$$

$$5a + 4b + 3c = 0 \tag{3.20}$$

$$20a + 12b + 6c = 0 \tag{3.21}$$

Al resolver las tres ecuaciones anteriores para las tres incógnitas antes mencionadas, se obtiene

$$a = 6, \qquad b = -15, \qquad c = 10 \tag{3.22}$$

Por consiguiente, el polinomio normal buscado es

$$s(\tau) = 6\tau^5 - 15\tau^4 + 10\tau^3 \tag{3.23}$$

El cual es llamado polinomio 3-4-5. Este polinomio y sus primeras tres derivadas, normalizadas en el intervalo de (-1,1), se muestran en la figura 3.3.

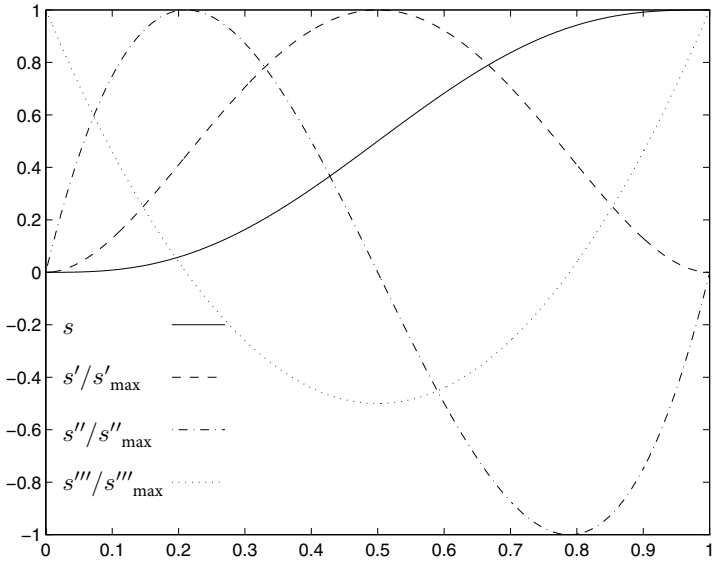


Figura 3.3: Interpolación del polinomio 3-4-5 y sus derivadas [20]

Por lo tanto, es posible determinar la evolución de cada variable de articulación si se conocen tanto sus valores finales y el tiempo T requerido para completar el movimiento. Se debe tener en cuenta, que este tiempo no puede ser un valor arbitrariamente pequeño, ya que se tienen que respetar las especificaciones de los motores en velocidad, aceleración y torque máximos. Aunque solo se especificarán la máxima velocidad de articulación y la máxima aceleración de articulación. De la función de $\theta_j(t)$ de la ecuación (3.10), con el fin de hallar valores de τ en la que la primera y segunda derivada de $s(\tau)$ alcanzan valores máximos, es necesario igualar a cero su segunda y tercer derivada. Estas derivadas se muestran a continuación:

$$s'(\tau) = 30\tau^4 - 60\tau^3 + 30\tau^2 \tag{3.24}$$

$$s''(\tau) = 120\tau^3 - 180\tau^2 + 60\tau \tag{3.25}$$

$$s'''(\tau) = 360\tau^2 - 360\tau + 60 \tag{3.26}$$

De la que es evidente que la segunda derivada se anula en los dos extremos del intervalo $0 \leq \tau \leq 1$. Además, la misma derivada se anula en el punto medio del mismo intervalo, es decir, en τ=1/2. Por lo tanto, el valor máximo de $s'(\tau)$, s'_{max}, se encuentra fácilmente como

$$s'_{max} = s'\left(\frac{1}{2}\right) = \frac{15}{8} \tag{3.27}$$

Y el valor máximo de la velocidad en la j-ésima articulación toma el valor

$$(\dot{\theta}_j) = \frac{15(\theta_j^F - \theta_j^I)}{8T} \tag{3.28}$$

De la misma manera, los valores máximo y mínimo de τ pueden conocerse con los extremos relativos de su segunda derivada. La tercera derivada se anula en dos puntos intermedios τ_1 y τ_2 del intervalo $0 \leq \tau \leq 1$, es decir, en

$$\tau_{1,2} = \frac{1}{2} \pm \frac{\sqrt{3}}{6} \tag{3.29}$$

El valor máximo se determina como

$$s''_{max} = s''\left(\frac{1}{2} - \frac{\sqrt{3}}{6}\right) = \frac{10\sqrt{3}}{3} \tag{3.30}$$

mientras que el valor mínimo está dado por

$$s''_{min} = s''\left(\frac{1}{2} + \frac{\sqrt{3}}{6}\right) = -\frac{10\sqrt{3}}{3} \tag{3.31}$$

El valor máximo de la aceleración en el conjunto de articulación se muestra a continuación:

$$(\ddot{\theta}_j) = \frac{10\sqrt{3}}{3}\frac{(\theta_j^F - \theta_j^I)}{T^2} \tag{3.32}$$

Del mismo modo

$$s'''_{max} = s'''(0) = s'''(1) = 60$$

Y por lo tanto

$$(\dddot{\theta}_j) = 60\frac{(\theta_j^F - \theta_j^I)}{T^3} \tag{3.33}$$

Así, las ecuaciones (3.28) y (3.32) permiten determinar T de cada articulación de modo que los cambios en las articulaciones en velocidad y aceleración se encuentran dentro de los límites máximos y mínimos de τ en s y s''.

3.2.3 Interpolación cúbica usando splines.

La trayectoria polinómica discutida anteriormente no permite la especificación de poses cartesianas intermedias del efector final. Lo único que garantiza es que se cumplan las trayectorias cartesianas prescritas en los instantes inicial y final. Una forma de comprobar la viabilidad de las trayectorias cartesianas, así como sintetizarlas, consiste en utilizar un sistema de gráficos, con capacidades de animación, para producir una representación animada del movimiento del robot, pretendiendo de esta manera verificar las posibles colisiones. Si esto último ocurre, se debe de tener una ruta alternativa posible con la cinemática inversa, calculada en la pose final o modificar completamente la trayectoria para evitar colisiones.

Esto se lleva a cabo con las denominadas **posturas**, también llamadas vía poses, es decir, poses del efector final definidas en el espacio cartesiano que están entre la pose inicial y la pose final, y deben estar determinadas a fin de evitar colisiones. Por ejemplo, si al acercarse a la pose final de una aplicación de pick and place, el manipulador detecta que va a interferir con la superficie donde se va a colocar la pieza de trabajo, una postura es seleccionada cerca del punto final de manera que en esta posición, la pieza de trabajo esté lo suficientemente lejos de la superficie. Para la cinemática inversa, los valores de las variables de articulación se pueden determinar con las posturas mencionadas anteriormente. Estos valores pueden ahora ser considerados como puntos en la trayectoria del espacio de articulación y son llamados **puntos vía**. Obviamente, bajo el trazado de cada variable de articulación en función del tiempo, los puntos vía aparecen como puntos trazados en el espacio, representando una trayectoria en el espacio.

Ahora, cuando el número de posturas se incrementa, se vuelve cada vez más impráctico que la interpolación sea lineal. De hecho forzar a una trayectoria a que pase a través de una serie de puntos vía y conocer las condiciones de cada punto es equivalente a la interpolación. Y se ha visto que al aumentar el número de condiciones el grado del polinomio de interpolación aumenta y será necesario calcular los coeficientes del polinomio, esto requiere resolver un sistema de ecuaciones lineales. Cuando se aumentan las condiciones o se incrementa el número de posturas en una trayectoria su cálculo se torna impráctico.

Una alternativa de un polinomio de grado alto, son los funciones Spline. Las **funciones spline**, en forma breve, son polinomios a trozos con propiedades de continuidad impuestas en los **puntos de apoyo**. Estos últimos son aquellos puntos en los que se unen dos polinomios vecinos.

La característica más importante de los Splines es que se definen como un conjunto de polinomios de grado menor unidos por un número de puntos de apoyo. Además, las matrices que se generan de un problema de interpolación asociada con una función spline son tales que su número de condiciones sólo es ligeramente dependiente de la cantidad de puntos de apoyo, y por lo tanto, los splines ofrecen la posibilidad de interpolación en un número virtualmente ilimitado de puntos sin producir serios problemas en los cálculos matemáticos. Usando la interpolación con spline, se puede seguir con la planificación de trayectorias con el manipulador articulado.

Una función cúbica spline $s(x)$ que conecta N puntos $P_k(x_k, y_k)$, para $k = 1, 2, \ldots, N$, es una función a trozos definida por $N-1$ polinomios cúbicos unidos por los puntos de P_k, de manera que $s(x_k) = y_k$.

Sean entonces $P_k(x_k, y_k)$ y $P_{k+1}(x_{k+1}, y_{k+1})$ dos puntos de apoyo consecutivos. El k-ésimo polinomio cúbico $s_k(x)$ entre dos puntos se supone que está dado por

$$s_k(x) = A_k(x - x_k)^3 + B_k(x - x_k)^2 + C_k(x - x_k) + D_k \tag{3.34}$$

para $x_k \leq x \leq x_{k+1}$. Así, para la función spline $s(x)$, $4(N-1)$ coeficientes de las variables A_k, B_k, C_k, D_k, para $k = 1, \ldots, N-1$, se determinarán más adelante. Estos coeficientes se calcuarán en términos de los valores de las

funciones dadas $\{s_k\}_1^N$ y de las segundas derivadas de la curva de la sección en los puntos de apoyo, $\{s_k''\}_1^N$, que se explican más abajo: Primero, se necesitan la primera y segunda derivada de s_k, esto es,

$$s_k'(x) = 3A_k(x - x_k)^2 + 2B_k(x - x_k) + C_k \tag{3.35}$$

$$s_k''(x) = 6A_k(x - x_k) + 2B_k \tag{3.36}$$

Donde las relaciones de los coeficientes inmediatamente son, [20]:

$$B_k = \frac{1}{2}s_k'' \tag{3.37}$$

$$C_k = s_k' \tag{3.38}$$

$$D_k = s_k \tag{3.39}$$

Utilizando las abreviaturas siguientes,

$$s_k \equiv s(x_k), \quad s_k' \equiv s'(x_k), \quad s_k'' \equiv s''(x_k) \tag{3.40}$$

Por lo tanto, sea

$$\Delta x_k \equiv x_{k+1} - x_k \tag{3.41}$$

De las relaciones descritas anteriormente, se tienen las expresiones para los coeficientes B_k y D_k en términos de s_k'' y s_k, respectivamente, pero la expresión para C_k está en términos de s_k'. La cual se desea que tenga las mismas expresiones tanto para A_k y C_k, esto es, en términos de s_k y s_k''. Las relaciones buscadas se encuentran mediante la imposición de las condiciones de continuidad en la función spline y sus primera y segunda derivadas con respecto a x en los puntos de apoyo. Estas condiciones son, entonces, para $k = 1, 2, \ldots, N - 1$,

$$s_k(x_{k+1}) = s_{k+1} \tag{3.42}$$

$$s'_k(x_{k+1}) = s'_{k+1} \tag{3.43}$$

$$s''_k(x_{k+1}) = s''_{k+1} \tag{3.44}$$

Tras la sustitución de $s''_k(x_{k+1})$, dada por la ecuación (3.36), en la Ec. (3.44), se obtiene

$$6A_k\Delta x_k + 2B_k = 2B_{k+1}$$

Pero de la ecuación (3.37), ya se tiene una expresión para B_k, y por lo tanto, una para B_{k+1} también. Sustituyendo estas dos expresiones en la ecuación anterior, se obtiene una expresión para A_k, esto es, [20];

$$A_k = \frac{1}{6\Delta x_k}\left(s''_{k+1} - s''_k\right) \tag{3.45}$$

Por otra parte, si se sustituye s_{k+1}, según la ecuación (3.34), en la Ec. (3.42), resulta

$$A_k(\Delta x_k)^3 + B_k(\Delta x_k)^2 + C_k\Delta x_k + D_k = s_{k+1}$$

Pero ya se tienen valores para A_k, B_k y C_k de las ecuaciones (3.45) y (3.37), respectivamente. Al sustituir estos valores en la ecuación anterior, se obtiene la expresión deseada para C_k en términos de la función y los valores de la segunda derivada, es decir,

$$C_k = \frac{\Delta s_k}{\Delta x_k} - \frac{1}{6}\Delta x_k\left(s''_{k+1} + 2s''_k\right) \tag{3.46}$$

En resumen, ahora se tienen las expresiones para los cuatro coeficientes del k-ésimo polinomio en términos de valores de la función y de la segunda derivada en los puntos de apoyo,

$$A_k = \frac{1}{6\Delta x_k}\left(s''_{k+1} - s''_k\right) \tag{3.47}$$

$$B_k = \frac{1}{2}s''_k \tag{3.48}$$

$$C_k = \frac{\Delta s_k}{\Delta x_k} - \frac{1}{6}\Delta x_k\left(s''_{k+1} + 2s''_k\right) \tag{3.49}$$

$$D_k = s_k \tag{3.50}$$

teniendo en cuenta que

$$\Delta s_k \equiv s_{k+1} - s_k \tag{3.51}$$

Por otra parte, de la condición de continuidad en la primera derivada, Ec. (3.43), después de la sustitución de la ecuación (3.35), se obtiene;

$$3A_k(\Delta x_k)^2 + 2B_k\Delta x_k + C_k = C_{k+1}$$

O, si se regresa a los polinomios anteriores

$$3A_{k-1}(\Delta x_{k-1})^2 + 2B_{k-1}\Delta x_{k-1} + C_{k-1} = C_k$$

Ahora bien, si se sustituyen las expresiones de (3.47) a (3.49) en la ecuación que se acaba de indicar, se obtiene un sistema lineal de $N-2$ ecuaciones simultáneas para las N incógnitas $\{s''_k\}_1^N$:

$$\begin{aligned}&(\Delta x_k)s''_{k+1} + 2(\Delta x_{k-1} + \Delta x_k)s''_k + (\Delta x_{k-1})s''_{k-1}\\ &= 6\left(\frac{\Delta s_k}{\Delta x_k} - \frac{\Delta s_{k-1}}{\Delta x_{k-1}}\right), \quad para \quad k = 2, \ldots, N-1.\end{aligned} \tag{3.52}$$

Además, sea $\mathbf{s}$ el vector N-dimensional, cuyo componente de orden k es s_k, con el vector $\mathbf{s}''$ siendo definido de la misma manera, es decir,

$$\mathbf{s} = [s_1, \ldots, s_N]^T, \qquad \mathbf{s}'' = [s''_1, \ldots, s''_N]^T \tag{3.53}$$

La relación entre $\mathbf{s}$ y $\mathbf{s}''$, de la ecuación (3.52), entonces se puede escribir en forma vectorial como

$$\mathrm{A}\mathbf{s}'' = 6\mathrm{C}\mathbf{s} \tag{3.54}$$

Donde A y C son matrices de $(N-2)\times N$ definidas como

$$\mathrm{A} = \begin{bmatrix} \alpha_1 & 2\alpha_{1,2} & \alpha_2 & 0 & \cdots & 0 & 0 \\ 0 & \alpha_2 & 2\alpha_{2,3} & \alpha_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \alpha_{N'''} & 2\alpha_{N''',N''} & \alpha_{N''} & 0 \\ 0 & 0 & 0 & \cdots & \alpha_{N''} & 2\alpha_{N'',N'} & \alpha_{N'} \end{bmatrix} \tag{3.55}$$

$$\mathrm{C} = \begin{bmatrix} \beta_1 & -\beta_{1,2} & \beta_2 & 0 & \cdots & 0 & 0 \\ 0 & \beta_2 & -\beta_{2,3} & \beta_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \beta_{N'''} & -\beta_{N''',N''} & \beta_{N''} & 0 \\ 0 & 0 & 0 & \cdots & \beta_{N''} & -\beta_{N'',N'} & \beta_{N'} \end{bmatrix} \tag{3.56}$$

Mientras que para $i, j, k = 1, \ldots, N-1$

$$\alpha_k \equiv x_k, \qquad \alpha_{i,j} \equiv \alpha_i + \alpha_j \tag{3.57}$$

$$\beta_k \equiv 1/\alpha_k, \qquad \beta_{i,j} \equiv \beta_i + \beta_j \tag{3.58}$$

y

$$N' \equiv N-1, \qquad N'' \equiv N-2, \qquad N''' \equiv N-3 \tag{3.59}$$

Por lo tanto, se necesitan dos ecuaciones adicionales para hacer que la ecuación (3.54) sea un sistema determinado. Las ecuaciones adicionales se obtienen, a su vez, dependiendo de la clase de funciones que uno está tratando, lo que induce a diversos tipos de funciones spline. Por ejemplo, si s_1'' y s_N'' se definen como cero, entonces se obtiene *splines cúbicos naturales*. En este caso, el vector $\mathbf{s}''$ se convierte de dimensión $N-2$, y por lo tanto la matriz A se convierte, en consecuencia, de $(N-2) \times (N-2)$, es decir,

$$\mathrm{A} = \begin{bmatrix} 2\alpha_{1,2} & \alpha_2 & 0 & \cdots & 0 \\ \alpha_2 & 2\alpha_{2,3} & \alpha_3 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \alpha_{N'''} & 2\alpha_{N''',N''} & \alpha_{N''} \\ 0 & 0 & \cdots & \alpha_{N''} & 2\alpha_{N'',N'} \end{bmatrix} \tag{3.60}$$

Por otro lado, si se tiene interés en funciones periódicas, entonces se imponen las condiciones $s_1 = s_N$, $s_1' = s_N'$, $s_1'' = s_N''$, produciendo de esta manera splines cúbicas periódicas. La última de estas condiciones se utiliza para eliminar una incógnita en la ecuación (3.54), mientras que la segunda condición, también llamada de continuidad de la primera derivada, se utiliza para añadir una ecuación. Así,

$$s_1' = s_N' \tag{3.61}$$

que puede ser escrita, usando (3.43), como

$$s'_1 = s'_{N-1}(x_N) \tag{3.62}$$

Tras la sustitución de $s'_{N-1}(x_N)$, dada por la ecuación (3.35), en la ecuación anterior, se obtiene

$$s'_1 = 3A_{N-1}\Delta x_{N-1}^2 + 2B_{N-1}\Delta x_{N-1} + C_{N-1} \tag{3.63}$$

Ahora se usan las Ecs. (3.47)-(3.49) y se simplifica la expresión resultante que conduce a

$$s'_1 = 3A_{N-1}\Delta x_{N-1}^2 + 2B_{N-1}\Delta x_{N-1} + C_{N-1} \tag{3.64}$$

siendo ésta, la última ecuación que se requiere para resolver el sistema de ecuaciones dadas por las ecuaciones (3.54)−(3.56). De esta manera se tienen $(N-1)$ ecuaciones independientes para resolver $(N-1)$ incógnitas, esto es, s''_k, para $k = 1, \ldots, N-1, s''_N$ siendo igual a s''_1. Las expresiones para las matrices A y C, que corresponden a una spline cúbica periódica, se dan en las ecuaciones (3.96) y (3.97).

3.3 Operaciones con trayectorias continuas.

En operaciones de pick and place, la pose, el giro y la velocidad de giro en el efector final son especificados sólo en dos extremos de la trayectoria, sin embargo, el propósito de la planificación de trayectorias continuas es la integración de estas dos poses o más con movimientos suaves.

En esta sección se analizarán dos tipos de interpolaciones en trayectorias que producen movimientos suaves tanto en el espacio cartesiano como en las variables de articulación del robot.

3.3.1 Geometría de la curva.

El sector industrial ha demandado diversas aplicaciones en las que está involucrado el seguimiento de trayectorias curvilíneas, con mayor rapidez, precisión y calidad. Algunas aplicaciones requieren que estas tareas se efectúen

a lo largo de una curva con movimientos suaves, por ejemplo, en fuselajes de aviones, la trayectoria o camino es trazado como una función de tiempo prescrita. Esta función por otra parte es una tarea dependiente, ya que controla y/o especifica la geometría de tal trayectoria.

La mayoría de los métodos de planificación de trayectorias se enfocan en la obtención de la curva en un espacio cartesiano seguida por el punto de operación. Sin embargo, la orientación que debe tomar el sistema de referencia cuyo origen es el punto que se estará trasladando en la misma curva, se basa en una tríada de vectores conocida como Frenet-Serret (ver Fig. 3.4). En presencia de manipuladores articulados con muñeca esférica, el posicionamiento y la orientación son fácilmente separables por lo tanto, la planificación de las dos tareas se puede hacer por separado [2].

La orientación de una trayectoria se puede determinar por su geometría. Sea entonces $\mathbf{p}$ un vector de (3×1) y $\mathbf{f}(s)$ una función vectorial continua en un intervalo de $[s_I, s_F]$. Considerando la siguiente ecuación

$$\mathbf{p} = \mathbf{f}(s) \tag{3.65}$$

Haciendo referencia a la descripción geométrica de la curva. La ecuación (3.65) define la representación paramétrica de la curva Γ y el escalar s como el parámetro que estará variando. De manera, que cuando s vaya incrementándose, el punto p se irá moviendo en la curva en cierta dirección. Una curva cerrada está definida cuando $\mathbf{p}(s_F) = \mathbf{p}(s_I)$, y una curva abierta cuando $\mathbf{p}(s_F) \neq \mathbf{p}(s_I)$.

Sea entonces p_I el punto inicial en la curva abierta Γ y p_F el punto final de la misma curva como se muestra en la figura 3.4. Y el parámetro σ de la curva representa la longitud de arco de la trayectoria Γ.

A partir de la representación anterior se deduce que, cada valor de σ en la trayectoria, corresponde a un punto en la trayectoria Γ ya definida en forma paramétrica, como lo define la siguiente ecuación.

$$\mathbf{p} = \mathbf{f}(\sigma) \tag{3.66}$$

El rango de variación del parámetro σ está definido desde cero hasta el valor máximo o total de la longitud de arco de dicha trayectoria.

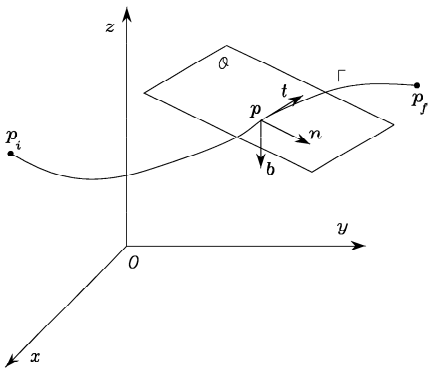


Figura 3.4: Tríada de vectores Frenet-Serret [2].

Sea p un punto cualquiera en una curva abierta Γ representada por la ecuación (3.66) que va ir variando con respecto al parámetro σ. El punto p tiene tres vectores unitarios los cuales definen su orientación en dicha curva, y estos vectores dependen exclusivamente de la geometría de la curva ya definida por la ecuación (3.66).

En estas condiciones, se puede asociar con cada punto de este intervalo una tríada ortonormal de vectores, es decir, un conjunto de vectores unitarios que son mutuamente ortogonales, que son, el vector tangente $\mathbf{t}$, el vector normal $\mathbf{n}$, y el vector binormal $\mathbf{b}$, de Γ. Por lo tanto, cuando este conjunto de vectores está dispuesto adecuadamente en una matriz de 3×3, se obtiene una matriz de rotación. Por tanto, esta matriz representa la orientación de Γ.

El primer vector unitario es el vector tangente $\mathbf{t}$, este vector está orientado a lo largo de la dirección inducida por la curva con la variación del parámetro σ. El segundo vector unitario conocido como vector normal $\mathbf{n}$, es orientado a lo largo de la línea de intersección $\mathbf{p}$ en un ángulo recto con $\mathbf{t}$ y se encuentra en el llamado **plano osculante** (ver Fig. 3.4), dicho plano es la posición límite del plano que contiene $\mathbf{t}$ y un punto $\mathbf{p}' \in \Gamma$ cuando $\mathbf{p}'$ tiende a $\mathbf{p}$ a lo largo de la trayectoria. Y finalmente el tercer vector unitario es el binormal $\mathbf{b}$, este vector es el producto cruz de los dos anteriores. Esta tríada de vectores Frenet-Serret conforman la orientación de $\mathbf{p}$ a lo largo de la curva Γ con respecto a la variación del parámetro σ.

Los vectores unitarios anteriores están representados por simples rela-

ciones con la curva representada por Γ como función de la geometría de tal curva. De manera que los vectores Frenet-Serret están representados de la siguiente manera, [2]:

$$\mathbf{t} = \frac{d\mathbf{p}}{d\sigma}, \qquad \mathbf{n} = \frac{1}{\left\| \frac{d^2\mathbf{p}}{d\sigma^2} \right\|} \frac{d^2\mathbf{p}}{d\sigma^2}, \qquad \mathbf{b} = \mathbf{t} \times \mathbf{n}. \tag{3.67}$$

3.3.2 Planificación de trayectorias con parametric splines.

A veces la trayectoria a seguir por la punta del efector final del robot se da sólo como un conjunto discreto de puntos $\{P_i\}_1^N$. Este es el caso, por ejemplo, si la ruta es la intersección de dos superficies deformadas, como en la soldadura por arco de dos placas del casco de un buque o la soldadura por puntos de dos láminas del fuselaje de un avión. En estos casos, las coordenadas de los puntos de muestra o bien se calculan numéricamente a través de la solución de ecuaciones no lineales o utilizando un sistema de visión. En cualquier caso, es evidente que sólo las coordenadas de los puntos está disponible, mientras que la planificación de la trayectoria demanda información sobre las derivadas del vector posición en los puntos de la trayectoria con respecto a la longitud del arco. Estas derivadas se pueden estimar a través de una interpolación adecuada de las coordenadas dadas.

Las funciones splines introducidas anteriormente son aplicables siempre que sean una *función*, no como una curva geométrica en el espacio tridimensional, la cual se interpola, y por lo tanto, estas splines, denominadas *no paramétricas*, ya no son aplicables. Lo que se necesita en este caso son splines paramétricas, como se describe a continuación.

Aunque las splines paramétricas, a su vez, pueden ser de diversos tipos, por simplicidad se analizarán splines paramétricas cúbicas.

Sea $P_i(x_i, y_i, z_i)$, para $i = 1, \ldots, N$, el conjunto de puntos de muestra en la ruta a ser trazada por la punta del efector final del robot, siendo $\{\mathbf{p}_i\}_1^N$ el conjunto de vectores de posición correspondiente. El objetivo en esta sección es producir una trayectoria suavizada que pase a través de $\{P_i\}_1^N$ y que contenga una tríada Frenet-Serret continua. Con este fin, se define una variable primordial en este conjunto de operaciones, el parámetro σ.

Primeramente se introducirá un par de definiciones: Sea que la derivada k-ésima del vector posición $\mathbf{p}$ de un punto arbitrario P de Γ con respecto al parámetro σ, evaluada en P_i se denota por $\mathbf{p}_i^{(k)}$, sus componentes se designan correspondientemente por $x_i^{(k)}$, $y_i^{(k)}$, y $z_i^{(k)}$. Luego, las coordenadas de P están expresadas del mismo modo en polinomios cúbicos de σ, esto es, [20];

$$x(\sigma) = A_{xi}(\sigma - \sigma_i)^3 + B_{xi}(\sigma - \sigma_i)^2 + C_{xi}(\sigma - \sigma_i) + D_{xi} \tag{3.68}$$

$$y(\sigma) = A_{yi}(\sigma - \sigma_i)^3 + B_{yi}(\sigma - \sigma_i)^2 + C_{yi}(\sigma - \sigma_i) + D_{yi} \tag{3.69}$$

$$z(\sigma) = A_{zi}(\sigma - \sigma_i)^3 + B_{zi}(\sigma - \sigma_i)^2 + C_{zi}(\sigma - \sigma_i) + D_{zi} \tag{3.70}$$

Para el parámetro real σ, de tal manera que $\sigma_i \leq \sigma \leq \sigma_{i+1}$, y $i = 1, \ldots, N-1$, con σ_i definido como,

$$\sigma_1 = 0, \quad \sigma_{i+1} \equiv \sigma_i + \Delta\sigma_i, \quad \Delta\sigma_i \equiv \sqrt{\Delta x_i^2 + \Delta y_i^2 + \Delta z_i^2} \tag{3.71}$$

$$\Delta x_i \equiv x_{i+1} - x_i, \qquad \Delta y_i \equiv y_{i+1} - y_i, \qquad \Delta z_i \equiv z_{i+1} - z_i \tag{3.72}$$

Y por lo tanto, $\Delta\sigma_i$ representa la longitud de la cuerda subtendida por el arco de la trayectoria entre P_i y P_{i+1}. Del mismo modo, σ denota una longitud de la trayectoria medida a lo largo de las uniones de cada polinomio en el espacio de los N puntos de $\{P_i\}_1^N$. Así, entre más cercanos los puntos antes mencionados, más cercana es la aproximación $\Delta\sigma_i$ a la longitud de arco entre esos dos puntos, y por lo tanto, son mejores las aproximaciones de las propiedades de la curva.

Los coeficientes spline anteriores $A_{xi}, A_{yi}, \ldots, D_{zi}$, para $i = 1, \ldots, N-1$, se determinan como se explica a continuación. Teniendo definidos vectores de dimensión N

$$\mathbf{x} \equiv [x_1, \ldots, x_N]^T, \qquad \mathbf{x}'' \equiv [x_1'', \ldots, x_N'']^T \tag{3.73}$$

$$\mathbf{y} \equiv [y_1, \ldots, y_N]^T, \qquad \mathbf{y}'' \equiv [y_1'', \ldots, y_N'']^T \tag{3.74}$$

$$\mathbf{z} \equiv [z_1, \ldots, z_N]^T, \qquad \mathbf{z}'' \equiv [z_1'', \ldots, z_N'']^T \tag{3.75}$$

La relación entre $\mathbf{x}, \mathbf{y}, \mathbf{z}$ y de sus contrapartes, $\mathbf{x}'', \mathbf{y}'', \mathbf{z}''$ son los mismas que las funciones splines no paramétricas que se encuentran en la ecuación (3.54), esto es,

$$\mathbf{A}\mathbf{x}'' = 6\mathbf{C}\mathbf{x} \tag{3.76}$$

$$\mathbf{A}\mathbf{y}'' = 6\mathbf{C}\mathbf{y} \tag{3.77}$$

$$\mathbf{A}\mathbf{z}'' = 6\mathbf{C}\mathbf{z} \tag{3.78}$$

las cuales son expresiones similares a la ecuación (3.54), excepto que las matrices A y C que aparecen en la Ec. (3.77) son ahora funciones propias de las coordenadas de los puntos de apoyo de la spline. De hecho, las matrices A y C de tamaño $(N-2) \times N$ se definen exactamente como las ecuaciones (3.55) y (3.56), que se repiten a continuación para una referencia rápida:

$$\mathrm{A} = \begin{bmatrix} \alpha_1 & 2\alpha_{1,2} & \alpha_2 & 0 & \cdots & 0 & 0 \\ 0 & \alpha_2 & 2\alpha_{2,3} & \alpha_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \alpha_{N'''} & 2\alpha_{N''',N''} & \alpha_{N''} & 0 \\ 0 & 0 & 0 & \cdots & \alpha_{N''} & 2\alpha_{N'',N'} & \alpha_{N'} \end{bmatrix} \tag{3.79}$$

$$\mathrm{C} = \begin{bmatrix} \beta_1 & -\beta_{1,2} & \beta_2 & 0 & \cdots & 0 & 0 \\ 0 & \beta_2 & -\beta_{2,3} & \beta_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \beta_{N'''} & -\beta_{N''',N''} & \beta_{N''} & 0 \\ 0 & 0 & 0 & \cdots & \beta_{N''} & -\beta_{N'',N'} & \beta_{N'} \end{bmatrix} \tag{3.80}$$

donde α_k *y* β_k ahora se definen de manera correspondiente, es decir, para $i, j, k = 1, \ldots, N-1$,

$$\alpha_k = \Delta\sigma_k, \quad \alpha_{i,j} = \alpha_i + \alpha_j, \quad \beta_k = 1/\alpha_k, \quad \beta_{i,j} = \beta_i + \beta_j, \tag{3.81}$$

y

$$N' \equiv N-1, \qquad N'' \equiv N-2, \qquad N''' \equiv N-3 \tag{3.82}$$

Hay que tener en cuenta que la curva spline de selección $\mathbf{p}(\sigma)$ está totalmente determinada una vez que se conozcan sus coeficientes. Éstos se calculan exactamente como sus contrapartes de splines no paramétricas, es decir, como en las Ecs. (3.47)−(3.51). A diferencia de los coeficientes de las ecuaciones mencionadas anteriormente, aquí los coeficientes de la spline paramétrica contienen tres coordenadas y por lo tanto, se requieren calcular tres conjuntos de coeficientes. Con el fin de simplificar las cosas, se introducen los vectores siguientes:

$$\mathbf{a_k} \equiv \begin{bmatrix} A_{xk} \\ A_{yk} \\ A_{zk} \end{bmatrix}, \quad \mathbf{b_k} \equiv \begin{bmatrix} B_{xk} \\ B_{yk} \\ B_{zk} \end{bmatrix}, \quad \mathbf{c_k} \equiv \begin{bmatrix} C_{xk} \\ C_{yk} \\ C_{zk} \end{bmatrix}, \quad \mathbf{d_k} \equiv \begin{bmatrix} D_{xk} \\ D_{yk} \\ D_{zk} \end{bmatrix} \tag{3.83}$$

Y así, el vector posición de cualquier punto arbitrario P en la curva paramétrica spline toma la forma

$$\mathbf{p}(\sigma) = \mathbf{a_k}(\sigma-\sigma_k)^3+\mathbf{b_k}(\sigma-\sigma_k)^2+\mathbf{c_k}(\sigma-\sigma_k)+\mathbf{d_k}, \quad k=1,\ldots,N-1 \tag{3.84}$$

En el intervalo $\sigma_k \leq \sigma \leq \sigma_{k+1}$. Entonces, el conjunto contraparte de las ecuaciones (3.47) − (3.51) es,

$$\mathbf{a_k} = \frac{1}{6\Delta\sigma_k}\left(\mathbf{p}''_{k+1} - \mathbf{p}''_k\right) \tag{3.85}$$

$$\mathbf{b_k} = \frac{1}{2}\mathbf{p}''_k \tag{3.86}$$

$$\mathbf{c_k} = \frac{\Delta\mathbf{p}_k}{\Delta\sigma_k} - \frac{1}{6}\Delta\sigma_k\left(\mathbf{p}''_{k+1} + 2\mathbf{p}''_k\right) \tag{3.87}$$

$$\mathbf{d_k} = \mathbf{p}_k \tag{3.88}$$

$$\Delta\mathbf{p}_k = \mathbf{p}_{k+1} - \mathbf{p}_k \tag{3.89}$$

mientras que los vectores $\mathbf{p}_k$ y $\mathbf{p}''_k$ están definidos como

$$\mathbf{p}_k \equiv \begin{bmatrix} x_k \\ y_k \\ z_k \end{bmatrix}, \qquad \mathbf{p}''_k \equiv \begin{bmatrix} x''_k \\ y''_k \\ z''_k \end{bmatrix} \tag{3.90}$$

Al considerar que $\mathbf{p}$ es una función cúbica definida a trozos en σ, $\mathbf{p}'$ es una función cuadrática a trozos, mientras que $\mathbf{p}''$ es una función lineal a trozos, $\mathbf{p}'''$ es una constante definida a trozos; en las derivadas de orden mayor sus argumentos van desapareciendo. Sin embargo, la constante a trozos de $\mathbf{p}'''$ hace que la cuarta derivada sea discontinua en los puntos de apoyo definidos en la curva, y en consecuencia, todas las derivadas de orden superior a ésta son discontinuas para estos puntos.

Además, las matrices P y P$''$ de tamaño $N \times 3$ están definidas como

$$\mathrm{P} \equiv \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \\ \vdots \\ \mathbf{p}_N^T \end{bmatrix}, \qquad \mathrm{P}'' \equiv \begin{bmatrix} (\mathbf{p}_1'')^T \\ (\mathbf{p}_2'')^T \\ \vdots \\ (\mathbf{p}_N'')^T \end{bmatrix} \tag{3.91}$$

Lo que permite reescribir las ecuaciones matriciales en la forma

$$\mathrm{AP}'' = 6\mathrm{CP} \tag{3.92}$$

Ahora, los coeficientes de la curva spline $\mathbf{a}_k, \cdots, \mathbf{d}_k$ pueden ser calculados una vez que se tenga calculado el vector $\mathbf{p}_k''$. Este vector puede ser calculado de la matriz (3.92) con P$''$. Sin embargo, encontrar esta solución requiere invertir la matriz A de tamaño $(N-2)\times N$, la cual no es cuadrada y por tanto no puede ser *invertida*, propiamente hablando. Así pues, se tiene un sistema indeterminado de ecuaciones lineales, y se necesitan más coeficientes con el fin de hacer un sistema determinado. Tales condiciones son las que definen el tipo de función spline que se desea manejar. Por ejemplo, trazos o trayectos cerrados son llamados **splines periódicas**, mientras que las trayectorias abiertas son llamadas como **splines naturales**. Las condiciones impuestas a las splines paramétricas periódicas son las siguientes:

$$\mathbf{p}_N = \mathbf{p}_1, \qquad \mathbf{p}_N' = \mathbf{p}_1', \qquad \mathbf{p}_N'' = \mathbf{p}_1'' \tag{3.93}$$

Por otra parte, las splines paramétricas naturales se obtienen bajo las condiciones siguientes

$$\mathbf{p}_1'' = \mathbf{p}_N'' = \mathbf{0} \tag{3.94}$$

De este modo si las splines paramétricas periódicas son requeridas, entonces los vectores $\mathbf{p}_N$ y $\mathbf{p}''_N$ pueden ser eliminados de las matrices P y P$''$, respectivamente, éstas entonces se convertirán en matrices de $(N-1)\times 3$, esto es

$$\mathrm{P} \equiv \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \\ \vdots \\ \mathbf{p}_{N-1}^T \end{bmatrix}, \qquad \mathrm{P}'' \equiv \begin{bmatrix} (\mathbf{p}_1'')^T \\ (\mathbf{p}_2'')^T \\ \vdots \\ (\mathbf{p}_{N-1}'')^T \end{bmatrix} \tag{3.95}$$

Además, se añade la condición de la primera derivada de la ecuación (3.93) a la condición de continuidad $N-2$ de la ecuación (3.52), obteniendo de este modo ecuaciones de la forma $N-1$. En consecuencia la matriz A se convierte en $(N-1)\times(N-1)$. Correspondientemente, la matriz C se convierte de tamaño $(N-1)\times(N-1)$, es decir,

$$\mathrm{A} \equiv \begin{bmatrix} 2\alpha_{1,N'} & \alpha_1 & 0 & 0 & \cdots & \alpha_{N'} \\ \alpha_1 & 2\alpha_{1,2} & \alpha_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \alpha_{N'''} & 2\alpha_{N''',N''} & \alpha_{N''} \\ \alpha_{N'} & 0 & 0 & \cdots & \alpha_{N''} & 2\alpha_{N'',N'} \end{bmatrix} \tag{3.96}$$

$$\mathrm{C} \equiv \begin{bmatrix} -\beta_{1,N'} & \beta_1 & 0 & 0 & \cdots & \beta_{N'} \\ \beta_1 & -\beta_{1,2} & \beta_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \beta_{N'''} & -\beta_{N''',N''} & \beta_{N''} \\ \beta_{N'} & 0 & 0 & \cdots & \beta_{N''} & -\beta_{N'',N'} \end{bmatrix} \tag{3.97}$$

Ahora que la matriz A es cuadrada, propiamente hablando, se puede invertir la matriz A para calcular P$''$

$$\mathrm{P}'' = 6\mathrm{A}^{-1}\mathrm{CP} \tag{3.98}$$

de este modo calculados todos los vectores $\{\mathbf{p}_k''\}_1^{N-1}$, del cual $\mathbf{p}''_N$ puede ser obtenido con la ecuación anterior.

Del mismo modo, si se usa la spline paramétrica natural, entonces la matriz P$''$ es de $(N-2)\times 3$, mientras la matriz A, consecuentemente es de $(N-2)\times(N-2)$, como está dada en la Ec. (3.60), y la matriz C es la misma que se encuentra en la ecuación (3.80).

$$\mathrm{A} \equiv \begin{bmatrix} 2\alpha_{1,2} & \alpha_2 & 0 & \cdots & 0 \\ \alpha_2 & 2\alpha_{2,3} & \alpha_3 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \alpha_{N'''} & 2\alpha_{N''',N''} & \alpha_{N''} \\ 0 & 0 & \cdots & \alpha_{N''} & 2\alpha_{N'',N'} \end{bmatrix} \tag{3.99}$$

$$\mathrm{C} \equiv \begin{bmatrix} \beta_1 & -\beta_{1,2} & \beta_2 & 0 & \cdots & 0 & 0 \\ 0 & \beta_2 & -\beta_{2,3} & \beta_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \beta_{N'''} & -\beta_{N''',N''} & \beta_{N''} & 0 \\ 0 & 0 & 0 & \cdots & \beta_{N''} & -\beta_{N'',N'} & \beta_{N'} \end{bmatrix} \tag{3.100}$$

Finalmente el capítulo demuestra el funcionamiento teórico y matemático de las posibles interpolaciones en las trayectorias que un robot puede trazar con el efector final. Particularmente las trayectorias continuas curvilíneas son de mayor desafío para su ejecución. En el siguiente capítulo se explicará a detalle el funcionamiento de la plataforma de simulación del robot para la ejecución de diversas trayectorias en el espacio con aplicaciones industriales.

Capítulo 4

Simulación en ADEFID

Contenido

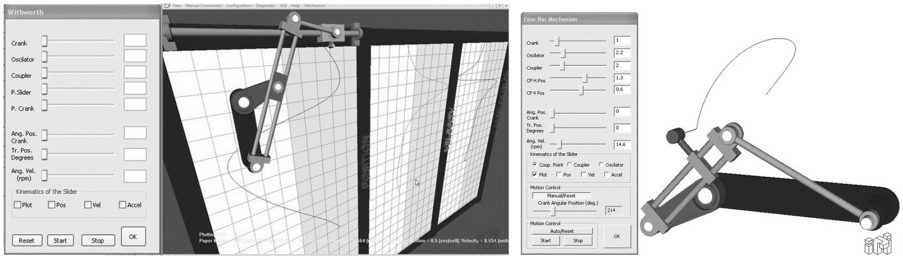
En el presente capítulo se introduce la herramienta de software en la cual se desarrolló la presente tesis. Este software visualiza el funcionamiento y control del manipulador articulado para aplicaciones industriales con diversas tareas. Las secciones 4.1 a 4.6, presentan las diversas aplicaciones y bibliotecas con que cuenta el software para aclarar el funcionamiento del mismo. Y las últimas secciones (4.7 - 4.8) presentan la contribución principal de esta tesis.

4.1 ADEFID.

`ADEFID` (ADvanced Engineering platForm for Industrial Development) [13] es una plataforma de desarrollo de software escrito en `MS Visual Studio C++` [21], con clases `MFC` (Microsoft Fundation Clases) y con apoyo de librerías OpenGL, pensada como una herramienta para diseño y simulación de dispositivos y algoritmos de desarrollo industrial.

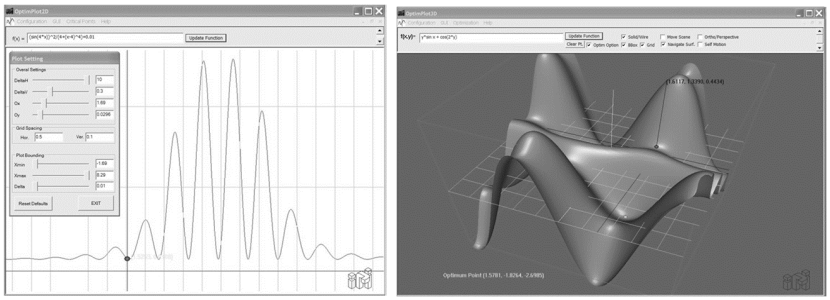
Este software ha estado desarrollándose de forma gradual para diversas aplicaciones de investigación e industriales, las bibliotecas han estado creciendo conforme a las necesidades surgidas durante el desarrollo de paquetes de software como `Mechanism-O`, `Vibrato`, `OptimPlot2D`, `OptimPlot3D`, `ADRS`, entre otros [22]. Las imágenes de algunas de estas aplicaciones se muestran en las Figuras 4.1a - 4.2. Y a partir de proyectos ya existentes hechos en `ADEFID`, se pueden generar nuevas aplicaciones. En el trabajo descrito por Peña-Gallo [23], se explican con más detalle los pasos necesarios para la creación de nuevos programas en relación con aplicaciones ya existentes.

El desarrollador puede crear aplicaciones personalizadas a partir de cero (Aplicación A) o desde una aplicación ya desarrollada en `ADEFID` (Aplicación B-Aplicación N), con el fin de implementar aplicaciones de alto nivel y flexibilidad en su manejo (ver Fig.4.4). Un proyecto `ADEFID` consta de diversos archivos, éstos se agrupan en una carpeta con el nombre `Ipi10`, que contiene otros directorios: `Bin`, `GL`, `Include`, `Lib`, `Projects` y `STL`. La carpeta `Bin` contiene los ejecutables de las aplicaciones desarrolladas en la plataforma. La carpeta `GL` contiene la librería principal para llamar funciones OpenGL, a su vez las carpetas `Include` y `Lib` contienen principal-



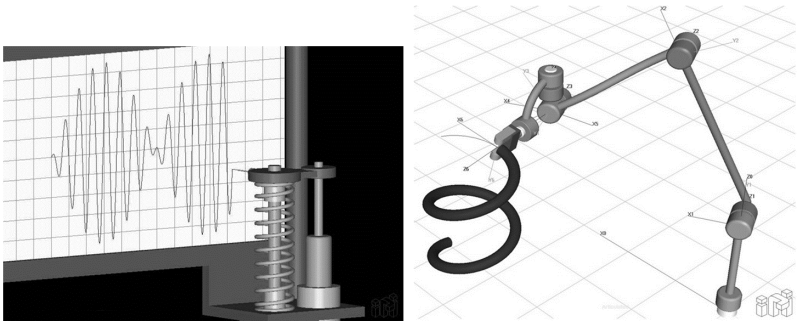
(a) Simulación de mecanismo Whitworth. (b) Mecanismo de cuatro barras trazando una curva.

Figura 4.1: Mechanism-O [22].



(a) Imágen fija de OptimPlot2D. (b) Imágen fija de OptimPlot3D.

Figura 4.2: Aplicaciones OptimPlot2D y OptimPlot3D [22].



(a) Vibrato: sistema de un grado de libertad. (b) ADRS.

Figura 4.3: Aplicaciones Vibrato y ADRS en la plataforma ADEFID [22].

mente librerías y cabeceras de aplicaciones básicas y esenciales de `ADEFID`. La carpeta `STL` contiene todos los diseños y archivos CAD de manipuladores, robots, herramientas y prototipos en 3D para una visualización más real en la plataforma `ADEFID`. Y finalmente la carpeta Projects contiene los proyectos y clases necesarias para trabajar en `Visual Studio C++`.

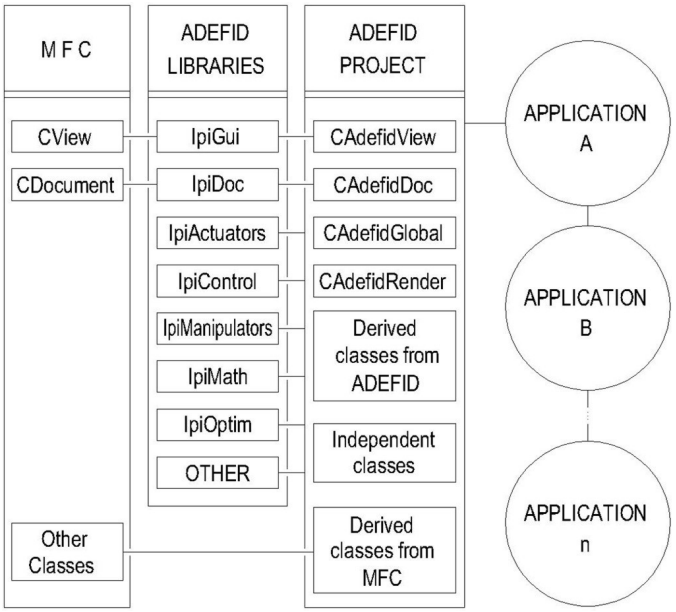


Figura 4.4: Diagrama de clases y librerías ADEFID [22]

La librería `MFC` provee clases básicas, como parte de la arquitectura Document/View, éstas son la clase CDocument y CView. De esta forma las clases `CADEFIDDoc` y `CADEFIDView` son elementos fundamentales en los proyectos basados en `ADEFID`. La clase `CADEFIDDoc` funciona como un administrador /director de todo el proyecto en el software, y la clase `CADEFIDRender`, cuyo propósito principal es generar los objetos gráficos y los movimientos de los mismos que aparecen en la visualización de la pantalla [22].

La clase **CADEFIDDoc** es la clase principal del proyecto, lleva a cabo procesos de inicialización de máquinas, ejecuta el algoritmo de manipulación de procesos conocido como `MaterialHandingProces()` que se mantiene activo en un ciclo infinito que sólo se termina si el usuario cierra la aplicación. Lo cual hace que la aplicación funcione como un sistema autónomo e independiente.

La parte visual de cada proyecto en `ADEFID` está relacionada con las clases `CADEFIDView` y `CADEFIDRender`. La clase **CADEFIDView** se encarga de manipular el entorno gráfico de la plataforma, esto es, se pueden agregar sentencias que involucran cambios en la apariencia de la escena, como cambios del color de la escena, entre otros. En esta clase se manejan variables relacionadas con transformaciones y rotaciones de la escena y de los objetos en ella.

La clase **CADEFIDRender** se encarga del manejo de los objetos presentes dentro de la escena del proyecto, es posible mostrar diversos objetos en la escena. Dentro de esta clase se manejan parámetros de las librerías OpenGL relacionados con la apariencia de los objetos que se agregan en la escena. `CADEFIDRender` cuenta con dos funciones principales: `SetupScene()` y `RenderUScene()`, la primer función sirve para generar el modelo gráfico, esto es, la función define los componentes en la escena y solo se llaman cuando se inicializa el proceso. La segunda función despliega el modelo y produce el efecto visual de movimiento, sin embargo, esta función no genera el modelo, solamente lo llama para aplicar transformaciones de movimiento, como rotación o traslación. `RenderUScene()` está en constante actualización para tener una visualización real en la plataforma.

4.2 ADRS.

La aplicación en la que se desarrolla la presente tesis, está basada en la herramienta **ADRS** (**A**rchitecture **D**esign and **R**obot **S**imulation) [24], la cual permite obtener un alto desempeño gráfico en el diseño y simulación de arquitecturas convencionales y no convencionales en manipuladores seriales, realizando cálculos con la cinemática del manipulador. `ADRS` está en continuo desarrollo, sin embargo, su etapa actual proporciona al usuario las herramientas para diseñar de forma interactiva una arquitectura robótica deseada, en otras palabras, los parámetros de diseño, como longitud, offsets y ángulos de rotación se pueden modificar en tiempo real mientras que su estructura se va actualizando constantemente.

`ADRS` fue desarrollado con características de simulación como seguimiento de la trayectoria, la generación de trayectorias, paletización, manipulación

de objetos como pick-and-place, el cálculo de la cinemática directa e inversa del manipulador serial. La figura 4.5 muestra un caso de seguimiento de trayectoria.

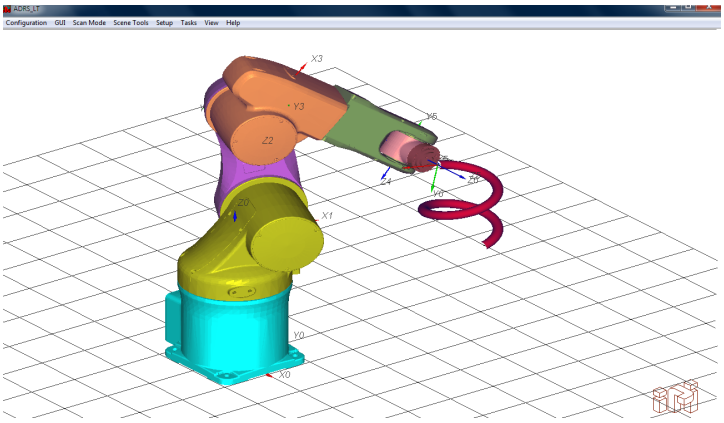


Figura 4.5: Aplicación de seguimiento de la trayectoria en ADRS.

De acuerdo con la figura 4.4, `ADRS` es un ejemplo de una aplicación B, ya que su precursor es SnAM [25] que sería la aplicación A. `ADRS` contiene funciones que ayudan a la simulación de un brazo robótico de hasta veinte eslabones. Sin embargo, con las necesidades que hoy en día surgen en la industria se ha podido extender y mejorar el programa con nuevas aplicaciones, por ejemplo, la operación pick & place, que ahora se puede tomar como una aplicación C, como se muestra en la figura 4.6, en el que el usuario no necesita volver a escribir el código para el manipulador, siendo éste un objeto de la clase **CSerialRobot** que deriva todos los cálculos y especificaciones cinemáticas del robot y su algoritmo de control de la librería `IpiManipulators`.

El diagrama de flujo principal de un proyecto en `ADEFID` como `ADRS` reside en la clase `CADEFIDDoc` (ver Fig. 4.7). Cuando se inicializa la aplicación, el sistema manda mensajes para indicar la conexión con dispositivos integrados (I/O cards) o sistemas de conexión remota (Ethernet). Una vez establecida la conexión se inicializan los objetos de todas las máquinas. Una máquina puede ser un dispositivo físico o virtual. El proceso de inicio principal (Start Main Process) es un bucle infinito que termina si el usuario cierra la aplicación. Se leen las entradas y salidas de los dispositivos y si se activa

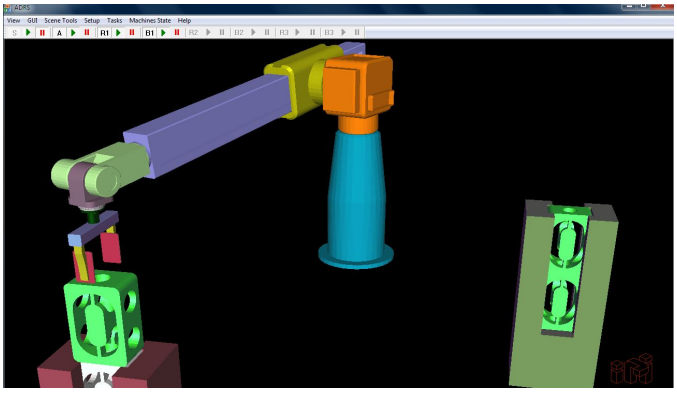


Figura 4.6: Operación pick-and-place [28].

cualquier entrada en los dispositivos de seguridad (SDI), el sistema cae en un estado de suspensión y no hay acción en la plataforma. Por otro lado si las entradas SDI permanecen desactivadas, el estado de cada máquina se evalúa y se actualiza de acuerdo a su estado correspondiente. Finalmente, el sistema ejecuta tareas predefinidas por el algoritmo de la máquina.

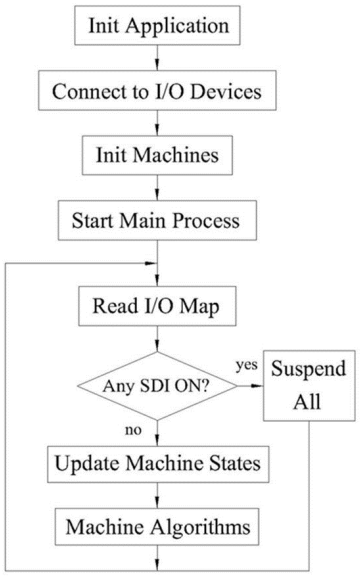


Figura 4.7: Diagrama de flujo ADEFID [22].

El funcionamiento de `ADRS` permite obtener un alto desempeño gráfico en el diseño y simulación "en-línea", de arquitecturas convencionales y no convencionales en manipuladores seriales. Se entiende por "en-línea", que

los cálculos relacionados con la cinemática de manipuladores se hacen simultáneamente con los cambios de los parámetros que definen la arquitectura y así la visualización de su transformación de forma suavizada. La necesidad de realizar dichos cálculos en conjunto con la ayuda visual, motivó a generar una formulación cerrada para resolver la cinemática inversa de manipuladores seriales industriales. De tal manera, con la finalidad de contar con una solución unificada, se presenta un estudio en el que se hace una clasificación con base a sus tres primeras articulaciones.

4.3 Notación en la arquitectura de los manipuladores.

La notación Denavit-Hartenberg define cuatro parámetros, es decir, ángulo (θ), desviación (b), distancia (a) y ángulo de torsión (α), y se enfoca a manipuladores seriales desacoplados con arquitectura ortogonal [24].

La característica más importante de los resultados de este estudio, es que la mayoría de los manipuladores industriales se basa en los valores de los ángulos de torsión de las tres primeras articulaciones. La solución cerrada de la cinemática inversa que se desarrolla para el cálculo de manipuladores de 6 grados de libertad desacoplados se logra cuando estos ángulos tienen valores dados. Por lo tanto, α_1 puede ser 90° ó -90°, si el manipulador es de hombro derecho (R) o de hombro izquierdo (L), respectivamente. Mientras que el codo es considerado externo (E) o interno (I) si α_2 es 0° ó 180°, respectivamente. Y el antebrazo puede estar dispuesto hacia abajo (D), hacia arriba (U), u horizontalmente (H), dependiendo del valor de α_3. En la Fig. 4.8 se muestra una representación gráfica de la nomenclatura que se definió con los parámetros previos. Esta convención, se basa en la configuración cero del manipulador, es decir, cuando todas las variables de articulación valen cero.

Una vez que la distancia y la desviación de cada articulación son establecidas, existen 16 arquitecturas posibles dependiendo de la combinación de los valores de los ángulos de torsión. Por simplicidad y porque además no intervienen en la clasificación, se omiten las últimas tres articulaciones que definen la muñeca.

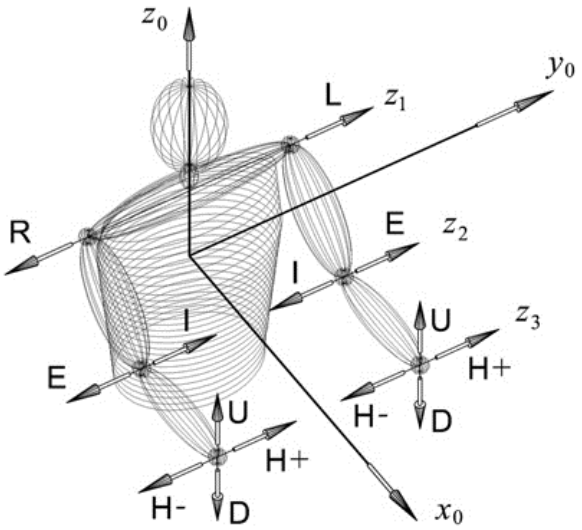


Figura 4.8: Nomenclatura de la clasificación antropomórfica [24].

La arquitectura de los manipuladores que frecuentemente se encuentran en la industria, queda comprendida por la siguiente clasificación. La tabla 4.1 muestra las 16 posibles combinaciones de ensambles que están asociados con los ángulos de torsión del hombro, codo y antebrazo del manipulador. Adicionalmente, la columna derecha de la tabla 4.1 indica cuales de los parámetros DH se requieren para definir el antebrazo.

Así, manipuladores con hombro derecho y codo externo se muestran en la figura 4.9, mientras que aquellos con hombro derecho y codo interior se muestran en la figura 4.10. De la misma manera pero con hombro izquierdo, las figuras 4.11 y 4.12 muestran manipuladores con un codo exterior e interior, respectivamente. Estas figuras fueron generadas con la configuración cero.

En algunos casos, los manipuladores con codo interno y con simetría en el diseño, pueden ser tratados como hombro derecho o izquierdo indistintamente. En tales casos, “R”o “L”de la nomenclatura se substituye por “S”. Tal es el caso del manipulador Fanuc que se muestra en la figura 4.13.

Tabla 4.1: Clasificación antropomórfica con base en los ángulos de torsión [24].

Arquitectura	Descripción				α_1 (grados)	α_2 (grados)	α_3 (grados)	Parámetro que define el antebrazo
RED	Hombro derecho	Codo Ext.	Antebrazo abajo		90	0	90	α_3 y/o β_4
REU			Antebrazo arriba				−90	
REH+			Antebrazo horizontal	Muñeca Ext.			0	α_3
REH−				Muñeca Int.			180	
RID		Codo Int.	Antebrazo abajo			180	−90	α_3 y/o β_4
RIU			Antebrazo arriba				90	
RIH+			Antebrazo horizontal	Muñeca Ext.			180	α_3
RIH−				Muñeca Int.			0	
LED	Hombro izquierdo	Codo Ext.	Antebrazo abajo		−90	0	−90	α_3 y/o β_4
LEU			Antebrazo arriba				90	
LEH+			Antebrazo horizontal	Muñeca Ext.			0	α_3
LEH−				Muñeca Int.			180	
LID		Codo Int.	Antebrazo abajo			180	90	α_3 y/o β_4
LIU			Antebrazo arriba				−90	
LIH+			Antebrazo horizontal	Muñeca Ext.			180	α_3
LIH−				Muñeca Int.			0	

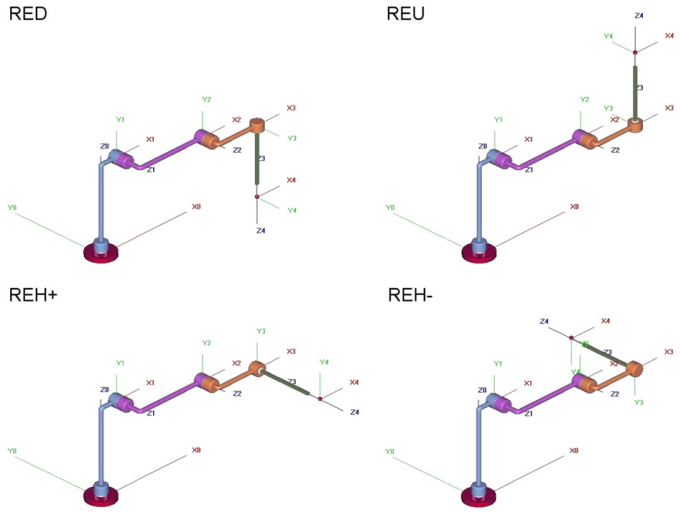


Figura 4.9: Manipuladores con hombro derecho codo externo [24].

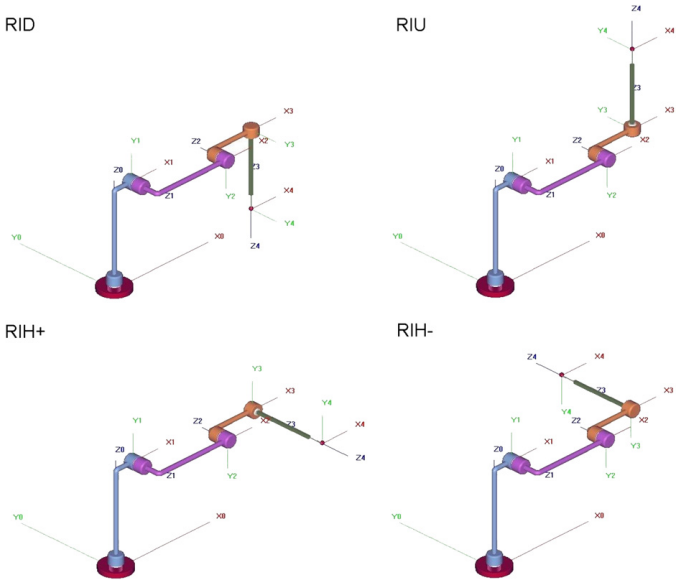


Figura 4.10: Manipuladores con hombro derecho y codo interno [24].

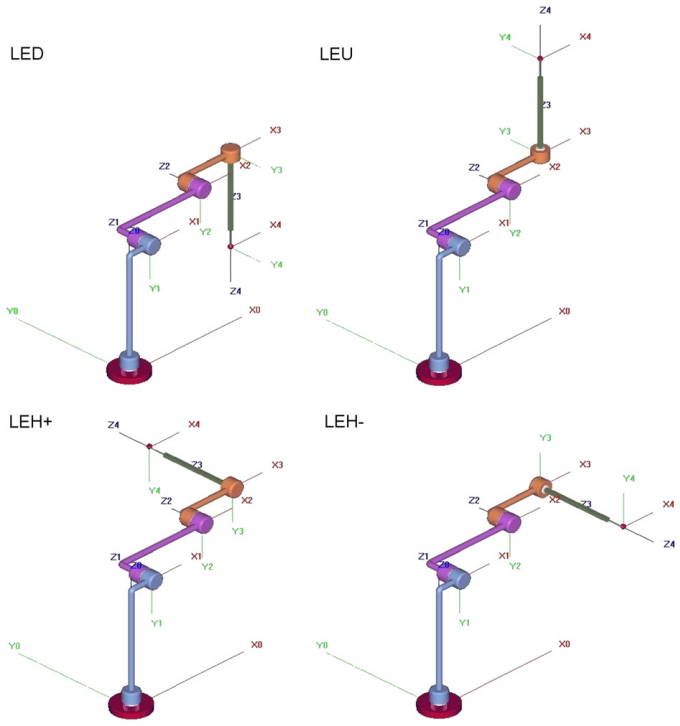


Figura 4.11: Manipuladores con hombro izquierdo y codo externo [24].

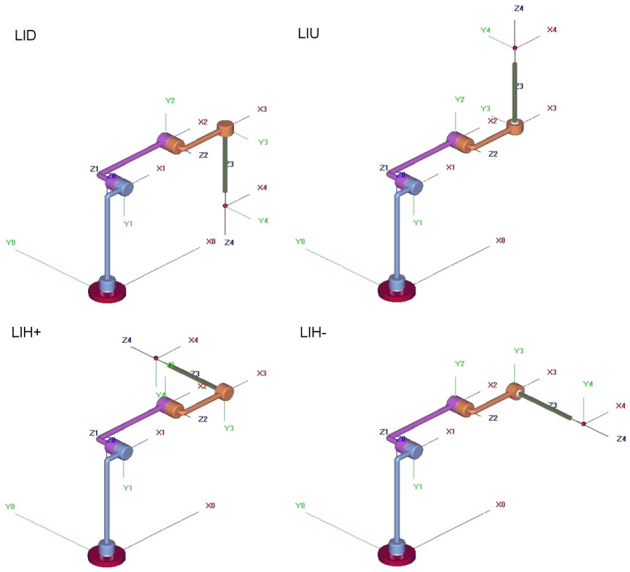


Figura 4.12: Manipuladores con hombro izquierdo y codo interno [24].

Figura 4.13: FANUC LR Mate 200*i*D/4SC [26].

4.4 Bibliotecas de ADRS.

`ADRS` contiene funciones que ayudan a la simulación de un brazo robótico de hasta veinte eslabones. Dentro de las bibliotecas que contiene ADRS se encuentran funciones que permiten realizar el análisis cinemático [27]. Las

funciones están divididas principalmente en cuatro clases:

- **CVectors**. Es la clase de vectores que contiene funciones para el cálculo de operaciones con vectores y matrices. Las funciones integran operaciones como el producto punto, producto cruz, magnitud de un vector, operaciones con matrices y funciones para obtener la matriz de orientación a partir de los ángulos de Euler. Estos ángulos describen la orientación de un cuerpo rígido por medio del producto de tres matrices de rotación, las cuales se basan en ángulos de giro, desviación y elevación (mejor conocidos en inglés como roll, pitch and yaw).

- **CRobot**. La clase Robot integra funciones útiles para las características geométricas del manipulador robótico. En esta biblioteca se incluyen funciones para la actualización de los parámetros geométricos, el dibujo del manipulador y el dibujo del entorno.

- **CRoboKin**. La clase RoboKin contiene funciones para el uso de cálculos cinemáticos, como la cinemática directa e inversa. Esta clase hereda las propiedades de la clase Robot.

- **CSerialRobot**. La clase SerialRobot es la base para los manipuladores seriales, de entre los cuales se encuentran la clasificación de Articulados, Cartesianos, Cilíndricos, Esféricos, Stanford, SCARA, entre otros. Es la clase base donde se hace el llamado de las variables y geometría del robot a la plataforma, también es fundamental para realizar tareas y aplicaciones deseadas para el robot en el ambiente gráfico.

Las clases que involucran la planificación de trayectorias para el manipulador son las siguientes.

- **CLinearPath**. Esta clase es fundamental para la planificación de trayectorias lineales en un espacio tridimensional. Con base en dos puntos en el espacio, el punto inicial y final, es posible crear los cálculos para el seguimiento de trayectos lineales para el efector final del manipulador.

- **CParametricSplines**. La clase CParametricSplines es aquella que hace todas las operaciones vectoriales y matriciales de la función Spline, la cual, es la planificación de trayectorias continuas de forma curvilínea en un espacio tridimensional. Teniendo como valores conocidos todos los puntos de apoyo que se desean que conformen una trayectoria curvilínea en un espacio tridimensional, es posible el trazo de la trayectoria continua con el efector final del manipulador. La teoría relacionada con la implementación de esta clase se expuso en el capítulo anterior y es una contribución de este proyecto de tesis.

4.5 Cinemática de los manipuladores seriales.

Una parte importante que conforma el paquete `ADRS` es la solución cinemática directa e inversa de los manipuladores seriales desacoplados. Esta formulación es aplicada en el robot utilizado en el caso de estudio desarrollado en esta tesis.

4.5.1 Cinemática directa.

Como se ha mencionado en capítulos anteriores el objetivo de la **cinemática directa** es determinar el efecto acumulado del conjunto que forman las variables de articulación, esto es, determinar la posición y orientación del efector final dados los valores de dichas variables. La librería **CRoboKin** contiene una función para el cálculo de la cinemática directa. La función es miembro de la clase **CRobot** y es conocida como *FullForwardKinematics*() que hace uso de la matriz de transformación homogénea de la ecuación (2.7).

$$T_j^i = A_{i+1} \cdots A_j = \begin{bmatrix} R_j^i & \mathbf{o}_j^i \\ \mathbf{0} & 1 \end{bmatrix}$$

La función está definida como:

Matrix **FullForwardKinematics**(D_H^* $pDHParam$)

Donde $pDHParam$ es del tipo D_H, que es una estructura definida en la clase **CRobot** y es la variable que contiene la dirección donde se encuentran los parámetros de Denavit Hartenberg. La función regresa una matriz del

tipo Matrix (definido en la clase **CVector**) y es definida por la Matriz T de la ecuación (2.7).

4.5.2 Parámetros Denavit-Hartenberg.

La notación Denavit-Hartenberg (DH) es necesaria para describir el modelo cinemático del robot, de manera que cada matriz de transformación homogénea es el producto de cuatro transformaciones homogéneas independientes A_i, donde los parámetros DH corresponden a una operación de traslación o rotación en un eje determinado.

$$A_i = \text{Rotación}_{z,\theta_i}, \text{Traslación}_{z,b_i}, \text{Traslación}_{x,a_i}, \text{Rotación}_{x,\alpha_i}$$

Donde las cuatro variables θ_i, a_i, b_i y α_i, son parámetros del eslabón i y la articulación $i-1$.

Realizando la multiplicación de matrices, se obtiene una matriz de transformación homogénea (ecuación (2.12)), que involucra todos los parámetros DH, de este modo se puede crear una matriz para modelar cada uno de los eslabones del brazo.

$$A_i = \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & b_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

De manera que para incluir los parámetros DH de cada manipulador, es necesario una clase específica para el manipulador y en la función $SetDefaults$ se definen todas las variables θ_i, a_i, b_i y α_i.

4.5.3 Cinemática inversa.

Por otro lado, la **cinemática inversa** busca determinar el valor de las variables de articulación conociendo la posición y orientación del efector final. La solución para este problema es de fundamental importancia para

especificar el orden de las condiciones en el espacio articular correspondiente al movimiento, permitiendo la ejecución del movimiento deseado.

La librería **CRobokin** integra una función que calcula la cinemática inversa. La función `SetFullInverse()` es aplicada para resolver manipuladores de tipo 6R, el pseudocódigo está escrito de tal manera que puede ser extendido a cualquier manipulador serial de n-ejes con las últimas tres articulaciones con ejes intersectándose mutuamente. La diferencia de 6 a n-ejes reside en la línea de código donde la solución correspondiente a la cinemática inversa de posición es llamada por la función `SetInversePosition()` [24].

La función está definida como:

BOOL **SetFullInverse** (D_H^* $pDHParam$, Matrix H_mat)

BOOL **SetFullInverse** (D_H^* $pDHParam$, Matrix Q_mat, CartPoint d_vect)

Existen dos funciones debido a que se puede conocer la matriz de transformación homogénea como una variable independiente y de la otra función se puede conocer por separado la matriz de orientación y el vector de posición que conforman la matriz de transformación homogénea.

Tres argumentos deben pasarse cuando la función `SetFullInverse()` es llamada, las cuales son,

- Los parámetros DH del manipulador a través de la estructura definida como DH la cual guarda los valores θ_i, b_i, a_i y α_i, con $i = 1, 2, 3, \ldots, n$.
- La matriz de orientación deseada del efector final dada por la matriz de rotación Q.
- El vector de posición **d** deseado para el punto de interés en el efector final del manipulador.

La función regresa un valor booleano para indicar si la función terminó de manera correcta, además se modifica el parámetro de entrada definido como H_mat que es donde se almacenan las variables de la articulación. Las figuras auxiliares utilizadas para la formulación están basadas en el manipulador tipo RED. De forma que el lado izquierdo de la Fig. 4.14

muestra los sistemas de coordenadas asignados a cada eslabón, mientras que el lado derecho presenta la distancia y los offsets de cada eslabón. Desde el origen del sistema de coordenadas fijo, el vértice opuesto del paralelepípedo representa el punto central de la muñeca.

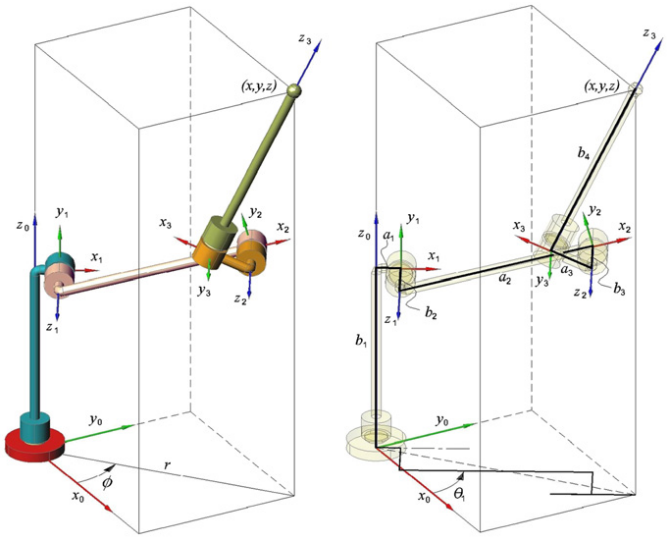


Figura 4.14: Sistemas de coordenadas asignados a un manipulador tipo RED [24].

La función `SetInversePosition` es llamada mientras `SetFullInverse` está corriendo [24]. Con el fin de hacer los cálculos de las variables utilizadas en la geometría del manipulador y finalmente obtener las variables de articulación, desde el primer eslabón hasta el punto central de la muñeca del manipulador, de ahí depende la cantidad de eslabones del robot, si es un manipulador 6R, solamente se calculan las primeras tres articulaciones $\theta_1,\ \theta_2,\ \theta_3$ que conforman la posición del manipulador en un espacio cartesiano. Con el fin de entender mejor el proceso de esta función, la geometría que soporta las variables utilizadas se muestra en la Fig. 2.31.

La función está definida como:

BOOL **SetInversePosition** (D_H^* pDH, CartPoint $point$)

Dos argumentos deben pasarse cuando la función `SetInversePosition()` es llamada, los cuales son las siguientes:

- Los parámetros DH del manipulador a través de la estructura definida como DH la cual guarda los valores θ_i, b_i, a_i y α_i, con $i = 1, 2, 3, \ldots, n$.
- **p**: El vector de posición del punto en el que se intersectan los ejes que definen las variables de articulación de la muñeca.

La función `SetInversePosition()` regresa un valor booleano para indicar si la función terminó de manera correcta, en esta función se consideran los 16 posibles casos dados en la Tabla 4.1, ya que los correspondientes valores de $\alpha_1,\ \alpha_2,\ \alpha_3$ están integrados.

4.6 Seguimiento de trayectorias lineales.

Como se ha mencionado anteriormente, la aplicación consiste en tener desplazamientos lineales y continuos en un espacio tridimensional para un manipulador serial, en este caso particular se estará implementando a manipuladores articulados 6R. Y en esta sección se presentarán las diversas funciones utilizadas para el movimiento y seguimiento del efector final del manipulador con trayectorias lineales.

Entonces se tienen estos puntos importantes para generar una trayectoria lineal en la plataforma `ADEFID`.

- Generación de poses discretas con respecto a la parametrización de una variable.
- Su funciomaniento radica en el conocimiento de las poses extremas de la trayectoria lineal.
- Cualquier pose deseada del efector final está representada por la matriz de transformación homogénea (Ec. (4.1))

$$T = \begin{bmatrix} Q & \mathbf{d} \\ \mathbf{0} & 1 \end{bmatrix} \tag{4.1}$$

Cada una de estas consideraciones es esencial para la generación de trayectorias lineales, y están desarrolladas en la clase **CLinearPath**.

4.6.1 Clase CLinearPath.

Con el fin de entender la distribución y estudio que presenta la Clase **CLinearPath**, se presentan los diferentes algoritmos de las funciones principales de la clase. El algoritmo tiene el propósito de generar poses discretas por un parámetro y su resultado está basado en el conocimiento de las posturas inicial y final. Cualquier pose deseada del efector final está representada por su sistema de coordenadas asociado a una matriz homogénea T que contiene la matriz Q y el vector $\mathbf{d}$, que representan respectivamente, la orientación y el vector posición con respecto a un sistema de referencia dado [28].

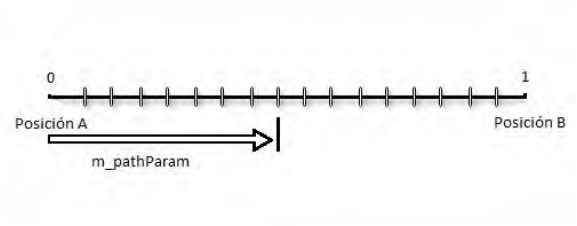


Figura 4.15: Parámetro η.

Entonces, teniendo conocimiento de las poses inicial y final, definidas por T_I y T_F, las poses intermedias T_m se obtienen con la especificación del parámetro η, con valores discretizados con $0 \leq \eta \leq 1$ (ver Fig. 4.15). Estas poses intermedias forman una trayectoria lineal con base en las dos poses ya conocidas. Sin embargo, para lograr esto, es necesario definir algunos invariantes, como el vector $\mathbf{d}$, que define el vector de posición del origen de la trama S_F con respecto a la trama S_I, y los ángulos roll, pitch and yaw, que se definen como φ_x, φ_y y φ_z. Estos ángulos definen la orientación del marco S_F con respecto al bastidor S_I mediante la realización de rotaciones básicas sobre x_I, y_I y z_I, representadas como $Q_x(\varphi_x)$, $Q_y(\varphi_y)$, y $Q_z(\varphi_z)$, respectivamente. En la figura 4.16 se observa la relación entre las posturas mencionadas. La función que evalúa los invariantes se llama `SetInvariants()` y el pseudocódigo 1 presenta su desarrollo.

Los argumentos que pasan por `SetInvariants()`, son las poses T_I y T_F, estas dos son las entradas necesarias para hacer todo el procesamiento del cálculo de los invariantes, como se observa en el Algoritmo 1, hay otras funciones que se van llamando mientras `SetInvariants()` se está ejecutando.

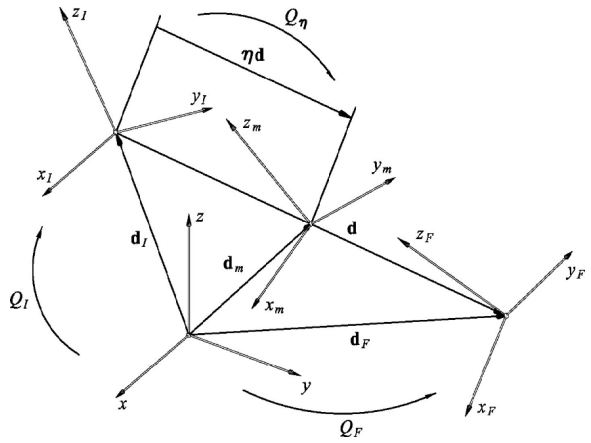


Figura 4.16: Seguimiento de una trayectoria lineal definiendo poses inicial y final [28].

Pseudocódigo 1 *SetInvariants()* de la clase *CLinearPath*

function SETINVARIANTS(T_I, T_F)

$Q_I \leftarrow$ GetRotMatFromPose(T_I)

$Q_F \leftarrow$ GetRotMatFromPose(T_F)

$\mathbf{d_I} \leftarrow$ GetVecMatFromPose(T_I)

$\mathbf{d_F} \leftarrow$ GetVecMatFromPose(T_F)

$\varphi_x, \varphi_y, \varphi_z \leftarrow$ GetRPYAngles($Q_I^T Q_F$)

$\mathbf{d} \leftarrow \mathbf{d_F} - \mathbf{d_I}$

end function

`GetRotMatFromPose()` es llamada dos veces para calcular las matrices de rotación Q_I y Q_F tanto para la pose inicial y la pose final T_I y T_F. La función `GetVecMatFromPose()` de igual manera es llamada dos veces para extraer el vector de posición de las poses inicial T_I y final T_F. Ahora bien, la función `GetRPYAngles()` regresa los valores de los ángulos roll, pitch and yaw resolviendo la siguiente ecuación:

$$Q_I^T Q_F \;=\; Q_z(\varphi_z) Q_y(\varphi_y) Q_x(\varphi_x) \tag{4.2}$$

Ahora, con los invariantes definidos con `SetInvariants()`, es posible obtener cualquier postura intermedia T_m con el valor de η como se muestra

en la figura 4.15, llamando a la función `GetPose()`, donde el pseudocódigo 2 representa el desarrollo de la función `GetPose`(η).

Pseudocódigo 2 *GetPose()* de la clase *CLinearPath*

function GETPOSE(η)

$\mathbf{Q_m} \leftarrow Q_I Q_z(\eta\varphi_z) Q_y(\eta\varphi_y) Q_x(\eta\varphi_x)$

$\mathbf{d_m} \leftarrow \mathbf{d_I} + \eta\mathbf{d}$

$$T_m \leftarrow \begin{bmatrix} Q_m & \mathbf{d_m} \\ \mathbf{0} & 1 \end{bmatrix}$$

return T_m

end function

4.7 Seguimiento de trayectorias continuas.

Cuando se habla de trayectorias continuas, se habla de trayectorias curvilíneas o trazos de trayectos que tienen cambios en sus pendientes, ya sea poco pronunciadas o muy pronunciadas. Como se describió en la subsección 3.3.2, hay dos formas de definir una trayectoria continua, una es con una función en el espacio ya definida en forma paramétrica, donde las variables pueden ser el tiempo, un ángulo o la longitud de arco de la trayectoria total. De forma, que, en el sistema solamente es necesario establecer la orientación de la función ya definida. La segunda forma para definir una trayectoria curvilínea es por medio de múltiples puntos de apoyo, que conforman una aproximación de la curva que se desea establecer. Aunque en cada punto vecino con la clase `CLinearPath` podría recorrer cada punto de apoyo establecido de forma lineal, esto podría ser útil si no se requiere de mucha precisión y rapidez en el trayecto de la curva. Sin embargo, el objetivo principal del presente trabajo, es que con base a los puntos de apoyo establecidos el manipulador y su efector final puedan trazar una curva continua en el espacio, no de forma lineal como se vio anteriormente, sino de forma suave como en una función en el espacio tridimensional, como se muestra en la figura 4.17 donde se tienen presentes N cantidad puntos de apoyo que conforman una trayectoria curvilínea continua en un espacio cartesiano. Para esto se crearon diferentes algoritmos con el método matemático Spline [20].

La clase principal que hace todos los cálculos matemáticos para la generación y seguimiento de trayectorias continuas curvilíneas es `CParametricSpline`.

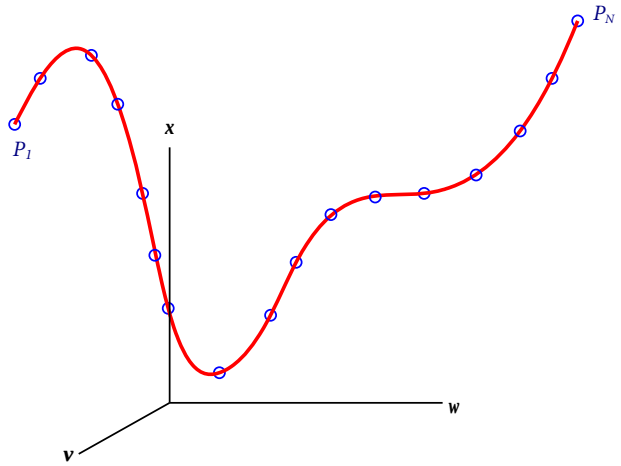


Figura 4.17: Curva spline generada con base en múltiples puntos de apoyo.

Los puntos más importantes que se deben considerar para generar una trayectoria continua en la plataforma, son los siguientes:

- Generación de polinomios cúbicos de forma paramétrica por cada conjunto de posiciones de apoyo que sean vecinos entre sí, como se muestra en la ecuación (3.84).

  $$\mathbf{p}(\sigma) = \mathbf{a_k}(\sigma-\sigma_k)^3 + \mathbf{b_k}(\sigma-\sigma_k)^2 + \mathbf{c_k}(\sigma-\sigma_k) + \mathbf{d_k}, \quad k = 1,\ldots,N-1$$

- Con los polinomios obtenidos, se obtienen poses discretas con el parámetro η.

- Su funciomaniento radica en el conocimiento de todos los puntos de apoyo P.

- Cualquier pose deseada del efector final está representada por la matriz de transformación homogénea T (ec. (4.1))

Cada una de estas consideraciones son esenciales para la generación de trayectorias continuas, y están desarrolladas en la clase `CParametricSpline`.

4.7.1 Clase CParametricSpline.

Teniendo la clase **CParametricSpline** se pueden generar trayectorias continuas con base en N cantidad de puntos de apoyo. Y la conforman cuatro funciones principales: `InitCartPoints()`, `InitSplineMatrices()`, `GetPose()` y `PathParameter()`.

La función `InitCartPoints`($pDoc, path$) está definida como:

void **InitCartPoints**(`CADEFIDDoc`* $pDoc$, `int` $path$)

La función inicializa todos los puntos de apoyo de una ruta deseada, esto es, el sistema guarda previamente N cantidad de poses, y de estas poses hay diferentes rutas. Por ejemplo, se definieron 200 poses para almacenamiento y edición, de estas poses se tomaron 7 trayectorias con 25 puntos cada una, de manera que el usuario puede seleccionar una de esas siete rutas.

Entonces, la función `InitCartPoints()` debe pasar dos argumentos importantes para su ejecución:

- Un puntero al documento $pDoc$, que guarda todos las variables y parámetros del manipulador que se esté utilizando.
- La variable entera $path$, que indica la trayectoria que se desea utilizar.

El pseudocódigo 3 representa la función `InitCartPoints`($pDoc, path$), se describe a continuación:

Como se muestra al principio de la función, hay un `swith()` que describe los posibles casos de las rutas existentes en el sistema. Como se mencionó, hay 200 poses que el usuario puede manipular, y 7 trayectos posibles para la aplicación de corte con 25 puntos de apoyo en cada trayecto, lo que hace que se tengan 175 poses disponibles para hacer las trayectorias continuas con *Spline*. Los 25 puntos restantes se utilizaron para almacenar poses arbitrarias como $Home_1$, $Home_2$, etc. La tabla 4.2, ilustra de que pose a otra pose almacenada hay puntos de apoyo con sus respectivas rutas y poses arbitrarias.

Teniendo la trayectoria deseada, se van seleccionando los puntos de apoyo de las poses existentes en la trayectoria, con $\mathbf{p_k}$, se almacenan todos los

Pseudocódigo 3 *InitCartPoints()* de la clase *CParametricSpline*

function INITCARTPOINTS(*pDoc*, *path*)
  beginSwitch (*path*)
    begincase 1:
      $TableIndex \leftarrow 5$
      break
    endcase
  ⋮
    begincase 7:
      $TableIndex \leftarrow 155$
      break
    endcase
    begincase default:
      $TableIndex \leftarrow 180$
      break
    endcase
  endSwitch
  ⋮
  for $k \leftarrow 0$ **till** $n-1$ **do**
    $\mathbf{p_k} \leftarrow$ KawasakiBX100N.PosBX100N$[k + TableIndex]$
    $\mathbf{\Delta p_k} \leftarrow \mathbf{p_{k+1}} - \mathbf{p_k}$
    $\sigma_0 \leftarrow 0$
    $\Delta\sigma_k \leftarrow \sqrt{\Delta x_k^2 + \Delta y_k^2 + \Delta z_k^2}$
    $\sigma_{k+1} \leftarrow \sigma_k + \Delta\sigma_k$
  end for
end function

puntos cartesianos de las poses y las variables $\Delta\sigma_k$ y σ_k (ec. (3.71)) se van calculando con base en todos los puntos almacenados de la trayectoria seleccionada. Para esto es necesario incluir un ciclo `for` que genere $n-1$ veces los cálculos de las variables para los siguientes cálculos matriciales. Ahora, en esta función solamente se tienen almacenados los puntos de apoyo de la trayectoria que se desea generar en forma de curva continua, sin embargo,

Tabla 4.2: Trayectorias disponibles con sus respectivas poses

Path	Pose Inicial	Pose Final	Número de Poses
Home	0	4	5
1	5	29	25
2	30	54	25
3	55	79	25
4	80	104	25
5	105	129	25
6	130	154	25
7	155	179	25
Arbitrario	180	199	20

por el momento todavía no se ha generado ninguna Spline, solo se han definido los N puntos de apoyo de la trayectoria seleccionada (*path*) y sus aproximaciones de longitud de arco de cada punto de apoyo vecino entre sí, una manera de visualizar el funcionamiento de `InitCartPoints`() se muestra en la figura 4.18.

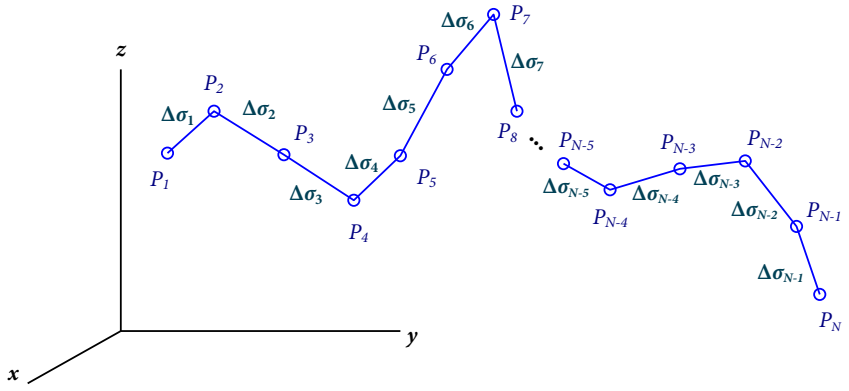


Figura 4.18: N puntos de apoyo designados para elaborar una curva spline.

Esto es, por ejemplo, si el usuario quiere que se calcule y genere la trayectoria Spline con la trayectoria 3 de la tabla 4.2, se toman los puntos

cartesianos almacenados en las poses 55 a 79, en total son 25 poses, entonces la variable $n-1$ es igual a 25. De esta manera el ciclo `for` se establecería de 0 a 24 para hacer los cálculos previos de la variables iniciales necesarias para los cálculos de las matrices que más adelante se irán calculando.

Ahora que se tienen las variables previas, se hace uso de la función `InitSplineMatrices()` para crear todas las variables y matrices de Spline para la trayectoria seleccionada.

La diferencia de esta función es que no hay argumentos que reciba directamente de otras funciones, ni algún valor o dato que se deba regresar. Sin embargo, en la ejecución de dicha función se van haciendo los cálculos previos y modificaciones necesarias en las operaciones para la función Spline, que más adelante serán de suma importancia.

El pseudocódigo 4 que representa la función `InitSplineMatrices`(*void*), se describe a continuación:

La función inicia con un ciclo `for` de 0 a $< n-1$ de alguna trayectoria seleccionada (en este caso, como todas son de 25 puntos, $n-1=25$), y calcula los parámetros α_k y β_k de los puntos de apoyo, estas dos variables son importantes para ir creando la matriz A y la matriz C de las ecuaciones (3.99) y (3.100). Ya que son trayectorias abiertas las 7 trayectorias posibles, son de tipo Splines naturales. Entonces la matriz que presenta mayor conflicto y que define la problemática de las Splines es P$''$, como se muestra en la ecuación (3.98) y para los fines de este capítulo, se ilustra con más información en la ecuación (4.3); primero se necesita hacer que la matriz P tenga todos los puntos de apoyo, esto es, que sea de $N\times 3$; después invertir A y denominarla M_1, se debe tener cuidado en esta operación ya que A debe ser cuadrada para que se pueda invertir; la siguiente operación es obtener M_2 como se muestra en la ecuación (4.4); posteriormente obtener M_3 de acuerdo a la ecuación (4.5); y finalmente la última operación matricial M_4 es el producto M_3P, conforme a la ecuación (4.6).

$$\overbrace{\mathrm{P}''}^{N\times 3} = \underbrace{6}_{Escalar}\underbrace{\overbrace{\mathrm{A}^{-1}}^{(N-2)\times(N-2)}}_{M_1}\underbrace{\mathrm{C}}_{(N-2)\times N}\overbrace{\mathrm{P}}^{N\times 3} \tag{4.3}$$

Pseudocódigo 4 *InitSplineMatrices()* de la clase *CParametricSpline*

function INITSPLINEMATRICES(*void*)

for $k \leftarrow 0$ **till** $n-1$ **do**

$$
\begin{aligned}
\alpha_k &\leftarrow \Delta\sigma_k \\
\beta_k &\leftarrow 1/\alpha_k \\
\mathbf{A} &\leftarrow [MatrixA] \\
\mathbf{C} &\leftarrow [MatrixC] \\
\mathbf{p}_k &\leftarrow \begin{bmatrix} x_k & y_k & z_k \end{bmatrix} \\
\mathbf{P''} &= 6\mathbf{A}^{-1}\mathbf{CP} \\
M_1 &\leftarrow \text{InverseMatrix}(A) \\
M_2 &\leftarrow \text{ScalarProductMatx}(6, M_1) \\
M_3 &\leftarrow \text{MatrixProduct}(M_2, C) \\
M_4 &\leftarrow \text{MatrixProduct}(M_3, P) \\
\mathbf{P''} &\leftarrow M_4 \\
\mathbf{p}''_k &\leftarrow \begin{bmatrix} x''_k & y''_k & z''_k \end{bmatrix} \\
\mathbf{a_k} &\leftarrow \frac{1}{6\Delta\sigma_k}\left(\mathbf{p}''_{k+1} - \mathbf{p}''_k\right) \\
\mathbf{b_k} &\leftarrow \frac{1}{2}\mathbf{p}''_k \\
\mathbf{c_k} &\leftarrow \frac{\Delta\mathbf{p}_k}{\Delta\sigma_k} - \frac{1}{6}\Delta\sigma_k\left(\mathbf{p}''_{k+1} + 2\mathbf{p}''_k\right) \\
\mathbf{d_k} &\leftarrow \mathbf{p}_k
\end{aligned}
$$

end for

end function

$$
\overbrace{\mathrm{P''}}^{N\times 3} = \underbrace{\overbrace{\begin{matrix} 6 & \mathrm{A}^{-1} \end{matrix}}^{M_2}}_{(N-2)\times(N-2)} \underbrace{\mathrm{C}}_{(N-2)\times N} \overbrace{\mathrm{P}}^{N\times 3} \tag{4.4}
$$

$$
\overbrace{\mathrm{P''}}^{N\times 3} = \underbrace{\overbrace{\begin{matrix} 6 & \mathrm{A}^{-1} & \mathrm{C} \end{matrix}}^{M_3}}_{(N-2)\times N} \overbrace{\mathrm{P}}^{N\times 3} \tag{4.5}
$$

$$\overbrace{\mathrm{P}''}^{N\times 3} = \overbrace{\underbrace{6 \quad \mathrm{A}^{-1} \quad \mathrm{C} \quad \mathrm{P}}_{(N-2)\times 3}}^{M_4} \tag{4.6}$$

La forma en la que se presentan las ecuaciones (4.1) - (4.6) muestra la dimensión de las matrices resultantes, y se ve claramente la diferencia existente en la ecuación (4.6). La manera en la que se describió en la subsección 3.3.2, una Spline natural tiene dos condiciones importantes, de acuerdo a la ecuación (3.94), la matriz M_4 tiene el mismo tamaño de la matriz P'', esto es, $M_4 \rightarrow (N-2+2)\times(3) \equiv \mathrm{P}'' \rightarrow (N\times 3)$.

Por lo que con la ecuación (4.7), la dimensión de la matriz resultante en el lado derecho de la ecuación (4.6) cambia conforme se muestra en la ecuación (4.8).

$$\mathbf{p}_1'' = \mathbf{p}_N'' = \mathbf{0} \tag{4.7}$$

$$\overbrace{\mathrm{P}''}^{N\times 3} = \overbrace{\underbrace{6 \quad \mathrm{A}^{-1} \quad \mathrm{C} \quad \mathrm{P}}_{N\times 3}}^{M_4\rightarrow(N-2+2)\times 3} \tag{4.8}$$

Ahora que se conoce la matriz P'', que representa los puntos de apoyo de la segunda derivada de la curva que se desea construir, se pueden calcular los coeficientes $\mathbf{a}_k$, $\mathbf{b}_k$, $\mathbf{c}_k$ y $\mathbf{d}_k$ que son los que definen la trayectoria curvilínea paramétrica de todos los puntos de apoyo inicialmente definidos.

La última función llamada `GetPos()`, es muy similar a la función `GetPos()` de la clase `CLinearPath`, la diferencia es que en la clase `CParametricSpline` se crean los polinomios cúbicos en forma paramétrica en función de las aproximaciones longitudinales de arco totales de la trayectoria planeada. La función está definida como:

Matrix **GetPos**(`float` *mParam*, `int` *path*);

De manera que debe pasar dos argumentos importantes para su ejecución:

- La variable double *mParam*, que ingresa valores discretizados desde 0 a 1 para el control y aproximación de poses intermedias en la trayectoria deseada.

- La variable entera *path*, que indica la trayectoria que se desea utilizar.

La función `GetPos()` regresa la matriz de transformación homogénea, que tiene los datos para conocer la matriz de orientación y el vector de posición del efector final en determinado instante de la trayectoria que se esté trazando.

El pseudocódigo 5 muestra como está estructurada la función `GetPos`() de la clase `CParametricSpline`.

Pseudocódigo 5 *GetPos()* de la clase *CParametricSpline*

function GETPOS(*mParam*, *path*)

InitCartPoints (*pDoc*, *path*)

$\vdots$

$\sigma \leftarrow mParam * \sigma_{n-1}$

$k \leftarrow \text{PathParameter}(\sigma)$

$\sigma_p \leftarrow \sigma - \sigma_k$

$\vdots$

$\mathbf{x} \leftarrow \mathbf{a_{xk}}(\sigma_p)^3 + \mathbf{b_{xk}}(\sigma_p)^2 + \mathbf{c_{xk}}(\sigma_p) + \mathbf{d_{xk}}$

$\mathbf{y} \leftarrow \mathbf{a_{yk}}(\sigma_p)^3 + \mathbf{b_{yk}}(\sigma_p)^2 + \mathbf{c_{yk}}(\sigma_p) + \mathbf{d_{yk}}$

$\mathbf{z} \leftarrow \mathbf{a_{zk}}(\sigma_p)^3 + \mathbf{b_{zk}}(\sigma_p)^2 + \mathbf{c_{zk}}(\sigma_p) + \mathbf{d_{zk}}$

$\vdots$

$\mathbf{d_m} \leftarrow x, y, z$

$Q_m \leftarrow definida$

$$T_m \leftarrow \begin{bmatrix} Q_m & \mathbf{d_m} \\ \mathbf{0} & 1 \end{bmatrix}$$

return T_m

end function

Como se muestra en el pseudocódigo 5 la función `GetPos`(*mParam*, *path*) inicia llamando la función `InitCartPoints`(*pDoc*, *path*) para inicializar los puntos de apoyo en la trayectoria (*path*) seleccionada, y a su

vez esta función llama a la otra función `InitSplineMatrices`($pDoc$, $path$), para crear todas las operaciones matriciales necesarias. Teniendo todas las variables y matrices necesarias Splines, es momento de crear los polinomios cúbicos que se aproximan a una curva continua en el espacio con respecto a todos los puntos de apoyo existentes. Es así, que ahora la parametrización de la curva ya no va de 0 a 1 como se vio en la clase `ClinearPath`, ahora la parametrización de la curva va del valor mínimo de sigma ($\sigma_1 = 0$) hasta el valor máximo de σ ($\sigma_{n-1} = \sigma_{max}$). Dependiendo del valor que el sistema arroje para σ, es posible crear el trozo de curva en función de un polinomio cúbico en forma paramétrica, esto es los valores x, y, z que ilustra el pseudocódigo 5. Estos valores son el vector $\mathbf{d}$, que representa el vector de posición en la matriz de transformación homogénea T y la matriz de orientación Q_m en la trayectoria es tomada con base en las poses de apoyo que fueron guardadas en el algoritmo 3. Y finalmente se obtiene la trama intermedia T_m en la cual el efector final del manipulador debe posicionarse y/o moverse. Ahora, como se explicó al principio de esta función, el parámetro $mParam$, también conocido como η, que toma valores discretizados de 0 a 1, va introduciendo esos valores a la función `GetPose()` para obtener las diversas poses intermedias de la trayectoria continua en Spline, sin embargo, el que controla y distribuye los polinomios cúbicos es σ, parametrizado de 0 a σ_{max}.

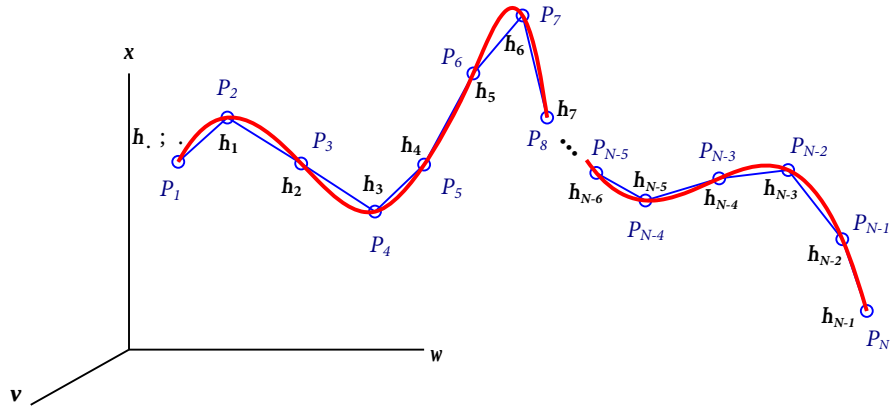


Figura 4.19: Curva paramétrica cúbica spline generada con N puntos de apoyo.

Como se muestra en el pseudocódigo 5, la función `PathParameter`(σ) es llamada mientras `GetPose`($mParam$, $path$) está corriendo. La figura 4.19 muestra una curva en un espacio tridimensional, conformada por N puntos de apoyo y la curva cúbica generada por Spline. La función está definida como se muestra a continuación:

int **PathParameter**(`float` m_sigma)

La función `PathParameter`(m_sigma) (pseudocódigo 6), solo recibe un parámetro, m_sigma (σ) es una variable tipo double que está parametrizada de 0 a σ_{max}, como se ha estado mencionando, σ es una aproximación de longitud de arco de la trayectoria conformada por todos los puntos de apoyo. La función al final regresa un valor entero k, el cual es muy importante para la función `GetPose`($mParam$, $path$), ya que con esta variable se puede reconocer en qué sección de la curva va a generarse el trozo de polinomio cúbico entre dos puntos de apoyo vecinos entre si, y es fundamental conocer la variable k de manera que, es posible seleccionar los coeficientes $\mathbf{a}_k$, $\mathbf{b}_k$, $\mathbf{c}_k$ y $\mathbf{d}_k$ de la curva creada por Spline.

Pseudocódigo 6 *PathParameter()* de la clase *CParametricSpline*

```
function PathParameter(σ)
    for k ← 0 till n − 1 do
            if(σ ≥ σ_k && σ < σ_{k+1}) then
                return k
            end if
    end for
        return −1
end function
```

Un ejemplo gráfico del funcionamiento de `PathParameter`() se ilustra en la figura 4.21a, en la imagen se muestran los puntos de apoyo los cuales son entradas importantes para crear la curva paramétrica Spline, de manera que hay 10 aproximaciones de longitudes de arco ($\Delta\sigma_i$), que van de punto en punto con una longitud semejante (idealmente). Y que la aproximación total de las longitudes de arco es 100 mm, este valor es entonces σ_{max}. Sin

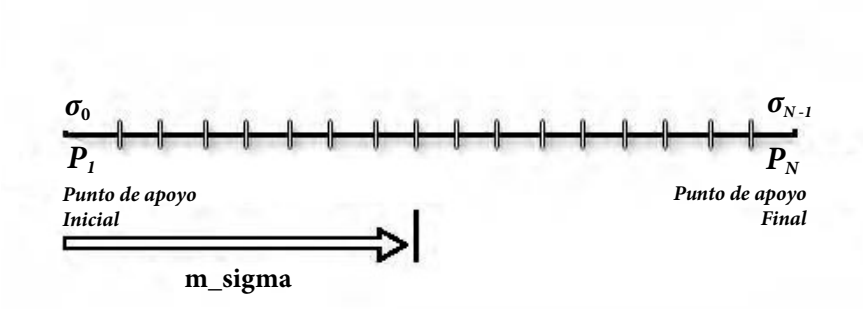
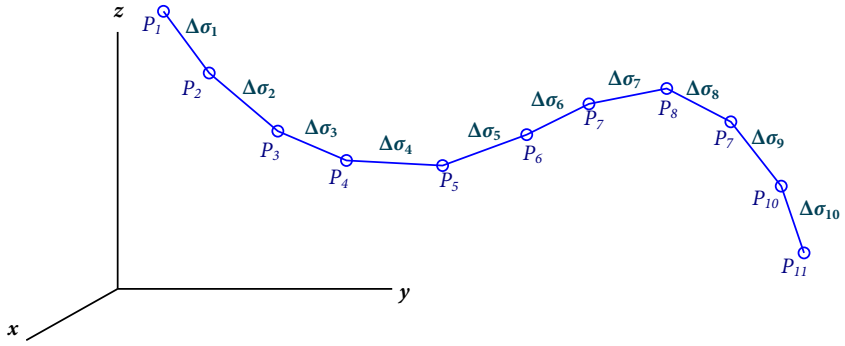


Figura 4.20: Parámetro m_sigma para la función PathParameter().

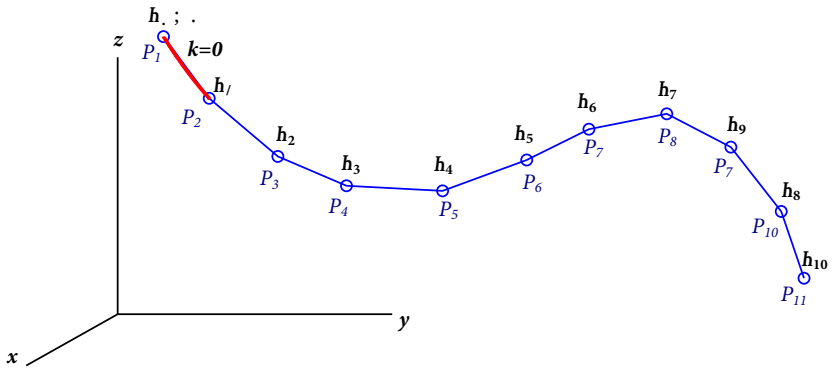
embargo, como en las explicaciones previas, la función solamente puede crear un trozo de polinomio cubico por cada conjunto de puntos que sean vecinos entre sí. De manera que al ser llamada la función `GetPose`$(mParam, path)$, $mParam$ toma valores discretizados de 0 a 1.

En un caso particular, $mParam$ ahora es 0.35, entonces este valor entra a la función `GetPose`() y σ ahora es $mParam * \sigma_{max} = 0.35 * (100) = 35$. Y la función `PathParameter`() recibe el valor $\sigma = 35$ y si se tiene que cada longitud de arco mide igual a las otras, cada $\Delta\sigma_i$ equivale a 10 mm. Y $\sigma_0 = 0$, $\sigma_1 = 10$, $\sigma_2 = 20$, $\sigma_3 = 30$, $\sigma_4 = 40, \ldots, \sigma_{10} \equiv \sigma_{max} = 100$. De tal forma que la función `PathParameter`(σ) va ir comparando el valor de sigma con cada σ_i. Esto es, se compara $\sigma = 35$ es mayor a $\sigma_0 = 0$ y menor a $\sigma_1 = 10$ (ver Fig. 4.21b); de manera que no lo es y se incrementa el valor de k a 1; ahora compara, $\sigma = 35$ es mayor a $\sigma_1 = 10$ y menor a $\sigma_2 = 20$ (ver Fig. 4.21c); de manera que no lo es y se incrementa el valor de k a 2; ahora compara, $\sigma = 35$ es mayor a $\sigma_2 = 20$ y menor a $\sigma_3 = 30$ (ver Fig. 4.22a); de manera que tampoco lo es y se incrementa el valor de k a 3; ahora compara, $\sigma = 35$ es mayor a $\sigma_3 = 30$ y menor a $\sigma_4 = 40$, donde ahora este caso si se cumple (ver Fig. 4.22b), y $k = 3$ y regresa el valor a la función `GetPose`$(mParam, path)$, para hacer el polinomio cúbico entre los puntos que contenga σ_3 y σ_4.

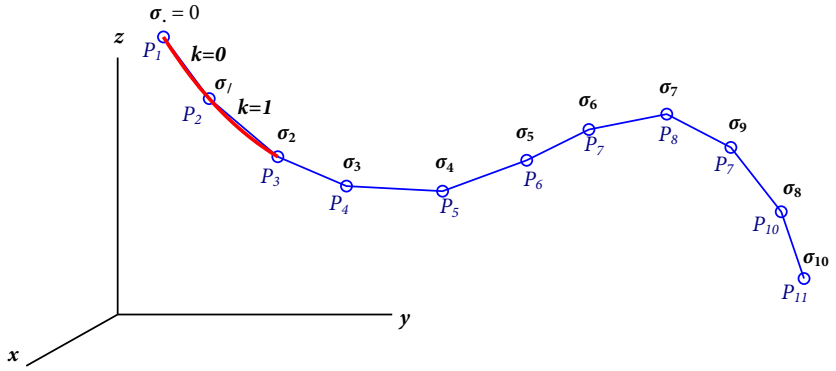
Finalmente, una vez que el parámetro sigma haya recorrido todos los σ_k, desde σ_0 hasta σ_{N-1} que equivale al σ_{max}, la curva Spline es generada con todas las matrices y variables que se definieron con los puntos de apoyo P_N, como se muestra en la figura 4.22c.



(a) Puntos de apoyo para la creación de una curva spline.

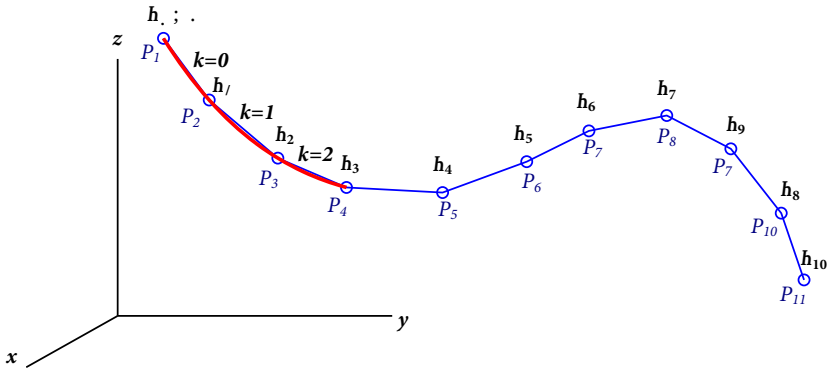


(b) Curva spline con 1 polinomio creado, k=0.

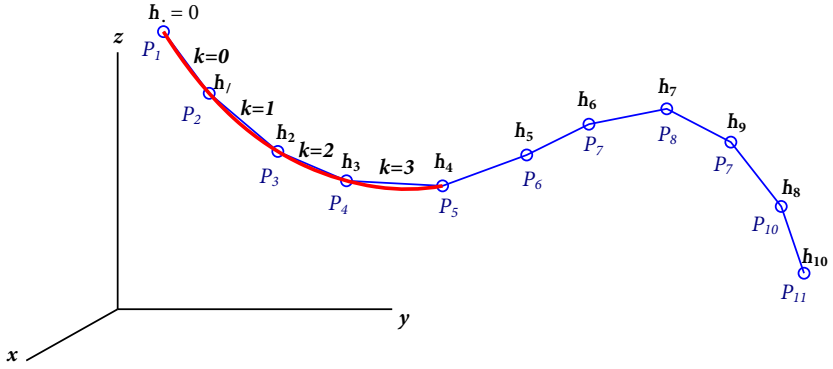


(c) Curva spline con 2 polinomios creados, k=1.

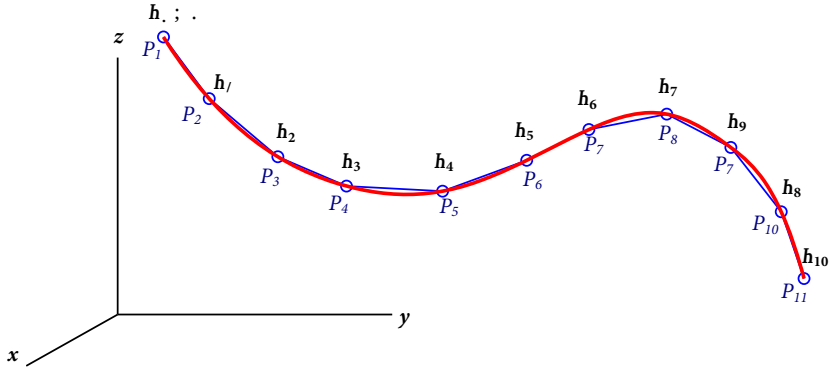
Figura 4.21: Creación de curva spline con 11 puntos de apoyo.



(a) Curva spline con 3 polinomios creados, k=2.



(b) Curva spline con 4 polinomios creados, k=3.

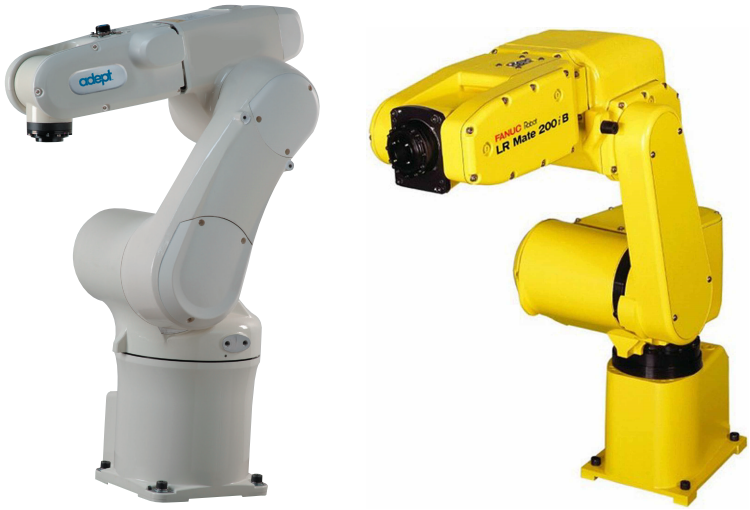


(c) Curva spline con 11 puntos de apoyo.

Figura 4.22: Curva spline creada con base en las funciones GetPos() y PathParameter().

4.8 Diseño del manipulador en ADEFID.

En cuanto a la parte de la simulación de manipuladores robóticos en `ADRS`, se tomaron como patrón las funciones que se utilizaron para dos manipuladores ya existentes en la biblioteca de `ADRS`, tal es el caso de los modelos Adept Viper S650 y Fanuc LR Mate 200iB mostrados en la figura 4.23, para implementar e integrar el manipulador industrial Kawasaki BX100N (ver Fig. 4.24), que se escogió para la aplicación de corte en un automóvil.



(a) Adept Viper S650 [29]. (b) Fanuc LR Mate 200iB [30].

Figura 4.23: Modelos utilizados en la plataforma ADEFID.

Para cada manipulador se crearon clases específicas, donde se incluyen funciones para el modelo de cada eslabón por separado, así como los parámetros Denavit-Hartenberg (DH) del mismo.

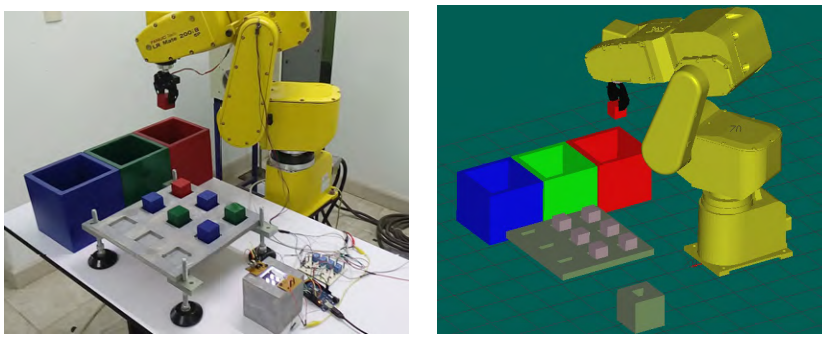
Cada vez que se desea modelar un nuevo manipulador en `ADRS`, es necesario asignar los parámetros DH correspondientes a las variables globales que se encargan de ensamblar al manipulador. Tal función conocida como `SetDefaults()`, es la que contiene los parámetros DH, el tipo de manipulador que lo conforma, la cantidad de eslabones, la longitud máxima posible de alcance, entre otros parámetros más y asigna sus valores a la cadena cinemática del manipulador, donde se va formando su estructura en forma de eslabones articulados, dándole forma al robot con base en sus parámetros,

Figura 4.24: Robot Kawasaki BX100N [31].

de tal manera, que se puede visualizar un esqueleto del mismo robot. Este diseño se utiliza posteriormente para hacer los movimientos deseados en la trayectoria como rotaciones y traslaciones en la simulación dentro de la plataforma `ADEFID`.

Con respecto al concepto de la simulación mixta, se han venido desarrollando proyectos de investigación, tal es el caso del trabajo de Morales-González [32], que trabajó con el manipulador articulado Fanuc LR Mate 200iB. Dicho concepto se aplica en este trabajo y consiste en generar un ambiente gráfico con todos los dispositivos que intervienen en el proceso de simulación y al mismo tiempo conectarse físicamente con uno o varios dispositivos para controlar y/o monitorear su respuesta. La figura 4.25 presenta evidencias en las que se ilustran el manejo del prototipo virtual y real del manipulador trabajando coordinadamente.

Para darle una visualización o aspecto más real a la aplicación del proceso, la plataforma `ADEFID` tiene la capacidad de procesar archivos CAD, como se muestra en la figura 4.25b. Para esto es necesario generar los diseños en algún software de diseño, en este caso se utilizó SolidWorks [33], de tal

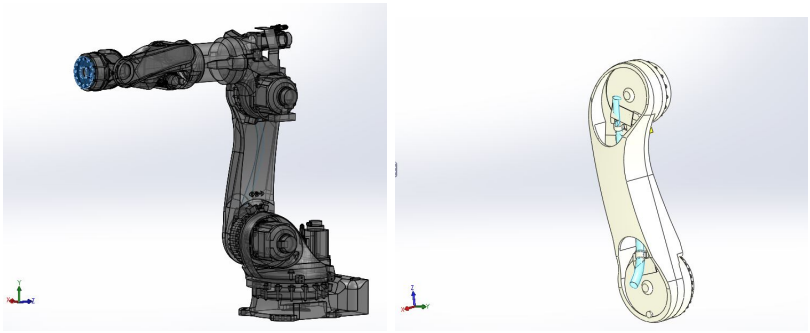


(a) Robot Fanuc real. (b) Robot Fanuc virtual.

Figura 4.25: Simulación del robot Fanuc LR Mate 200iB en ADEFID [32].

manera que se generen diseños que la plataforma `ADEFID` puede procesar, a partir de hacer lecturas de archivos STL, sin embargo, su proceso requiere una serie de pasos que se deben considerar, y que a continuación se describen:

Primer Paso. Teniendo los diseños o diseño único del manipulador (Fig. 4.26a), es importante separar cada pieza que conforma al robot, esto es, que si en el diseño se tiene un solo archivo 3D es necesario separar cada una de sus partes que conforman sus grados de libertad como se muestra en la figura 4.26b, ya que en `ADEFID`, cada eslabón tiene su movimiento en el espacio.

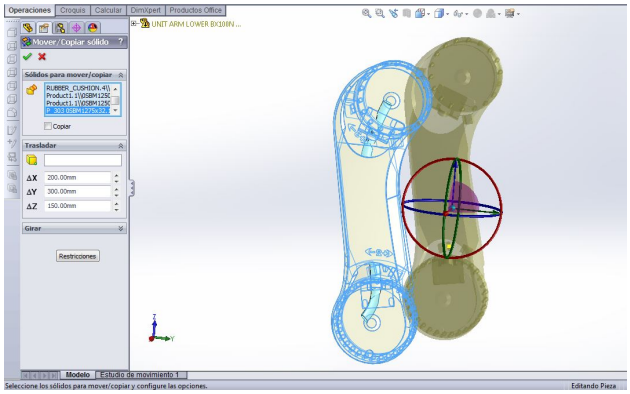


(a) CAD Robot Kawasaki. (b) CAD Eslabón 2 del robot.

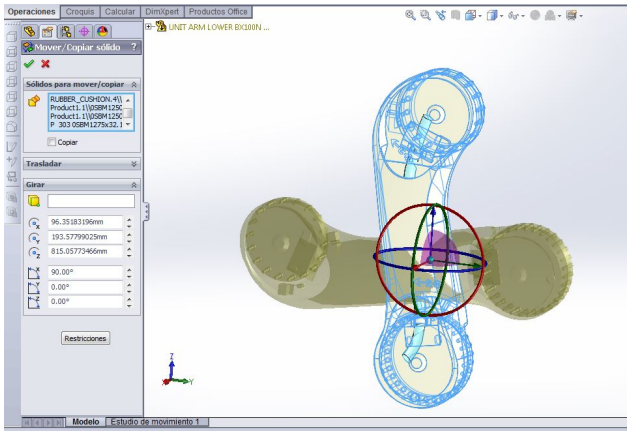
Figura 4.26: CAD del robot y sus eslabones.

Segundo Paso. Con las piezas debidamente desacopladas del diseño inicial, se deben hacer ciertas transformaciones para poder ingresar las piezas

a la plataforma. Cuando se exporta el objeto desde el software CAD, si éste no está en el primer octante, el CAD lo posiciona automáticamente y se pierde la información sobre el sistema de referencia, por lo que es recomendable que el usuario manipule la posición de la pieza para que se tenga el conocimiento de las coordenadas de traslación.



(a) Traslación de la pieza.



(b) Rotación de la pieza.

Figura 4.27: Operaciones de las piezas del robot en SolidWorks.

Tercer Paso. Ahora que las piezas del manipulador ya están en el espacio del octante positivo (Fig. 4.28), por recomendación, es necesario cambiar su orientación en el mismo software CAD. Esto es debido a que al momento de hacer el ensamble de las piezas de acuerdo a su cadena cinemática, va a ser más sencillo su acoplamiento final, correspondiendo al sistema de referencia planteado por los parámetros DH en un cuadro de

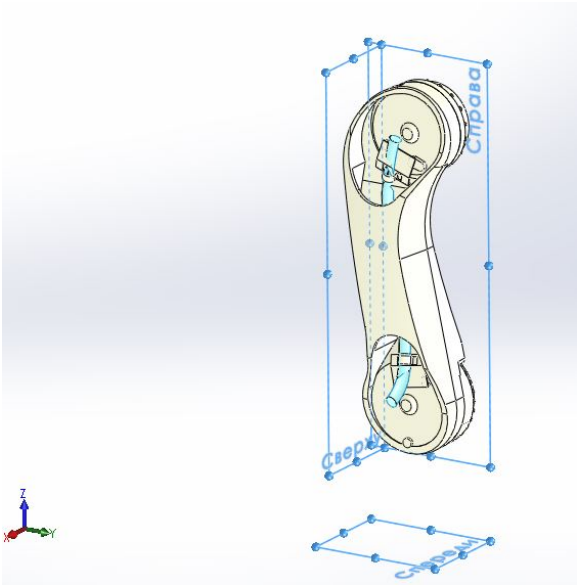
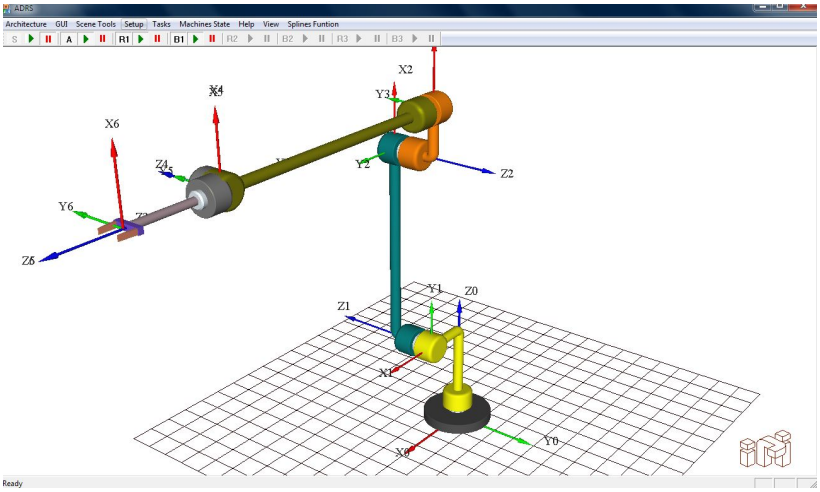


Figura 4.28: Pieza del eslabón 2 ubicada en el octante positivo.

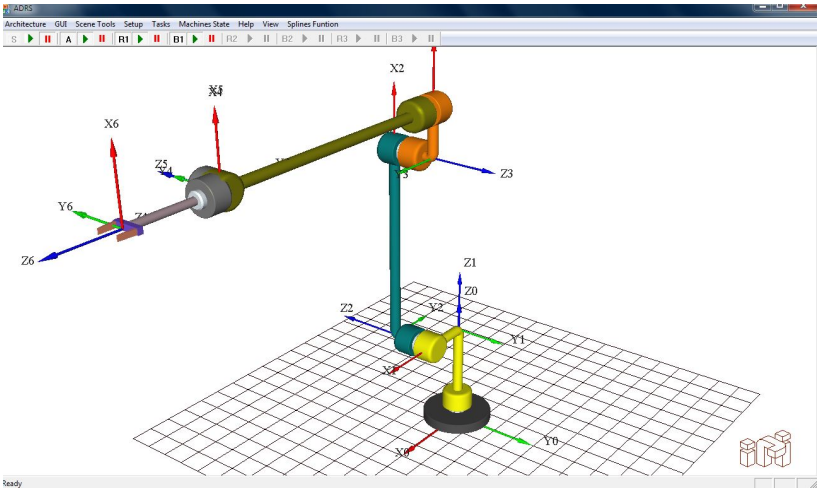
coordenadas Distal, como lo muestra la Fig. 4.29a. Y para poder hacer el ensamble en la plataforma se debe tomar en cuenta el sistema de coordenadas Proximal del esqueleto del manipulador (ver Fig. 4.29b), con respecto a la tabla DH.

Cuarto Paso. Los archivos deben ser guardados en formato STL (ver Fig.4.30), seleccionando ASCII y en milímetros como unidad de medición.

Quinto Paso. Con los debidos cambios y archivos generados, es posible agregarlos a la plataforma `ADEFID` y poder hacer la importación de tales archivos. Con la función `ReadSTLFiles` de la clase del manipulador que se esté usando, es posible hacer el llamado de las piezas 3D a la plataforma, pero es importante tener la referencia del sistema de coordenadas en la cadena cinemática del manipulador de acuerdo a la tabla de parámetros DH. Un ejemplo, de acuerdo a la figura 4.31, en la cual se hicieron los cambios de origen al espacio del primer octante positivo y su debido cambio de orientación a la pieza del eslabón 4 de las figuras 4.31a y 4.31b. De acuerdo con el sistema Proximal de la cadena cinemática, la distancia del punto de origen (al nuevo punto de origen) al origen Proximal del cuarto eslabón, X está desplazado 210 mm, a su vez Y está desplazado 1300 mm



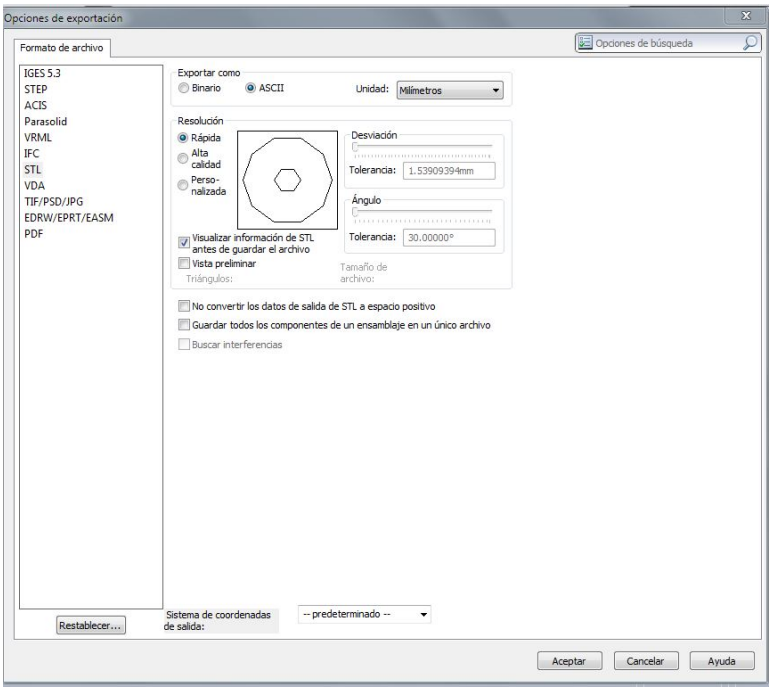
(a) Sistema de coordenadas distal.



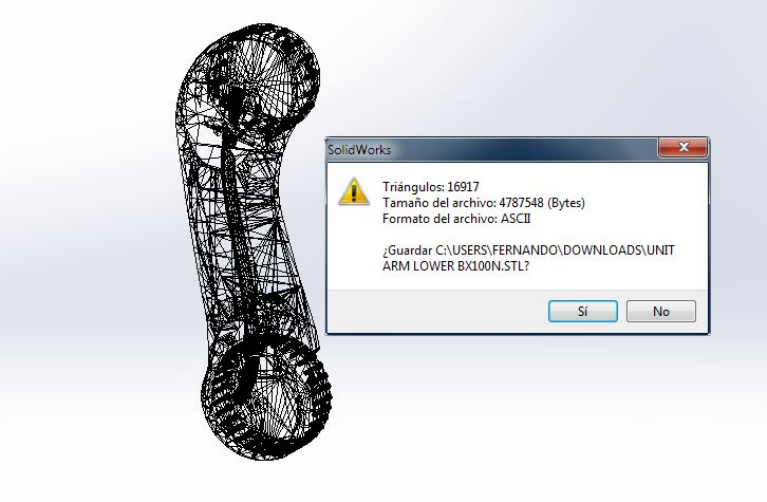
(b) Sistema de coordenadas proximal.

Figura 4.29: Sistema de coordenadas distal y proximal del manipulador.

y Z está desplazado 1555 mm. Cuando se hicieron todos los cambios en el desplazamiento de origen anterior al octante positivo, es útil saber cuánto se desplazó del antiguo sistema de referencia, ya que va ser necesario al momento de posicionar su estructura en el acoplamiento final. Y finalmente para la pieza del eslabón 4 quedaría acomodada en la cadena cinemática como se muestra en la figura 4.32.

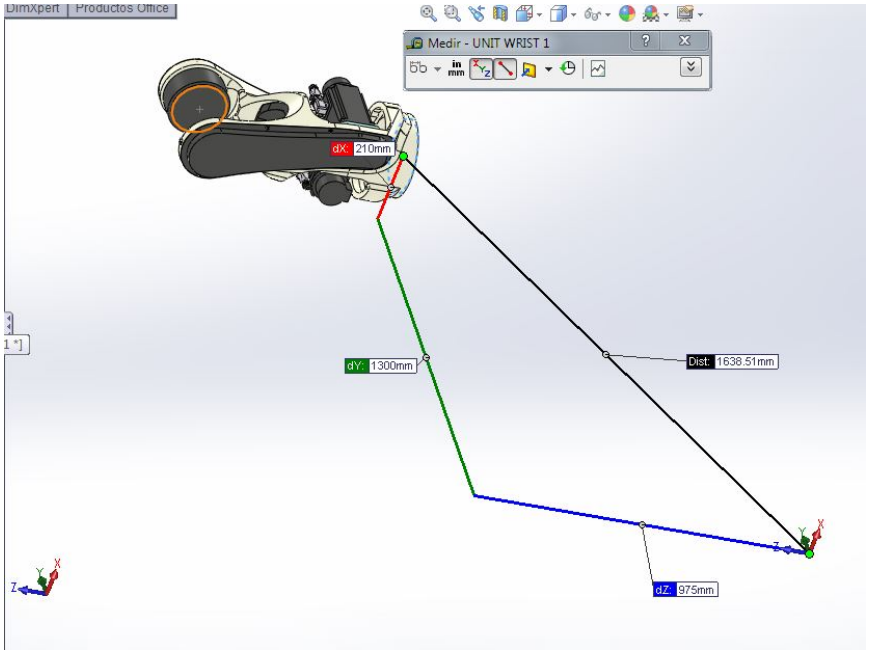


(a) Opciones de exportación de las piezas a STL.

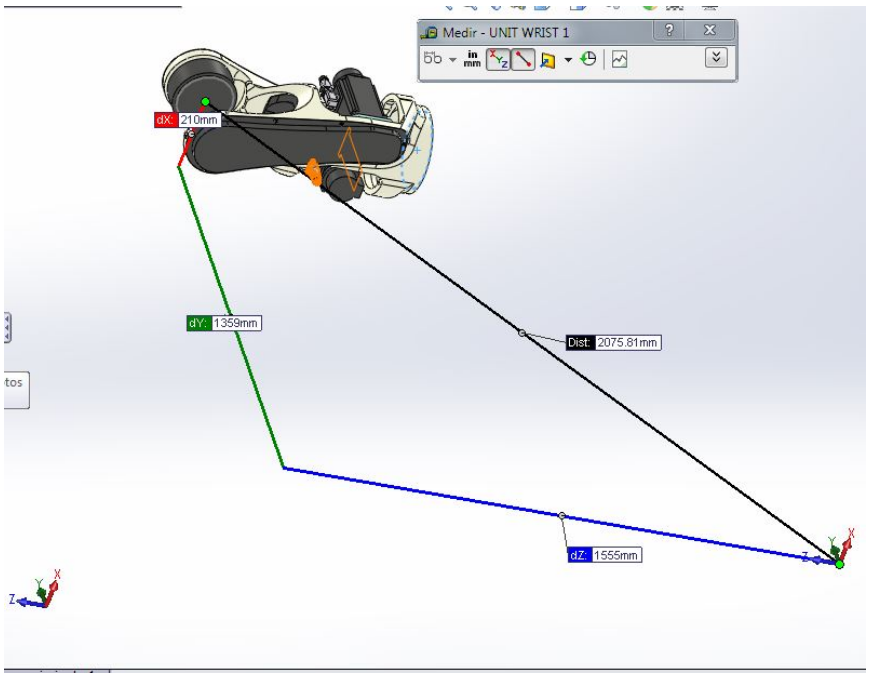


(b) Pieza del segundo eslabón en formato STL.

Figura 4.30: Formato STL.

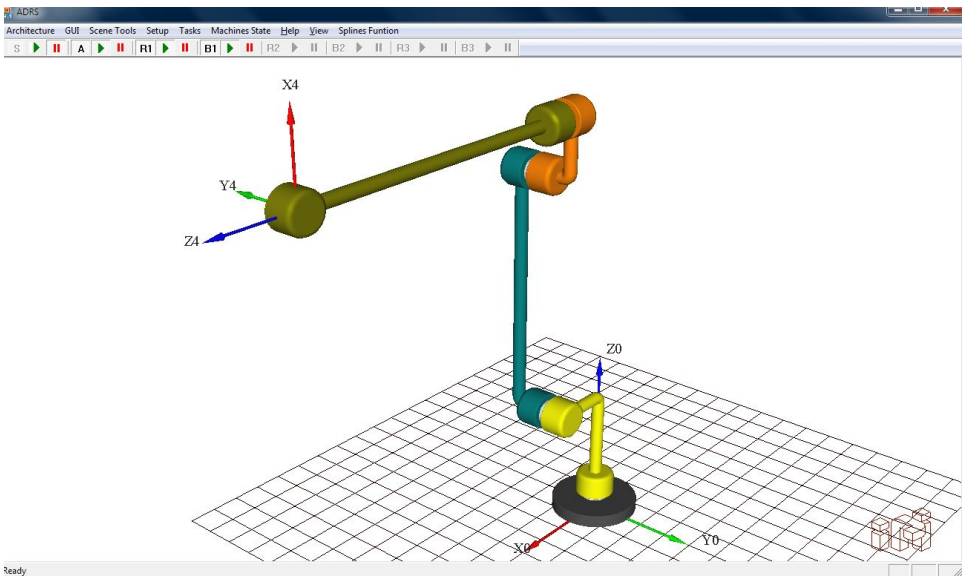


(a) Primera medición del 4° eslabón Kawasaki BX100N.

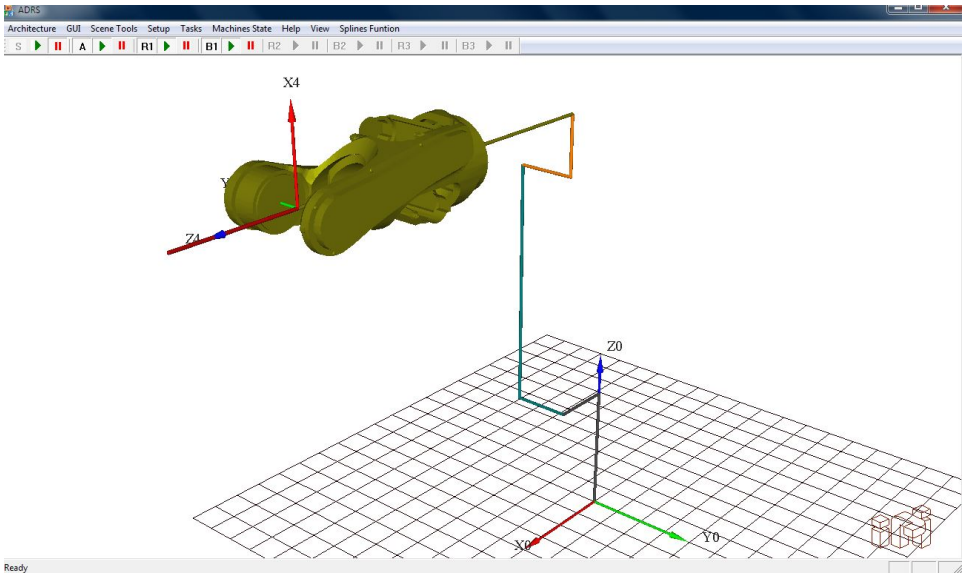


(b) Segunda medición del 4° eslabón Kawasaki BX100N.

Figura 4.31: Mediciones en los desplazamientos del eslabón 4° del robot.



(a) Referencia Proximal del 4° eslabón.



(b) CAD del 4° eslabón en la cadena cinemática del robot.

Figura 4.32: Implementación del 4° eslabón del robot en ADEFID.

Finalmente, una vez realizado lo anterior con todos los eslabones del manipulador en la clase del manipulador, el acoplamiento queda debidamente ensamblado, de manera que tendría una visualización con un aspecto que parece más real, tal como se ve en la figura 4.33.

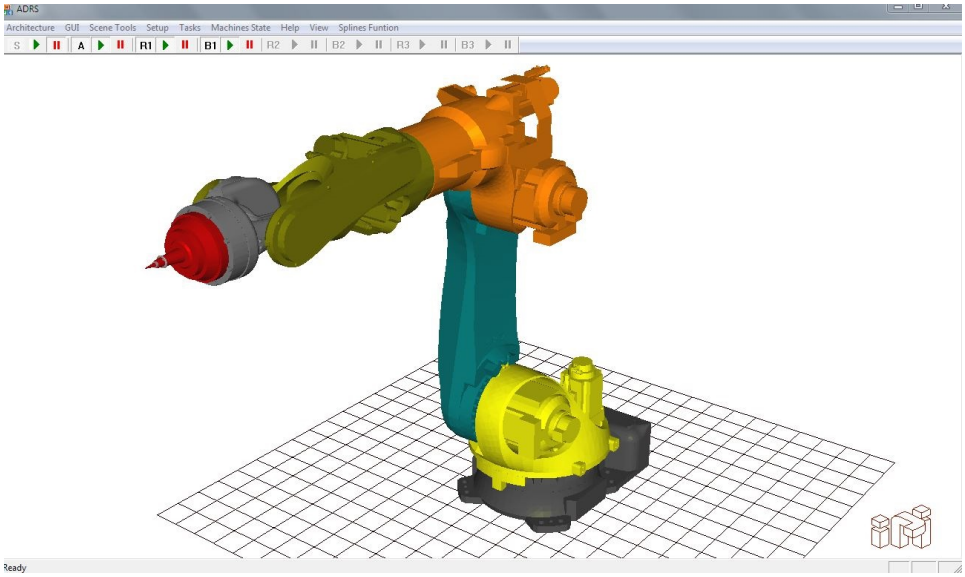


Figura 4.33: Manipulador Kawasaki en la plataforma ADEFID y en el programa ADRS.

En este capítulo se introdujo la nueva clase de trayectorias curvilíneas conocida como `CParametricSplines`, clase que es clave para que el manipulador trace cualquier trayectoria curvilínea en el espacio tridimensional utilizando N puntos de apoyo, gracias a las funciones Splines. El siguiente capítulo presenta los resultados obtenidos de la planeación de la clase anteriormente mencionada en el entorno virtual de `ADEFID`.

Capítulo 5

Resultados

Contenido

Este capítulo tiene como objetivo dar a conocer el funcionamiento del algoritmo de control del movimiento y simulación del robot Kawasaki en la plataforma `ADEFID`. En forma más específica, se presentan los resultados obtenidos de la creación de las diversas funciones y clases de las Splines.

5.1 Clase CMKawasakiBX100N.

De acuerdo a las descripciones que se dieron en el capítulo anterior para incluir un manipulador industrial a la plataforma `ADEFID`, se creó la clase del manipulador Kawasaki BX100N, con la estructura cinemática del robot basada en las especificaciones del fabricante [34], y con las variables DH del manipulador. Como primer paso, se visualiza el esqueleto y su cadena cinemática como se muestra en la figura 5.1.

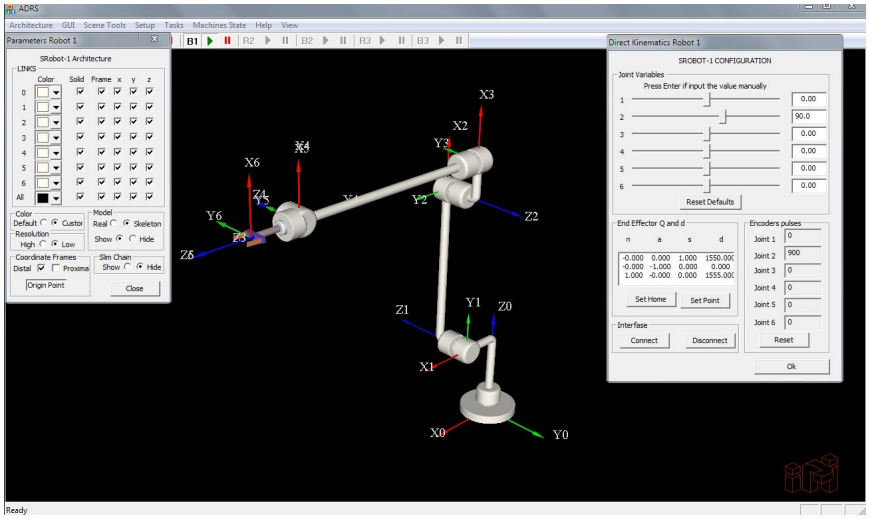


Figura 5.1: Visualización del esqueleto del manipulador Kawasaki en la plataforma ADEFID.

El cálculo de la cinemática inversa conlleva encontrar la relación que existe entre las articulaciones del manipulador con la posición y la orientación del efector final [35]. En la tabla 5.1 se muestran los parámetros Denavit-Hartenberg del manipulador Kawasaki BX100N incluyendo la herramienta de corte en el efector final.

La clase `CMKawasakiBX100N` consta de 4 secciones importantes que de-

Tabla 5.1: Parámetros DH de Kawasaki BX100N con la herramienta de corte en el EF

Eslabón	θ_i	a_i	b_i	α_i
1	θ_1^*	200	465	$\pi/2$
2	θ_2^*	880	187	π
3	θ_3^*	210	187	$-\pi/2$
4	θ_4^*	0	1100	$-\pi/2$
5	θ_5^*	0	0	$\pi/2$
6	θ_6^*	0	385	0

finen los parámetros y tareas del manipulador en la plataforma `ADEFID`. Cada uno tiene una o varias funciones que son esenciales para su funcionamiento. Las secciones que son parte de esta clase son: Parámetros iniciales del manipulador, generación visual del robot (Render GUI), generación del listado de poses de apoyo y finalmente el algoritmo de simulación del robot.

5.1.1 Inicialización de parámetros del manipulador.

La función `SetDefaults`(), Fig. 5.2, es parte importante en la visualización de la estructura inicial del manipulador, ya que ésta contiene todos los parámetros DH, más aparte algunas variables secundarias para el manipulador. Sin embargo, es importante definir todos estos parámetros en la clase del manipulador ya que son los que definen la estructura cinemática del robot, su área de trabajo, su máximo alcance, la cantidad de grados de libertad que lo conforman y el tipo de manipulador serial que lo forma.

Esta función también es conocida como la que arranca o inicializa todas las variables del manipulador, y cuando se abre el proyecto `ADEFID`, ésta se ejecuta al principio para actualizar todos los parámetros del robot, en este caso del Kawasaki.

```
BOOL CCMKawasakiBX100N::SetDefaults(void)
{
    TimerGripper.m_period = m_timeGripper;
    m_gripperType = GRIPPERLINEAR;
    BOOL result=TRUE;
    if(!CSerialRobot::SetDefaults())
        result = FALSE;

    SetManipType(ARTICULATED);
    Dh4.joints = 6;
    for(int i = 0 ; i<Dh4.joints; i++)
        Link[i].IsRevolute = TRUE;
    m_ScaleFactor = 0.12;
    m_MaxLength = 2200;
    m_quadrantSize = 1000;
    m_gridSize = 100;

    Link[0].alpha=PI/2;
    Link[1].alpha=PI;
    Link[2].alpha=-PI/2;
    Link[3].alpha=-PI/2;
    Link[4].alpha=PI/2;
    Link[5].alpha=0.;

    Link[0].m_a=200;
    Link[1].m_a=880;
    Link[2].m_a=210;
    Link[3].m_a=0;
    Link[4].m_a=0;
    Link[4].m_sphRad=50;
    Link[5].m_a=0.;
```

(a) Primera parte del código.

```
    Link[0].m_b=465;
    Link[1].m_b=187;
    Link[2].m_b=187;
    Link[3].m_b=1100;
    Link[4].m_b=0;
    Link[5].m_b=225+160;

    m_Poses.m_CurrentName=DataMainPath+
        m_Name+_T("\\Poses_Kawasaki.txt");
    m_Poses.m_NParam=4;
    if(!m_Poses.LoadData())
    {
        AfxMessageBox("No se encuentra el archivo
            "y se generarán matrices I_4");
        InitPoses();
    }
    UpdatePoses();
    // Write here
    // code to set HD parameters of Custom Manip.

    UpdateDHParam(&Dh4);
    SaveGeometryValues();
    return TRUE;
}
```

(b) Segunda parte del código.

Figura 5.2: Función SetDefaults().

5.1.2 Generación gráfica del robot.

El apartado de Render GUI del robot ya se ha visto anteriormente, y en este apartado hay varias funciones que definen la generación y llamado de archivos adicionales para la visualización del robot de forma real, estos archivos CAD son especificaciones muy aproximadas a un manipulador industrial y para poder introducirlas a la plataforma `ADEFID` se requieren llamar en formato STL. Las principales funciones para la generación de los archivos STL del robot son las que se muestran en la figura 5.3.

```
// Render GUI
virtual void SetBaseLabels(void);
void Draw_3DFrameProx(D_H* pDHParam, int linkNumber);
void Draw_3DFrameDist(D_H* pDHParam, int linkNumber);
virtual bool ReadSTLFiles(CString * pFilesPath);
```

Figura 5.3: Funciones Render GUI del manipulador.

Las funciones `Draw_3DFrameProx`() y `Draw_3DFrameDist`() generan los vectores de los ejes coordenado de cada eslabón del manipulador, y como su nombre lo indica están referidos a los sistemas de coordenadas proximal y distal. Estas funciones reciben solo dos parámetros, un puntero que contiene las variables DH y un entero que especifica el eslabón que se esté utilizando del manipulador, de forma que pasa por cada eslabón para generar su sistema de referencia. Una visualización de estas coordenadas se muestra en las figuras 4.29a y 4.29b.

La función `ReadSTLFiles`() es aquella que hace el llamado y el posicionamiento de cada uno de los eslabones del robot en el formato STL, es importante tener cada diseño del eslabón en el primer octante ya que es probable que no se visualice en el entorno gráfico si no se encuentra situado en el octante positivo. Es necesario saber las distancias desplazadas de cada diseño al origen para poder situar de manera correcta cada eslabón en la cadena cinemática del robot, como lo muestra el diseño final del robot en la figura 5.4. La función `ReadSTLFiles` recibe una cadena de texto String la cual almacena e indica la ubicación donde se encuentran almacenados los archivos STL de los eslabones del robot.

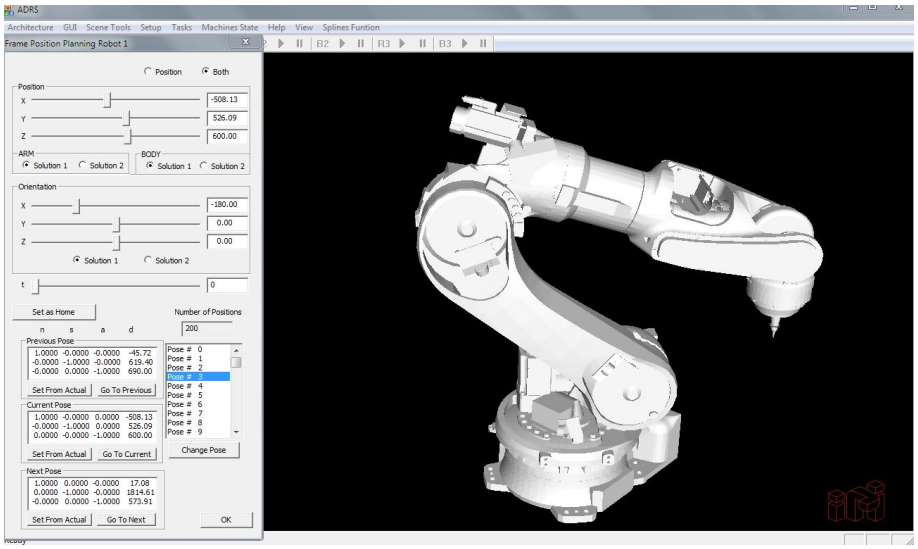


Figura 5.4: Cuadro final del robot Kawasaki en escena en ADEFID.

5.1.3 Generación del listado de poses de apoyo.

En esta sección se encuentran tres funciones importantes para la creación, almacenamiento y actualización de poses de apoyo que el usuario quiera manipular.

La función `InitPoses`() crea e inicializa el listado de las poses que el usuario quiere almacenar, por cada pose del efector final en el manipulador se crean cuatro renglones y en cada renglón hay cuatro variables, esto representa a la matriz de transformación homogénea T, como en un inicio se mencionó que se tienen disponibles 200 poses, por lo cual esta función crea por lo menos 800 líneas que contienen todas las variables de la matriz T $((200 * 4) = 800)$. Sin embargo, esta función sólo inicializa y crea una lista vacía, no tiene las poses que el usuario quiere almacenar, Fig. 5.5.

Aquí la función `SavePoses`(), Fig. 5.6, es la que almacena cada una de las poses en el archivo de texto que creo anteriormente la función `InitPoses`(). Ahora con la clase de diálogo `CFramePositionPlanningDlg` es posible interactuar con las poses que se quieran almacenar, la cual utiliza la función `SavePoses`() para guardar cada parámetro de la matriz T. De manera que cada renglón y columna no tiene un valor cero, sino que ahora es cada variable de la pose de apoyo la que es almacenada. La figura 5.8 muestra el listado total de las 200 poses de apoyo guardadas en un archivo de texto.

Finalmente, la función `UpdatePoses`(), Fig. 5.7, es la que actualiza todas las poses de apoyo guardadas previamente. Para interactuar visualmente también se utiliza la clase diálogo `CFramePositionPlanningDlg`, la cual llama esta función cuando se requiera cambiar o editar la posición u orientación de la pose de apoyo. Esta función también es llamada cada vez que se abre el proyecto con la función `SetDefaults`() para hacer una actualización de las poses más recientemente guardadas.

5.1.4 Algoritmo de simulación.

```
void CCMKawasakiBX100N::Algorithm(void)
```

El algoritmo de simulación se encarga de hacer todas las actualizaciones de pantalla, el cálculo de posiciones y la creación de trayectorias lineales y

```
void CCMKawasakiBX100N::InitPoses(void)
{
    int ij=1;
    m_Poses.m_Str[0]=_T("Not used");
    for(int k=0;k<4;k++)
        m_Poses.m_Vector[0][k]=0;
    //Defines Reference pose of the manipulator
    for(int j=0;j<4;j++){
        for(int k=0;k<4;k++){
            if(j==k)
                m_Poses.m_Vector[ij][k]=1;
            else
                m_Poses.m_Vector[ij][k]=0;
        }
        m_Poses.m_Str[ij].Format("Ref P.");
        ij++;
    }
    //defines the poses of the manipulator
    for(int i=0;i<N_POSBX100N;i++)
    {
        PosBX100N[i].row=PosBX100N[i].col=4;
        for(int j=0;j<4;j++)
        {
            for(int k=0;k<4;k++)
            {
                if(j==k)
                    m_Poses.m_Vector[ij][k]=1;
                else
                    m_Poses.m_Vector[ij][k]=0;
            }
            m_Poses.m_Str[ij].Format("Pose %d",i);
            ij++;
        }
    }m_Poses.SaveData();
}
```

Figura 5.5: Función InitPoses().

cuvilíneas para que el usuario pueda observar en la pantalla la simulación de movimiento. De forma que la función `Algorithm`() involucra la generación de movimientos que debe realizar el robot en una o varias tareas designadas. En tal caso, como la tarea que se desea realizar es el corte ya sea por láser o por alguna herramienta de corte en la superficie de un automóvil, hay una secuencia de poses de inicio, trayectorias lineales que se deben ejecutar antes de que comience a trazar las trayectorias continuas en forma curvilínea. Como en toda simulación, se pretende que lo que el usuario vea virtualmente, sea lo más cercano a lo que pasaría en la realidad.

El código realizado para este apartado es un tanto extenso ya que tiene todos los estados posibles y la secuencia de tareas necesarias para la ejecución de la aplicación de corte con trayectorias continuas.

```
void CCMKawasakiBX100N::SavePoses(void)
{
    int ij=1;
    for(int j=0;j<4;j++)
    {
        for(int k=0;k<4;k++)
        {
            m_Poses.m_Vector[ij][k]=m_RefPose.q[j][k];
        }
        ij++;
    }
    for(int i=0;i<N_POSES;i++)
    {
        for(int j=0;j<4;j++)
        {
            for(int k=0;k<4;k++)
            {
                m_Poses.m_Vector[ij][k]=PosBX100N[i].q[j][k];
            }
            ij++;
        }
    }
    m_Poses.SaveData();
}
```

Figura 5.6: Función SavePoses().

```
void CCMKawasakiBX100N::UpdatePoses(void)
{
    m_Poses.LoadData();
    int ij=1;
    for(int j=0;j<4;j++)
    {
        for(int k=0;k<4;k++)
        {
            m_RefPose.q[j][k]=m_Poses.m_Vector[ij][k];
        }
        ij++;
    }
    for(int i=0;i<N_POSBX100N;i++)
    {
        for(int j=0;j<4;j++)
        {
            for(int k=0;k<4;k++)
            {
                PosBX100N[i].q[j][k]=m_Poses.m_Vector[ij][k];
            }
            ij++;
        }
    }
}
```

Figura 5.7: Función UpdatePoses().

El pseudocódigo 7 de la función `Algorithm`() muestra la base en la que tiene la lista de cada estado que contiene poses, trayectorias lineales y trayectorias continuas que se estarán ejecutando en la plataforma `ADEFID`.

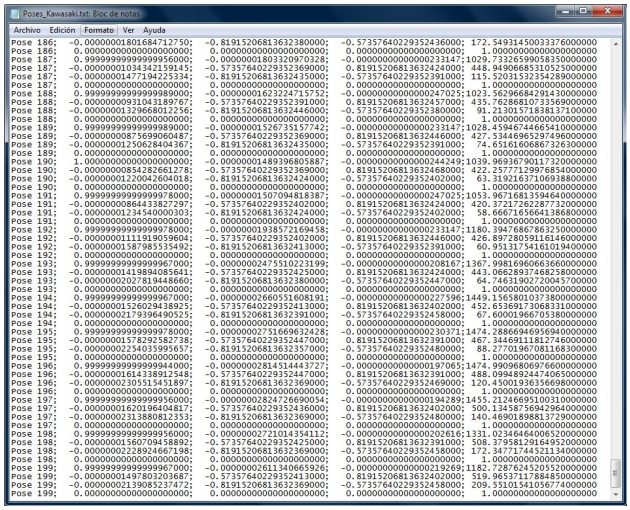


Figura 5.8: Lista de las poses de apoyo almacenadas en un archivo de texto.

Pseudocódigo 7 *Algorithm()* de la clase *CCMKawasakiBX100N*

function ALGORITHM(*void*)
  beginSwitch (nSate)
  ⋮
    begincase `STATE1`:
      if(`FALSE`==Timer.blsExpired()) **then**
        if(`TRUE`==*G*etFrameParameter(Timer, &η)) **then**
          $Q_m, d_m \leftarrow T_m \leftarrow$ GetPose(η)
          SetFullInverse(DH,$\mathbf{Q_m}$, $\mathbf{d_m}$)
        end if
      else
        SetState(`SUSPEND`)
      end if
    endcase
  ⋮
  endSwitch
end function

La función `Algorithm`() solamente se ejecutará si el manipulador está en cualquier modo diferente a `SUSPEND`. Para comenzar a ejecutar cada es-

tado se debe haber dado inicio "Cicle Start"y ejecuta cada estado de forma automática `System.SetMode(AUTOMATIC)`.

Cada estado de la función del algoritmo consta de un temporizador (timer), esto es, que el estado en que se encuentra el robot debe ejecutarlo en un tiempo ya previamente definido, de manera que actúa en forma de temporizador en cada estado, y es usado como la variable de tiempo para realizar los desplazamientos en cada uno de los movimientos que presenta el robot de la tarea que define el estado. Si el temporizador no ha expirado puede continuar con su tarea de movimiento en el manipulador, sin embargo, si ha terminado el timer termina la ejecución del actual estado y continua con el siguiente estado por medio del control `MaterialHandlingProcess` de la clase `CADEFIDDoc`.

Una parte muy importante en la ejecución de cada estado es la generación de movimiento del manipulador en la escena de la plataforma, de manera que la función `GetFrameParameter`() se encarga de realizar la actualización de la escena mostrada en pantalla, es decir, participa de manera activa en el cálculo de la nueva posición y orientación del robot para cada iteración realizada. Uno de los parámetros que `GetFrameParameter`() recibe es el tipo de interpolación en el movimiento que el efector final debe realizar, de entre las que se encuentran en la plataforma `ADEFID`, son las siguientes: `LINEAR, POLYNOMIAL345, POLYNOMIAL4567` y `CYCLOIDAL`.

La primera interpolación (`LINEAR`) implica una interpolación lineal entre condiciones iniciales y finales, donde su velocidad es constante y su aceleración es nula en todo su movimiento. Las siguientes dos interpolaciones tienen una interpolación de orden mayor. Cada una de estas interpolaciones implica considerar desde 6 condiciones o más para su planteamiento en la trayectoria de articulación. De manera que a mayor número de condiciones el grado del polinomio se incrementa, por ejemplo, la interpolación `POLYNOMIAL345` que representa la interpolación polinomial 3-4-5 tiene un polinomio de quinto grado, la interpolación `POLYNOMIAL4567` que representa la interpolación polinomial 4-5-6-7 tiene un polinomio de séptimo grado, y la última interpolación utiliza funciones armónicas (`CYCLOIDAL`), las cuales hacen visualizar movimientos repetitivos y de manera suave. Gracias a esta metodología es posible obtener un movimiento más real y dinámico, de forma

que sus movimientos en las articulaciones del manipulador tienen incrementos y decrementos de velocidad y aceleración.

Para el presente trabajo sólo se utilizaron las primeras dos interpolaciones en el movimiento del manipulador, la interpolación `LINEAR`, donde todos sus movimientos tienen velocidades constantes y aceleraciones nulas, y la interpolación polinomial 3-4-5 (que se explica con más detalle en la subsección 3.2.2), de manera que con el parámetro `POLYNOMIAL345` es posible observar cómo el manipulador acelera al inicio de su movimiento, después mantiene su velocidad constante y finalmente desacelera hasta llegar a su posición final en un tiempo definido. Con estas condiciones de interpolaciones se controla la velocidad del efector final.

Teniendo en cuenta qué tipo de movimiento es el que se va a ejecutar, ahora se debe indicar qué tipo de trayectoria es necesario trazar. Para seleccionar el tipo de clase que debe ejecutarse en la trayectoria deseada, se hace uso de `CLinearPath` para moverse de punto a punto linealmente, y `CParametricSpline` para moverse por una curva paramétrica creada por N puntos de apoyo. En el algoritmo de control interviene la función GetFrameParameter() , donde el timer va generando valores de η en un tiempo determinado. Estos valores de η son fundamentales en las siguientes funciones, ya que son los que van ir parametrizando el movimiento del efector final en la trayectoria.

La función `GetPose`() recibe cada valor de η en un tiempo discreto, donde después genera la matriz de orientación y el vector de posición del efector final en cada valor de η que fue arrojado por `GetFrameParameter`.

Finalmente la función `SetFullInverse`() recibe los parámetros DH y la matriz de transformación homogénea por el parámetro η, la función por medio de la cinemática inversa hace los cálculos de las variables de articulación y así genera los movimientos de cada eslabón del robot.

Con los timers utilizados dentro del algoritmo se controlan el lapso de interpolación de movimiento en cada trayectoria, los timers parametrizan la variable η en cada estado, de esta manera es posible acoplar el movimiento del manipulador en la plataforma de simulación en tiempo real como si se estuviese ejecutando en un robot industrial real.

5.2 Clase FordMustang.

Como la aplicación de la tesis es el corte de partes de un automóvil, la clase `FordMustang` crea el objeto de la carcasa de un automóvil Mustang que se dibuja en el entorno gráfico de la plataforma `ADEFID`. Así como se utilizó para la clase `CCMKawasakiBX100N`, `ReadSTLFiles`() se utiliza aquí de manera similar en la clase `FordMustang`, la cual hace el llamado del archivo STL del automóvil.

De esta manera en la clase `CADEFIDRender`, las funciones `SetupScene`() y `RenderUScene`() hacen aparecer el diseño del automóvil en la plataforma `ADEFID`, como se muestra en la figura 5.9.

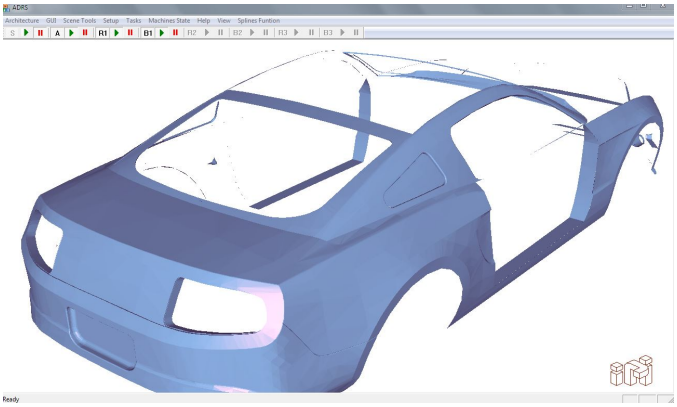


Figura 5.9: CAD del automóvil Ford Mustang en ADEFID.

5.3 Cuadros de diálogo.

Los cuadros de diálogo, son nombrados como diálogos, y se usan para presentar información y recopilar datos del usuario. Los cuadros de diálogo se proporcionan en todas las formas y tamaños, que van desde cuadros de mensajes sencillos que despliegan una sola línea de texto, hasta grandes cuadros de diálogo que contienen controles complejos. Los cuadros de diálogo son de dos tipos: modales y no modales. Un cuadro de diálogo modal debe estar cerrado para el usuario antes de que la aplicación continúe. Y un cuadro de diálogo no modal permite al usuario mostrar el cuadro de diálogo y volver a otra tarea sin cancelar o eliminar el cuadro de diálogo.

El entorno gráfico de `ADEFID` cuenta con elementos que permiten interactuar con la simulación del manipulador. De entre los elementos que son importantes en el presente trabajo, se encuentra la manipulación de las poses de apoyo que el robot debe trazar para formar una curva, que se logra a través de la clase de diálogo `CFramePositionPlanningDlg`. Más aún, la manipulación del robot se logra con los diálogos que evalúan tanto la cinemática directa como la cinemática inversa, a través de las clases diálogo `CDirectKinDlg` y `CFullInvKinDlg`, respectivamente, como se muestran en la Fig. 5.11.

La clase diálogo que intervine en el cambio de apariencias y de sistemas de referencias es `CManualSettingsDlg` (ver Fig. 5.10) en la cual se puede visualizar el manipulador en forma real con cada uno de sus componentes reales del fabricante en STL, o visualizar al manipulador en su forma de cadena cinemática y/o en su esqueleto que lo define la tabla DH. Adicionalmente se implementó la clase `CSplinesVariablesDlg` que corresponde al diálogo en el que se visualizan todas y cada una de las variables, constantes, parámetros y matrices que definen las operaciones Splines.

Cabe mencionar que hay más clases diálogo que están en funcionamiento en el entorno gráfico de `ADEFID`, el trabajo de Padierna-Garcia [36] describe los cuadros de diálogos útiles en la modificación de parámetros de escena y ambiente gráfico de la plataforma `ADEFID`. Sin embargo, los que se mencionaron anteriormente son los más usados para la simulación de manipuladores industriales con trayectorias lineales y continuas.

Las clases diálogo que contribuyen a la simulación del presente trabajo son `CFramePositionPlanningDlg` y `CSplinesVariablesDlg`, las cuales se describen en la siguiente sección.

5.4 Clase CFramePositionPlanningDlg.

Para poder crear los polinomios cúbicos entre cada postura vecina, se necesitan todos los puntos de apoyo de la trayectoria. Se definieron 7 trayectorias con 25 puntos (ver Fig. 5.12), dando un total de 175 puntos de apoyo y 25 puntos se utilizaron para posturas arbitrarias.

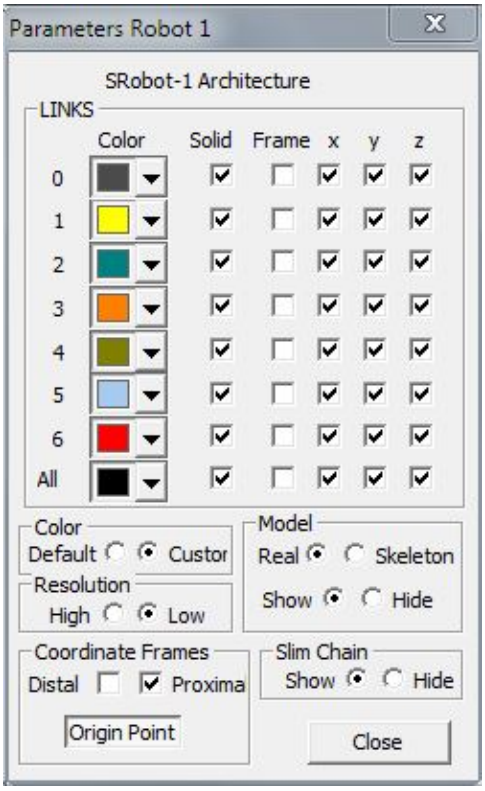
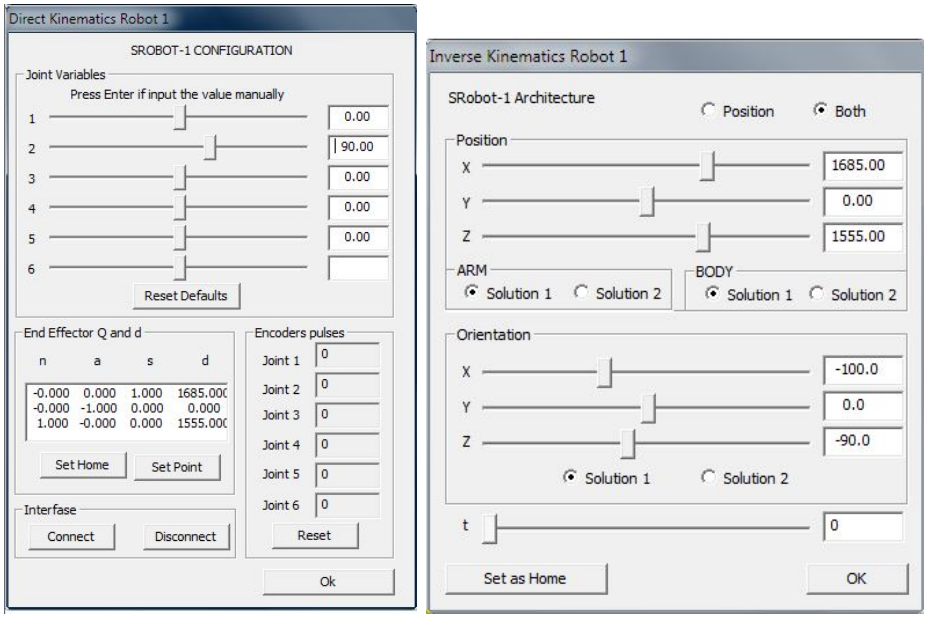


Figura 5.10: Edición de los parámetros del robot por la clase CManualSettingsDlg.

De manera que para conocer tales puntos de apoyo se utilizó un software CAD para trazar trayectorias sobre la superficie del automóvil y asignar puntos en la trayectoria. Así los puntos designados fueron posiciones reales en la posición de un automóvil Ford Mustang con dimensiones reales.

La clase diálogo `CFramePositionPlanningDlg` es donde interactúa la edición y almacenamiento de todas estas poses de apoyo en el documento. Esta clase forma parte de la clasificación de los no modales, ya que para la edición de los puntos de apoyo se necesitan otras clases diálogo para interactuar con las posiciones de cada articulación de los eslabones y/o la orientación y posicionamiento del efector final del robot. En tal caso, la clase `CFramePositionPlanningDlg` contiene una sección de control en la cinemática inversa del manipulador, en la figura 5.13 se muestra el control de la cinemática inversa (parte superior), se incluyó esta sección en el



(a) Clase diálogo DirectKinematics-Dlg.

(b) Clase diálogo InverseKinematicsDlg.

Figura 5.11: Ventanas de control en la cinemática directa e inversa del manipulador.

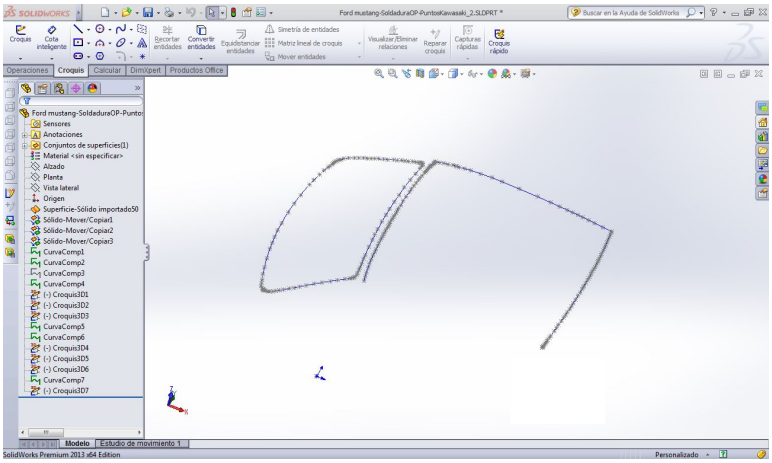


Figura 5.12: Puntos de apoyo definidos en diferentes trayectos

mismo cuadro de diálogo para hacer más rápida la edición de las poses de apoyo con la posición y orientación del robot en un espacio cartesiano. Cabe mencionar que también se pueden abrir las clases diálogos `CDirectKinDlg` y

`CFullInvKinDlg` para la manipulación del robot en trama de posicionamiento.

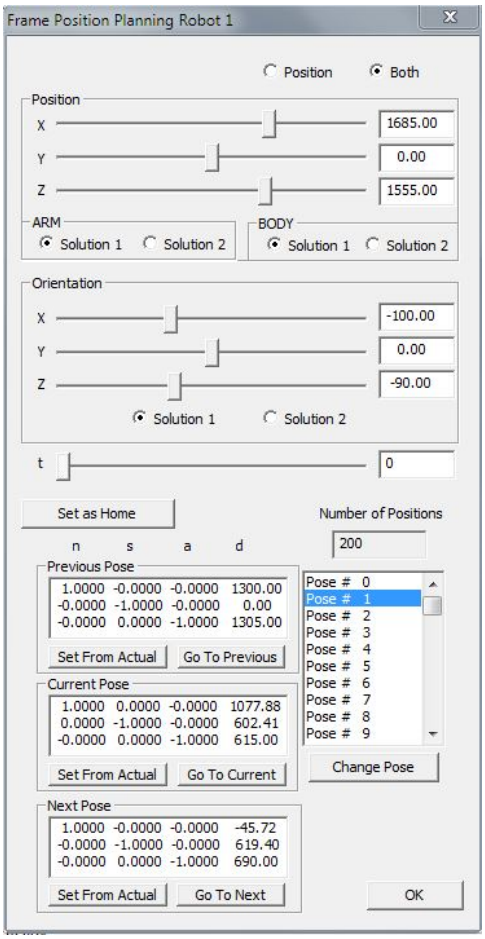


Figura 5.13: Cuadro de diálogo perteneciente a la clase CFramePosition-PlanningDlg

Sin embargo, la parte más importante en esta clase de diálogo es la manipulación de las poses de apoyo las cuáles son parte importante para crear los polinomios cúbicos Spline de una trayectoria curvilínea. La figura 5.14 muestra el control de la planeación de tramas de poses. En este control se muestran cuatro Listas de texto, en la primera que se encuentra en la parte derecha, se ilustran todas las poses disponibles (en este caso solo 200), donde con el cursor se tiene acceso a la selección de cualquier pose en el documento y ver su matriz de transformación homogénea (T) presionando el botón "Change Pose", que incluye la matriz de orientación y el vector de

posición del efector final.

La segunda parte de este control son las siguientes tres listas de texto que ilustran la matriz de orientación y el vector de posicionamiento de la pose deseada. Estas tres listas de texto muestran la pose previa (Previous Pose), la pose actual (Current Pose) y la pose siguiente (Next Pose) de la pose seleccionada en la lista de poses totales.

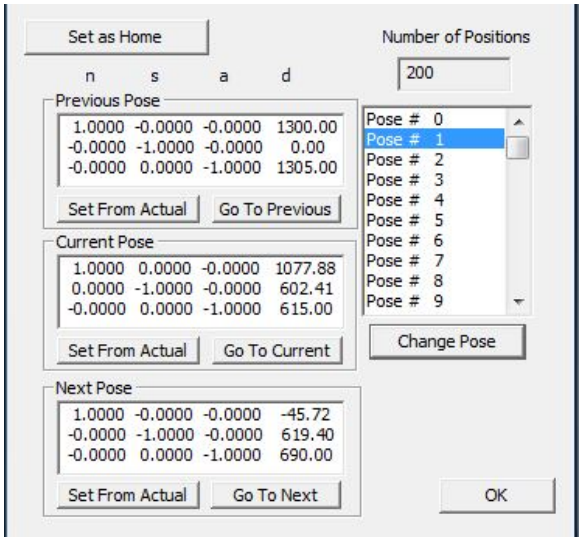


Figura 5.14: Sección de edición y almacenamiento de las poses de apoyo.

En la figura 5.14 está parte del control de la clase y contiene dos botones importantes para la edición de las poses de apoyo. El botón "Set From Actual"sirve para guardar la pose actual con los controles de la cinemática inversa o cinemática directa, esto es, que si se movieron las articulaciones del robot o se movieron la posición y orientación y posición del robot y se presiona este botón, automáticamente la clase diálogo guarda la trama matriz de transformación homogénea al documento y la guarda directamente en el archivo de texto. Para la edición de una pose ya anteriormente guardada es el mismo procedimiento como si no se tuviese una pose anterior, solamente se lleva a un nuevo posicionamiento y orientación del robot y se presiona el botón "Set From Actual"para editar una nueva pose del manipulador.

Por último, cuando se hayan guardado las poses necesarias que el usuario requiera, el botón "Go To. . .Previous, Current, Next"muestra en el ambien-

te gráfico la pose guardada con el manipulador, de manera que es posible visualizar con el robot las poses guardadas anteriormente.

5.5 Clase CSplinesVariablesDlg.

Las funciones Splines crean múltiples operaciones matriciales y gran cantidad de cálculos matemáticos, por lo que es fundamental monitorear y analizar cada operación para el correcto funcionamiento de la simulación del manipulador. La clase `CSplinesVariablesDlg` se enfoca en la visualización de las operaciones más importantes en los cálculos realizados para la creación de los polinomios cúbicos Spline con base en los puntos de apoyo almacenados mediante el uso de la clase `CFramePositionPlanningDlg`, la ventana de diálogo asociado a la clase `CSplinesVariablesDlg` se ilustra en la figura 5.15. Es importante mencionar que esta clase diálogo solamente sirve para la visualización y la correcta verificación de las operaciones matriciales de dichas funciones.

La clase `CSplinesVariablesDlg` se divide en tres secciones que son visualmente importantes al momento de verificar cálculos y operaciones. En primer lugar la sección número 1 se encuentra en la parte superior izquierda, que involucran dos listas de texto, la lista de texto más grande muestra una lista grande en la que están todos los puntos de apoyo definidos en una trayectoria previamente seleccionada, de estos puntos de apoyo se hacen los cálculos de las variables $\Delta_X, \Delta_Y, \Delta_Z, \Delta\sigma_k$, y σ_k. Como se vio en el capítulo 4, todas estas variables iniciales se calculan con la funcion `InitCartPoints`() de la clase `CParametricSpline`. La otra lista de esta misma sección muestra las poses de apoyo que conforman la trayectoria seleccionada a trazar.

La sección número 2 de esta ventana de diálogo consta de la visualización de todas las operaciones matriciales y las matrices para calcular la matriz final P''. Aparecen las matrices A, C , P y de éstas las operaciones matriciales de la función Spline. En un momento dado, fue importante tener esta sección ya que era difícil verificar múltiples operaciones para el resultado final de la creación de los polinomios cúbicos Spline. La sección se calcula por la función `InitSplineMatrices`() de la clase `CParametricSpline`.

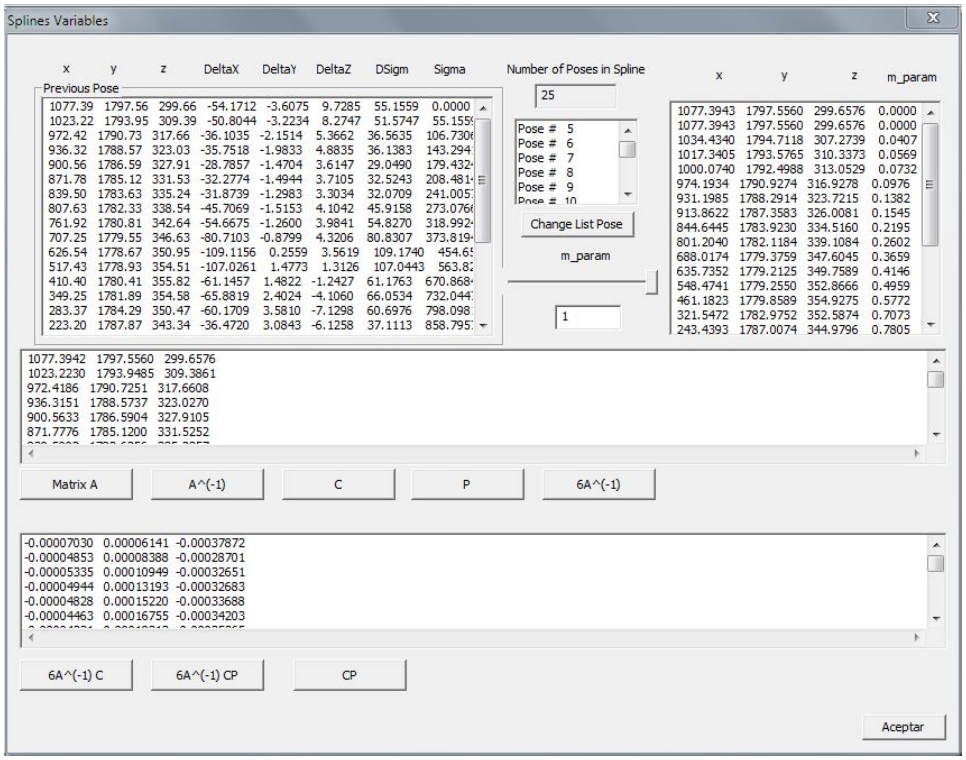


Figura 5.15: Cuadro de diálogo perteneciente a la clase CSplinesVariables-Dlg.

Finalmente la última sección, la sección número 3 se encuentra en la parte superior derecha, y la conforman un elemento de control y una lista de texto que muestra los resultados finales. El elemento de control es una barra de control (Scroll Bar) en la que está parametrizada la variable de entrada η, que tiene valores discretizados de 0 a 1, y opera en la función `GetPose`() y `PathParameter`() de la clase `CParametricSpline`. El otro elemento es una lista de texto que muestra una lista de posiciones resultantes al control del valor de η, estos puntos cartesianos representan las posiciones de la curva polinómica Spline creada por todo el sistema Spline. Conforme se va cambiando el valor de η, se obtienen los puntos que el efector final del manipulador tiene que trazar como tarea de corte en el automóvil. Aquí también salen los puntos de apoyo inicialmente almacenados, pero no solo estos puntos son obtenidos sino también todos los puntos necesarios para la nueva curva Spilne creada por las funciones `GetPose`() y `PathParameter`().

5.6 Control de movimiento.

El entorno gráfico que involucra todo el sistema de ADEFID, tiene acceso a todas las clases diálogo ya previamente mencionadas. Una vez que todas las poses de apoyo se hayan almacenado, que las trayectorias ya se hayan definido en la clase del manipulador y la función `Algorithm(void)` tenga toda la lista de secuencias de movimientos y trayectorias a realizar, es posible dar inicio a la simulación de la tarea que se ha planeado para el robot con "Cicle Start".

Dentro de ADEFIDDoc se encuentra la función `MaterialHandlingProcess` [22], que pertenece al bloque "Update Machine States"del diagrama de flujo principal de ADEFID, el robot establece diferentes estados, dependiendo de las variables de control:

- **RESET**. Si el Manipulador no está en HOME, se mueve en línea recta de la posición arbitraria hasta HOME, (ver Fig. 5.16).
- **PIN**. El manipulador se mueve en línea recta de HOME al punto inicial de la trayectoria principal (Primer punto en el corte del automóvil), (ver Fig. 5.18).
- **POUT**. El manipulador realiza la tarea principal (Sigue la trayectoria continua Spline conformada por los puntos de apoyo), (ver Fig. 5.19).
- **SUSPEND**. No se realiza ninguna acción en el sistema, no hay ningún movimiento en las trayectorias planeadas, sin embargo, se pueden hacer cambios y modificaciones de las posiciones de apoyo como se muestra en la figura 5.17.

El presente capítulo presentó los resultados finales del trabajo obtenido de los capítulos 3 y 4. Donde mayormente el capítulo 3 presenta el método Splines en forma matemática y su funcionamiento, el capítulo 4 presenta los algoritmos utilizados para generar las funciones Splines en la plataforma ADEFID. Este capítulo presenta la ejecución del robot adecuado a una tarea industrial en ADEFID con las funciones anteriormente mencionadas. El siguiente capítulo presentará las conclusiones finales de todo el trabajo y los posibles trabajos futuros.

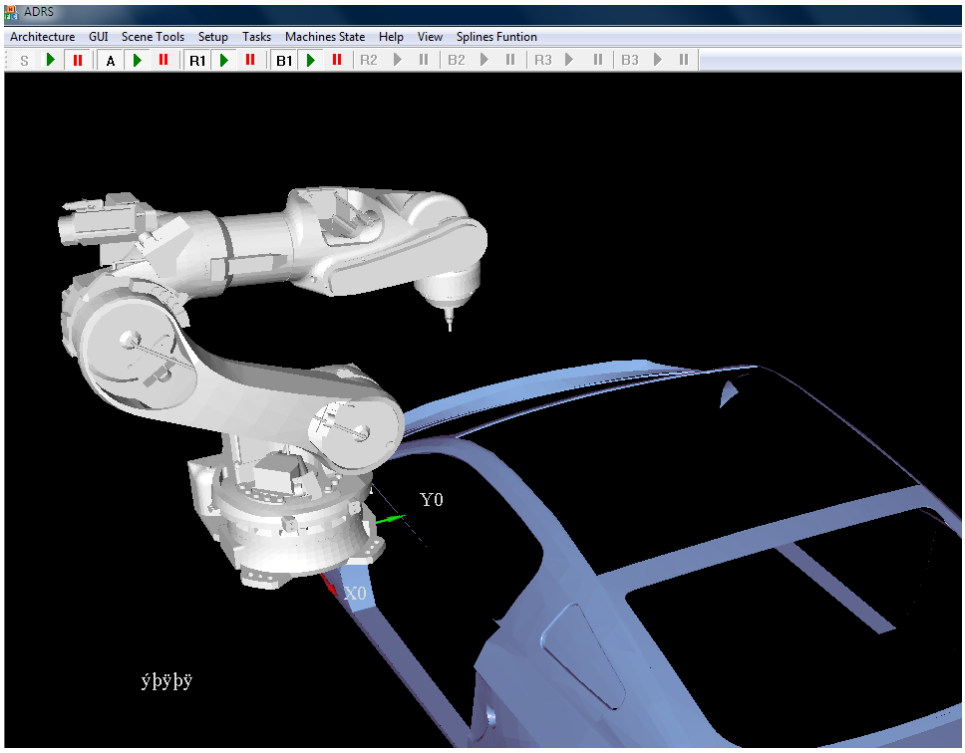


Figura 5.16: Estado RESET del manipulador.

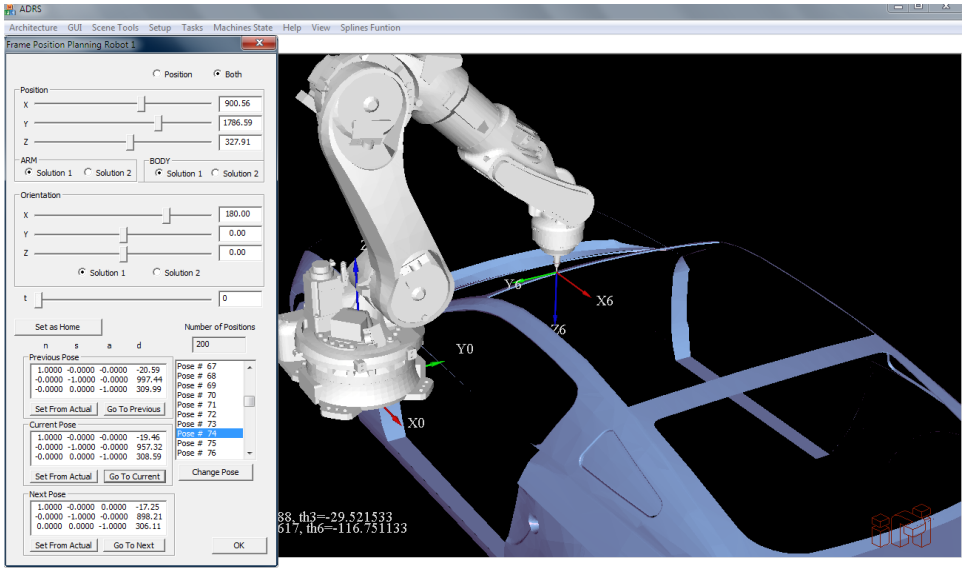
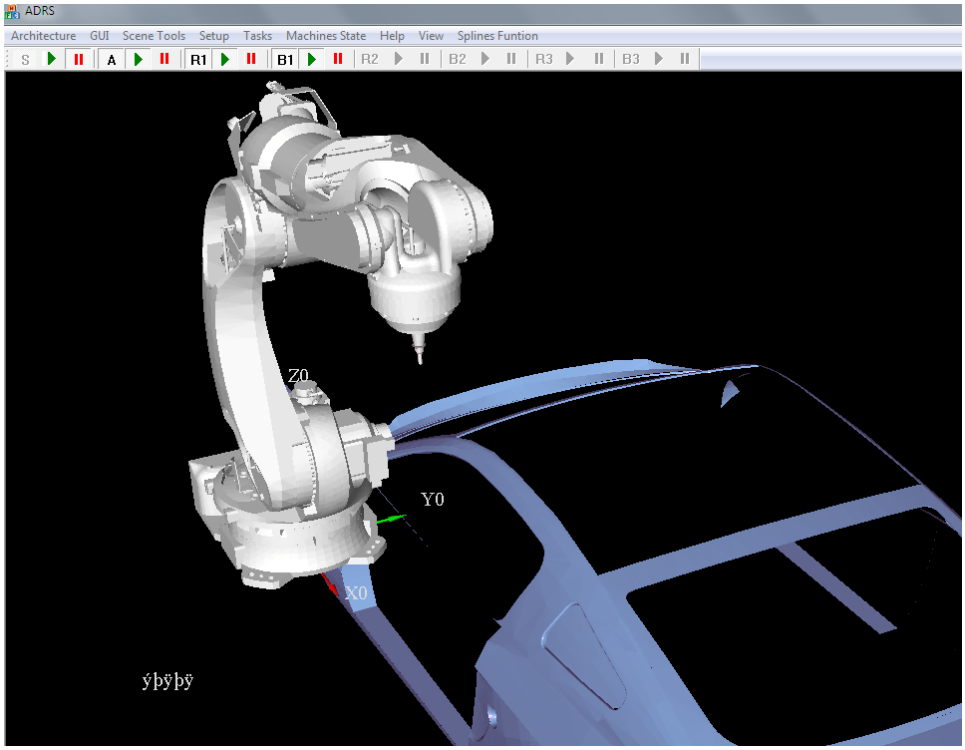
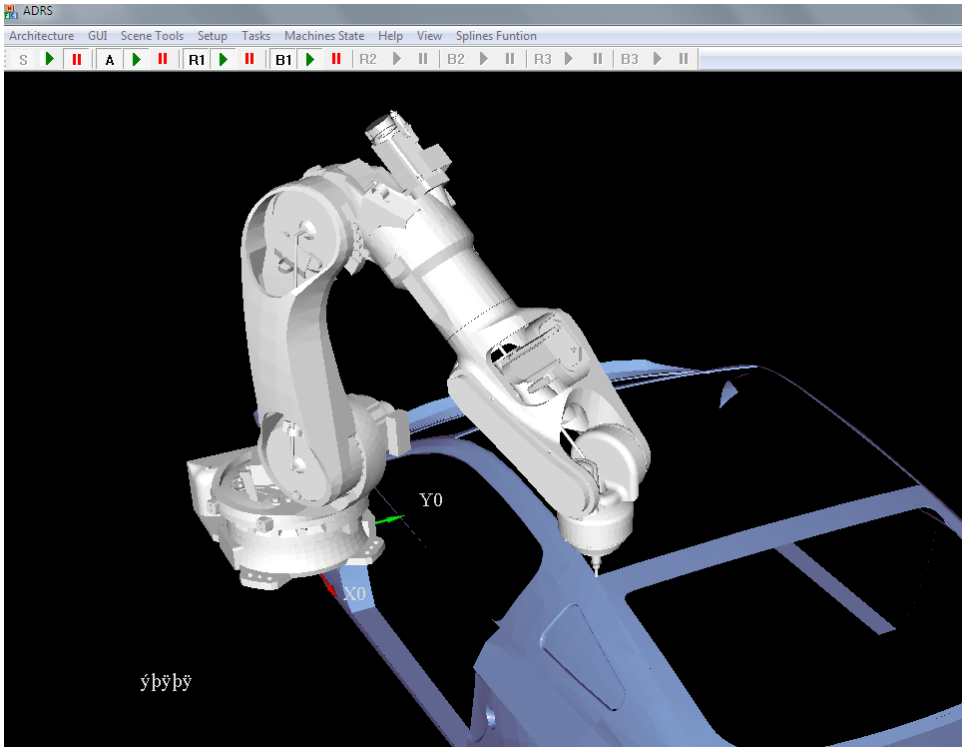


Figura 5.17: Estado SUSPEND del manipulador.

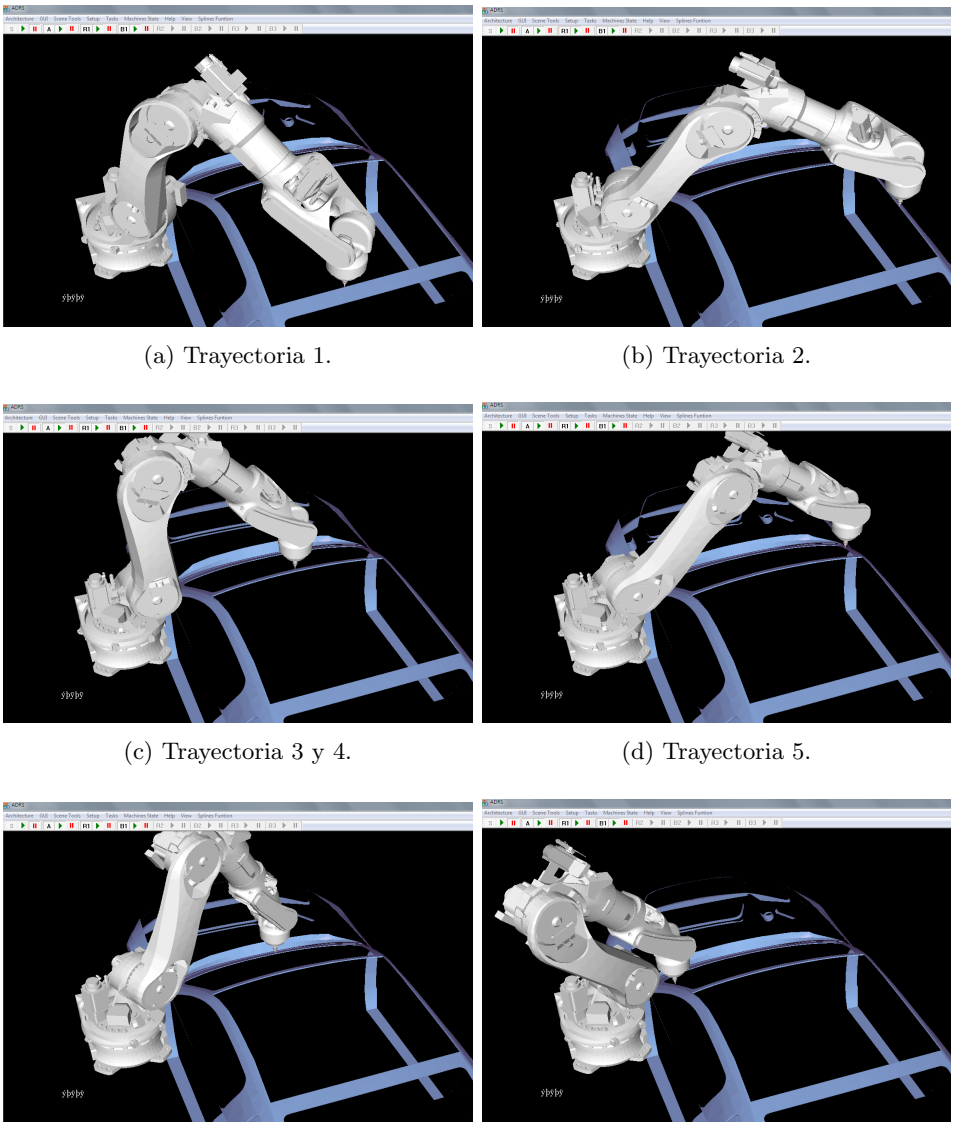


(a) Home.



(b) Punto inicial de la trayectoria.

Figura 5.18: Estado PIN del manipulador.



(a) Trayectoria 1.

(b) Trayectoria 2.

(c) Trayectoria 3 y 4.

(d) Trayectoria 5.

(e) Trayectoria 6.

(f) Trayectoria 7.

Figura 5.19: Estado POUT del manipulador.

Capítulo 6

Conclusiones

Contenido

6.1 Conclusiones generales.

La planificación de trayectorias curvilíneas con un brazo robótico fue parte clave para el desarrollo de esta tesis. En un principio se planteó tomar ecuaciones paramétricas ya definidas, tales como la ecuación de una Hélice, de manera que el usuario solo tenía que ingresar la función paramétrica al sistema. Sin embargo, gran cantidad de aplicaciones y procesos que la industria maneja, no cuenta con las funciones matemáticas ya definidas para que el robot las trace, ya que pueden tornarse muy complejas y se necesita de un gran procesamiento computacional para generarlas. Por lo que se optó en utilizar las funciones splines que generan tales trayectorias con poco procesamiento computacional. Estas funciones solo requieren del conocimiento de múltiples puntos de apoyo, de tal forma que pueden crearse polinomios cúbicos en forma paramétrica y se genera un polinomio diferente por cada conjunto de poses de apoyo que sean vecinos entre sí. Estos puntos de apoyo son específicos para la generación de la trayectoria que se desea trazar.

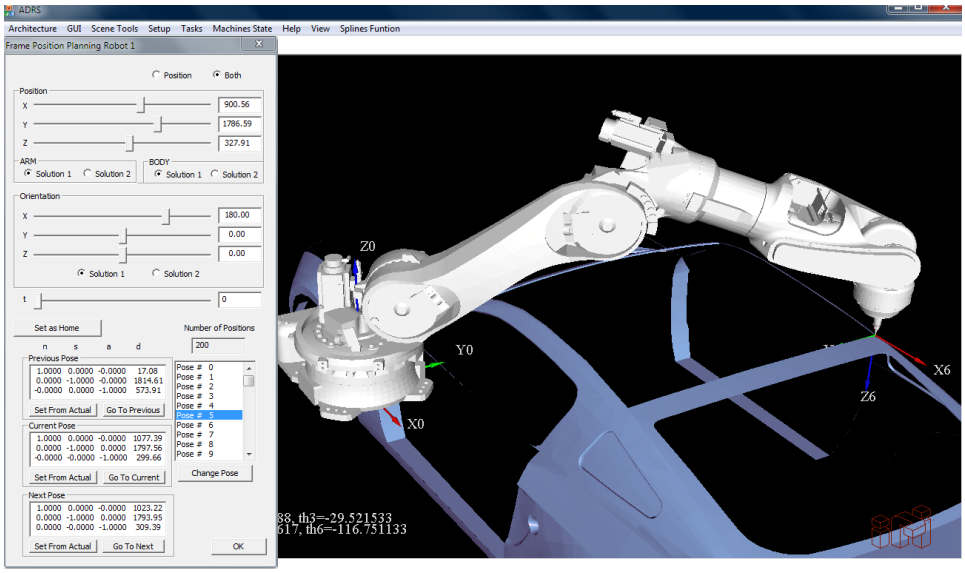


Figura 6.1: Aplicación de corte de un automóvil.

Con la integración de la clase `CParametricSplines` al sistema, pueden desarrollarse gran cantidad de aplicaciones industriales y/o procesos que requieren de una planificación de trayectorias curvilíneas complejas o de gran

precisión en un espacio de trabajo tridimensional. El manipulador Kawasaki BX100N fue uno de los manipuladores que por sus dimensiones físicas fue posible incluirlo a la aplicación de corte y/o soldadura en el área de la industria automotriz (ver Fig. 6.1). Sin embargo, gracias a la flexibilidad de la plataforma se pueden integrar otros manipuladores que tengan diferente geometría. Es importante considerar que si hay una curva muy cerrada, se debe poner la mayor cantidad de puntos de apoyo posibles en tal sección, para que su trayectoria sea la más aproximada a una curva suave. Entre más cercanos sean los puntos de apoyo, más precisa será su aproximación en las propiedades de una curva.

Una de las principales aportaciones de este trabajo fue crear el control de movimientos y ejecución de trayectorias del brazo robótico dentro de la plataforma `ADEFID`, con las múltiples poses de apoyo fue posible crear una curva que represente la trayectoria continua designada por el usuario y el monitoreo de las variables de la función spline fue de gran ayuda para verificar todos los cálculos matemáticos. Para una mayor visualización del sistema desarrollado, se hizo un video con la simulación del manipulador haciendo una tarea de corte a un automóvil, dicho video se pude ver en YouTube [38].

6.2 Trabajos futuros.

Con base en el sistema actual se puede implementar una gran cantidad de aplicaciones distintas, la función paramétrica spline puede generar múltiples trayectorias continuas en un espacio cartesiano, así que con esta función pueden desarrollarse aplicaciones industriales de mayor nivel. Una de ellas es que haya un sistema en el cual múltiples manipuladores estén aplicando tareas diversas que sigan trayectorias ya sean lineales o continuas en un transportador (ver Fig. 6.2), donde el producto que se esté tratando sea trabajado por múltiples manipuladores, esto se maneja frecuentemente en industrias con grandes producciones en sus líneas de operación.

Otra área donde este proyecto puede seguirse desarrollando, es la orientación del efector final del robot en una trayectoria definida por múltiples puntos de apoyo. Ya que con las funciones Splines se crean trayectorias

curvilíneas que el efector final debe seguir en el espacio cartesiano, sin embargo, la orientación de tal trayectoria puede definirse con el plano osculante de cada punto de apoyo perteneciente a la trayectoria (como se muestra en la figura 3.4). La diferencia radica en que no se cuenta con funciones geométricas definidas, sino que solo se cuenta con los puntos de apoyo que crean cierta trayectoria curvilínea.

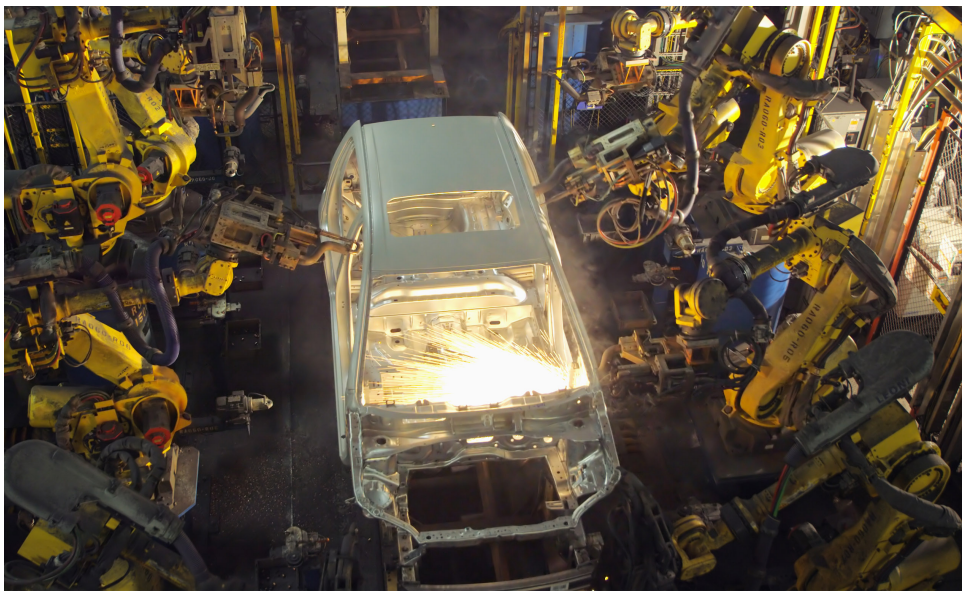


Figura 6.2: Robots ensamblando y soldando en una línea de producción [39].

Apéndice A

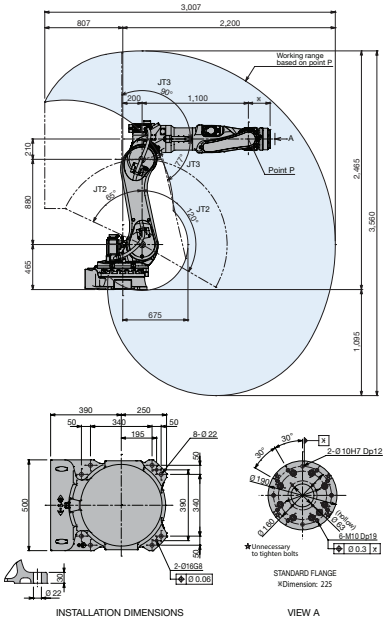
Hoja de datos Kawasaki BX100N

Hoja de datos del manipulador Kawasaki BX100N [17], datos y características fundamentales para su operación, tales como carga máxima, área de trabajo, velocidad, repetitividad, dimensiones específicas de cada eslabón del robot, entre otros.

Kawasaki

B series BX100N

MOTION RANGE & DIMENSIONS

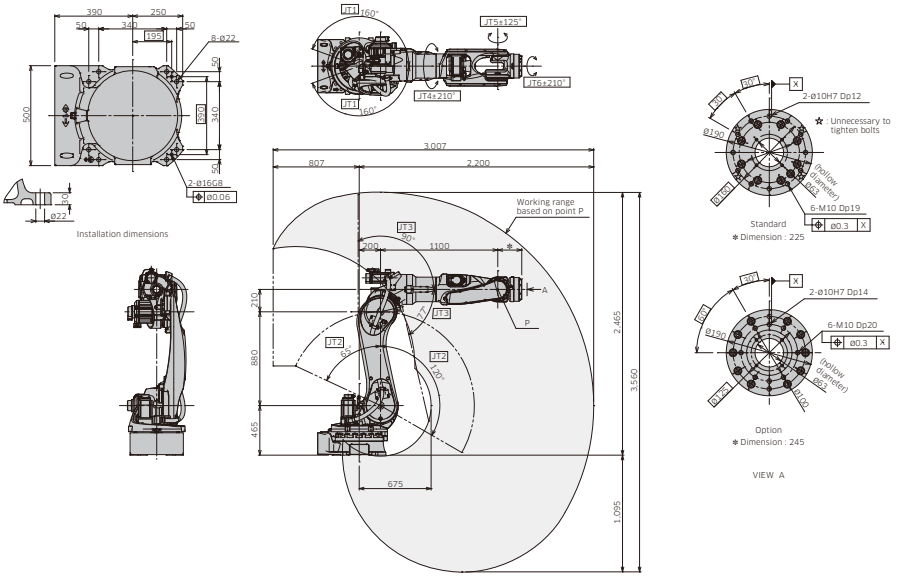


INSTALLATION DIMENSIONS VIEW A

(mm)

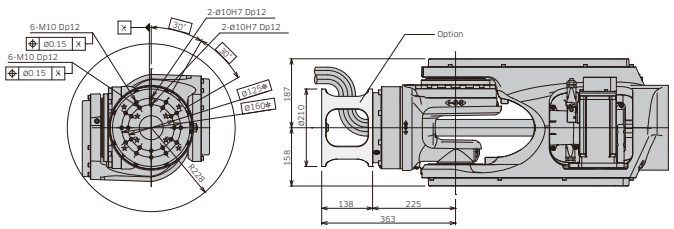
BX100N SPECIFICATIONS			
Type	Articulated		
Degrees of Freedom	6 axes		
Payload	100 kg		
Horizontal Reach	2,200 mm		
Vertical Reach	3,560 mm		
Repeatability	±0.2 mm		
Maximum Linear Speed	5,000 mm/s at robot flange		
Work Envelope (degrees) & Maximum Speed (degrees/s)	Axis	Motion Range	Maximum Speed
	JT1	±160°	135°/s
	JT2	+120° – -65°	110°/s
	JT3	+90° – -77°	140°/s
	JT4	±210°	200°/s
	JT5	±125°	200°/s
	JT6	±210°	300°/s
Wrist Load Capacity	Axis	Maximum Torque	Moment of Inertia
	JT4	588 N·m	60 kg·m²
	JT5	588 N·m	60 kg·m²
	JT6	294 N·m	30 kg·m²
Motor(s)	Brushless AC Servomotor		
Brakes	All axes		
Hard Stops	Adjustable mechanical stopper JT1		
Mass	740 kg (excluding Options)		
Body Color	Kawasaki Standard		
Installation	Floor		
Environmental Conditions	Temperature	0 – 45° C	
	Humidity	35 – 85 % (no dew, nor frost allowed)	
	Vibration	Less than 0.5 G	
Protection Classification	Wrist: IP67 Base: IP65 * Equivalent		
Options	Linear track options Risers Base plate Adjustable mechanical stopper JT2/JT3 Wrist flange adapter with ø125 mm mounting pattern Adapter bracket for weld guns and other tools Integrated dress package		
Controller	E32 (see E Controller data sheet for specifications)		

BX100N



Dimensions of the wrist

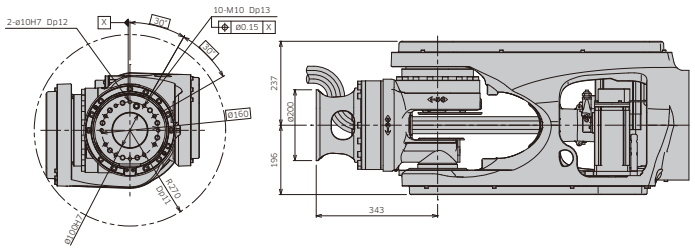
BX100S
BX100N
BX100L
BX130X
BX165N
BX165L
BX200L
BT200L



☆: Unnecessary to tighten bolts

*: There are two available specifications for the tool-mounting tap holes in the adapter bracket:
1) combination of 125 PCD (pitch circle diameter) and 160 PCD and 2) combination of 92 PCD and 125 PCD.

BX250L
BX300L



(mm)

Bibliografía

[1] Cubero, S. *Industrial Robotics: Theory, Modelling and Control*, Pro Literatur Verlag. Germany, Mammendorf, pages. 9–12, 2007.

[2] Sciavicco, L. & Siciliano, B. *Modelling and Control of Robot Manipulators*, 2nd edition, Springer-Verlag London. Great Britain, London, pages. 1–16, 185–212, 2000.

[3] Panasonic *Robot Panasonic-Sistema de Soldadura*, Disponible en: `http://orionautomation.com.au/`(Octubre 2015)

[4] Manipulación de Materiales *¿Qué es la automatización?*, Disponible en: `https://www.quiminet.com/articulos/que-es-la-automatizacion-27058.htm`(Octubre 2015)

[5] Producción Automatizada *Industria Manufacturera*, Disponible en: `http://www.campeche.com.mx/noticias/tecnologia/humanos-obsoletos-robots-reemplazaran-a-trabajadores/227261`(Octubre 2015)

[6] Robot Cartesiano *Dosificación Automática*, Disponible en: `http://www.dotestsl.com/robot-dosificador-cartesiano-2500.html`

[7] ABB *RobotStudio*, Disponible en: `http://new.abb.com/products/robotics/robotstudio` (Diciembre 2015)

[8] K-ROSET *Visual Components Robot Simulation with Kawasaki "K-ROSET"*, Disponible en: `https://youtu.be/MlkdhgfraD8` (Dic. 2015)

[9] Correa, J. C. & Vásquez, R. E. *Manipuladores Robóticos, una Mano para la Industria*, Revista Metal Actual. Colombia, Bogotá, Edición No. 29, pages. 1–6, 2013.

[10] Automatización industrial *Automatización industrial: del vapor a la luz* , Disponible en: `http://www.reporteroindustrial.com/temas/Automatizacion-industrial,-del-vapor-a-la-luz+98162?idioma=en` (Diciembre 2015)

[11] Proyectos Robóticos *Simuladores de Brazo Robot*, Disponible en: `https://sites.google.com/site/proyectosroboticos/Descargar-Simuladores` (Diciembre 2015)

[12] Coppelia Robotics *V-REP: Virtual Robot Experimentation Plataform*, Disponible en: `http://www.coppeliarobotics.com/index.html` (Febrero 2016)

[13] ADEFID *ADvanced Engineering platForm for Industrial Development*, Disponible en: `http://adefid.com/` (Junio 2016)

[14] The Atlantic *America's Coming Manufacturing Revolution*, Disponible en: `http://www.theatlantic.com/business/archive/2014/04/americas-coming-manufacturing-revolution/360931/` (Octubre 2015)

[15] Spong, M. W. & Hutchinson, S. *Robot Modeling and Control*, 2nd edition. Wiley. USA, New York, pages. 35–117, 2006.

[16] Los Robots *Un ejemplo de sistema de control.* , Disponible en: `http://roble.pntic.mec.es/jlop0164/archivos/ROBOTICA.pdf` (Octubre 2015)

[17] Kawasaki Robotics. *BX100N Robot data sheet*, Disponible en: `https://robotics.kawasaki.com/userAssets1/productPDF/BX100N.pdf` (Marzo 2016)

[18] KUKA Robotics. *KR 16 ARC HW*, Disponible en: `http://www.kuka-robotics.com/mexico/es/products/industrial_robots/special/arc_welding_robots/kr16_arc_hw/start.htm` (Marzo 2016)

[19] Efector Final *Corte y soldadura láser*, Disponible en: `http://www.interempresas.net/Deformacion-y-chapa/Articulos/`

62000-Rofin-FL-nuevo-motor-para-corte-y-soldadura-laser-de\
-fibra.html (Agosto 2015)

[20] Angeles, J. *Fundamentals of Robotic Mechanical Systems: Theory. Methods, and Algorithms*, Fourth edition. Springer. USA, New York, pages. 139–277, 465–501, 2014.

[21] Horton, I. *Beginning Visual C++ 2010*, 3rd edition. Wrox Press. USA, Austin, pages. 167–247, 2010.

[22] González-Palacios, M. A. *Advanced Engineering Platform for Industrial Development*, Journal of Applied Research and Technology. Vol. 10, No. 3, pages. 309–326, 2012.

[23] Peña-Gallo, R. *Diseño y construcción de una mesa de dos grados de libertad para pruebas de procesos industriales*. Tesis Maestría. Universidad de Guanajuato, 2009.

[24] González-Palacios, M. A. *The unified orthogonal architecture of industrial serial robots*, Robotics and Computer-Integrated Manufacturing. Vol. 29, No. 20, pages. 258–271, 2013.

[25] González-Palacios, M. A., González-Barbosa, E. A. and Aguilera-Cortés, L. A. *SnAM: A simulation software on serial manipulators*, Engineering with Computers (2013) 29: 87. Springer, 2012, DOI: 10.1007/s00366-011-0246-6.

[26] FANUC Robotics. *LR Mate 200iD/4SC*, Disponible en: http://www.fanuc.eu/es/es/robots/p%C3%A1gina-filtro-robots/serie-lrmate/lrmate-200id-4sc (Marzo 2016)

[27] González-Palacios, M. A. & Patlán-Rosales, P. A. *Manual de Referencia de librería ADEFID*, Universidad de Guanajuato, 2009.

[28] González-Palacios MA, Ortega-Alvarez CJ, Sandoval-Castillo JG, Cuevas-Ledesma SM and Mendoza-Patiño FJ *The Generalized Architecture of the Spherical Serial Manipulator*, Advances in Robotics & Automation. Vol. 5, No. 2, pages. 2–8, 2016.

[29] Adept Robotics. *Six-Axis Robot - Adept Viper s650*, Disponible en: `http://www.adept.com/products/robots/6-axis/viper-s650/general`

[30] FANUC Robotics. *LR Mate 200iB Datasheet*, Disponible en: `http://geek.nmt.edu/~bruder/final_project_files/Fanuc%20LR%20Mate_200ib_200ib_3l.pdf` (Marzo 2016)

[31] Kawasaki Robotics. *BX100N Robot*, Disponible en: `https://robotics.kawasaki.com/en1/products/robots/spot-welding/BX100N/` (Septiembre 2015)

[32] Morales-Gonzalez, G. A. *Interacción de Plataforma ADEFID con Robot Industrial Fanuc Para el Sensado y Manipulación de Objetos.* Tesis Licenciatura. Universidad de Guanajuato, 2016.

[33] SolidWoks 2013 *SolidWorks Tutorials*, Disponible en: `http://www.solidworks.com/sw/resources/solidworks-tutorials.htm` (Marzo 2016)

[34] Kawasaki Robotics. *B-Series robots brochure*, Disponible en: `https://robotics.kawasaki.com/userAssets1/productPDF/Kawasaki_B-Series.pdf` (Octubre 2015)

[35] Craig, J. J. *Robótica*, 3ra edición. Prentice Hall. México, pags. 62–134, 2006.

[36] Padierna-Garcia, J. J. *Construcción e Implementación de un Digitalizador Paramétrico Bidimensional con Interfaz ADEFID.* Tesis Licenciatura. Universidad de Guanajuato, 2013.

[37] Gutiérrez-Preciado, A. *Simulación e implementación de un Algoritmo de Planificación de Trayectorias en un manipulador Cartesiano.* Tesis Licenciatura. Universidad de Guanajuato, 2011.

[38] Youtube [en línea] *Planeación de trayectorias continuas en ADEFID*, Disponible en: `https://youtu.be/tXq_0Cqk4yw` (Junio 2016)

[39] GM Factory *GM Is Using the Cloud to Connect Its Factory Robots*, Disponible en: `http://fortune.com/2016/01/30/gm-cloud-factory-robots-fanuc/`