UNIVERSIDAD AUTÓNOMA DE SINALOA

FACULTAD DE INFORMÁTICA CULIACÁN
FACULTAD DE CIENCIAS DE LA TIERRA Y DEL ESPACIO

DOCTORADO EN CIENCIAS DE LA INFORMACIÓN

SISTEMA DE PROGRAMACIÓN FUERA DE LÍNEA PARA ROBOTS DE SOLDADURA BASADO EN UNA INTERFAZ HÁPTICA

T E S I S

QUE COMO REQUISITO PARA OBTENER EL GRADO DE DOCTOR EN CIENCIAS DE LA INFORMACIÓN

PRESENTA:
ÁNGEL SÁNCHEZ DÍAZ

DIRECTOR DE TESIS:
DR. ULISES ZALDÍVAR COLADO

CODIRECTOR DE TESIS:
DR. JOSÉ ALFONSO PÁMANES GARCÍA

CULIACÁN, SINALOA, MÉXICO. ENERO DE 2020

H. COMITÉ ACADÉMICO
POSGRADO EN CIENCIAS DE LA INFORMACIÓN
UNIVERSIDAD AUTÓNOMA DE SINALOA

Presente.

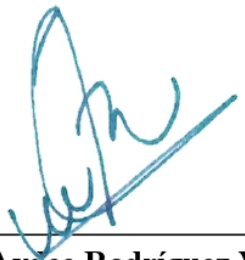
Estimados miembros del H. Comité Académico,

Por este medio me permito saludarlos cordialmente y enviarles mi opinión sobre el trabajo científico del estudiante de Doctorado:

"Ángel Sánchez Díaz"

Después de haber leído y revisado la tesis intitulada **"SISTEMA DE PROGRAMACIÓN FUERA DE LÍNEA PARA ROBOTS DE SOLDADURA BASADO EN UNA INTERFAZ HÁPTICA"**, considero que este trabajo ha dado cumplimiento a los objetivos planteados en el proyecto aprobado inicialmente por el Comité del Programa y reúne los requisitos y méritos suficientes para ser sometido a la presentación pública y evaluación por parte del jurado examinador que se designe.

Atentamente,
Culiacán, Sinaloa a 18 de diciembre de 2019



Dr. Dora Aydee Rodríguez Vega

H. COMITÉ ACADÉMICO
POSGRADO EN CIENCIAS DE LA INFORMACIÓN
UNIVERSIDAD AUTÓNOMA DE SINALOA

Presente.

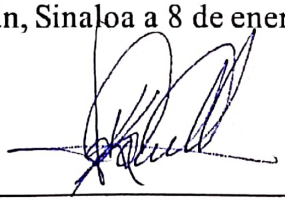
Estimados miembros del H. Comité Académico,

Por este medio me permito saludarlos cordialmente y enviarles mi opinión sobre el trabajo científico del estudiante de Doctorado:

"Ángel Sánchez Díaz"

Después de haber leído y revisado la tesis intitulada **"SISTEMA DE PROGRAMACIÓN FUERA DE LÍNEA PARA ROBOTS DE SOLDADURA BASADO EN UNA INTERFAZ HÁPTICA"**, considero que este trabajo ha dado cumplimiento a los objetivos planteados en el proyecto aprobado inicialmente por el Comité del Programa y reúne los requisitos y méritos suficientes para ser sometido a la presentación pública y evaluación por parte del jurado examinador que se designe.

Atentamente,
Culiacán, Sinaloa a 8 de enero de 2020



Dr. Jesús Roberto Millán Almaraz

H. COMITÉ ACADÉMICO
POSGRADO EN CIENCIAS DE LA INFORMACIÓN
UNIVERSIDAD AUTÓNOMA DE SINALOA

Presente.

Estimados miembros del H. Comité Académico,

Por este medio me permito saludarlos cordialmente y enviarles mi opinión sobre el trabajo científico del estudiante de Doctorado:

"Ángel Sánchez Díaz"

Después de haber leído y revisado la tesis intitulada **"SISTEMA DE PROGRAMACIÓN FUERA DE LÍNEA PARA ROBOTS DE SOLDADURA BASADO EN UNA INTERFAZ HÁPTICA"**, considero que este trabajo ha dado cumplimiento a los objetivos planteados en el proyecto aprobado inicialmente por el Comité del Programa y reúne los requisitos y méritos suficientes para ser sometido a la presentación pública y evaluación por parte del jurado examinador que se designe.

Atentamente,
Culiacán, Sinaloa a 6 de enero de 2020

Dr. Miguel Ángel Llama Leal

Resumen

La programación fuera de línea (PFL) es una alternativa para efectuar una planificación de tareas en robots industriales. Este tipo de programación permite resolver adecuadamente este proceso mediante la simulación tridimensional, solventando las fallas ocasionales antes de que el programa se ejecute en el robot real.

El sistema propuesto en este trabajo está orientado a la PFL para robots de soldadura, que permita la definición de trayectorias de soldadura, a través de la manipulación de una antorcha de soldadura virtual mediante una interfaz háptica, misma que facilita la interacción del usuario con el ambiente virtual, dando al usuario la impresión de tener la antorcha en sus manos, y percibir en ella los contactos con los objetos del ambiente virtual.

Además, se implementa un comportamiento dinámico en el ambiente virtual para evitar que haya un traspaso entre los objetos que lo conforman y ofrecer un comportamiento más realista para el usuario. Asimismo, a la antorcha virtual se le aplica el modelo resorte-amortiguador para calcular las fuerzas que eventualmente son enviadas a la interfaz háptica cuando haya colisiones entre la antorcha y otros objetos virtuales. Finalmente, para validar el sistema propuesto, se presentan resultados experimentales.

Abstract

Offline programming (OLP) is an alternative to carry out a planning of the welding process of industrial robots. This type of programming allows to solve adequately this process through three-dimensional simulation, solving occasional faults before the program is executed in the real robot.

The system proposed in this work is oriented to the OLP for welding robots, allowing the definition of welding paths, through the manipulation of a virtual welding torch via a haptic interface, which facilitates the interaction of the user with the virtual environment, providing the user the sensation of having the torch in his hands, and perceiving in it the contacts with the objects of the virtual environment.

In addition, a dynamic behaviour is implemented in the virtual environment for avoiding the pop-throughs between the virtual objects and offer a more realistic behaviour for the user. Likewise, the spring-damper model is applied to the virtual torch for calculating the forces which are eventually sent to the haptic inteface when collisions between the torch and other virtual objects are detected. Finally, for validating the proposed system, experimental results are shown.

A Dios,
por darme tanto en la vida.

A mi esposa Elvira,
por su entrañable paciencia y su apoyo en todo momento.

A mi hija Elisa,
por su amor incondicional pese a las largas ausencias.

A mi hija en camino,
por ser un motivo más para ser feliz.

A mis padres, Ángel y María del Carmen,
por la confianza y el aliento que siempre me han brindado.

A mis hermanos, Luisa Gpe., Jesús Ignacio, Ma. del Carmen y Martha Alejandra,
por demostrarme que los grandes esfuerzos dan buenos frutos.

A todos ellos, les dedico este trabajo.

Agradecimientos

A la Universidad Autónoma de Sinaloa, por el apoyo brindado durante el doctorado y las facilidades dadas para concluir satisfactoriamente estos estudios.

Al CONACyT, por la beca otorgada en estos cuatro años, que costeó la investigación realizada en este documento.

A mi asesor, Dr. Ulises Zaldívar Colado, por la confianza otorgada y por ayudarme tanto en la realización de esta tesis.

A mi coasesor, Dr. José Alfonso Pámanes García, por sus enseñanzas, así como por su hospitalidad durante mi estancia en el Instituto Tecnológico de la Laguna.

A la Dra. Xiomara Penélope Zaldívar Colado, por todas las gestiones hechas para que la conclusión de mi doctorado se hiciera realidad.

Al M.C. Zeus Del Valle Castillo-Nájera, por ser el vínculo inicial para mi ingreso al doctorado.

A la Dra. Dora Aydee Rodríguez Vega; al Dr. Jesús Roberto Millán Almaraz y al Dr. Miguel Ángel Llama Leal, por ser los revisores críticos de mi tesis.

Al director de la Preparatoria Concordia, Dr. Alejo Armando Valdez Camacho, por las diligencias efectuadas para poder cursar este doctorado.

Al personal administrativo del Posgrado en Ciencias de la Información (Tania e Itzel), por todas las atenciones hacia mi persona.

Contenido

Lista de figuras

Lista de tablas

Lista de ecuaciones

Capítulo 1

Introducción

En este capítulo se define el ámbito en el cual se desarrolla la presente investigación, ofreciendo un panorama de las investigaciones relacionadas a los sistemas de programación fuera de línea de robots y que sientan un antecedente a esta tesis. Asimismo se detalla el problema a resolver, se explica cuál es la justificación para la realización de este trabajo, así como los objetivos que se persiguen para el desarrollo del mismo. Finalmente, se plantea la hipótesis que encauzará esta tesis.

1.1 Contexto de investigación

La inclusión de robots en los procesos industriales se ha vuelto económicamente atractivo desde algunas décadas atrás. Un factor que influye en tal hecho, es que, entre las ventajas de dicha robotización, podemos señalar la reducción del tiempo de producción, así como la mejora de la calidad de las piezas de trabajo.

Tal es el impacto que ha tenido tal robotización en las industrias, que se puede ver un crecimiento en el número de robots en las plantas industriales a nivel mundial. La IFR (International Federation of Robotics) [1] publicó que en 2017 estaban en funcionamiento un poco más de 2 millones de robots en las industrias en todo el orbe y se espera que para 2021 se alcance la cifra de 3 millones. En 2017 se incorporaron a las industrias 381,335 unidades robóticas, de las cuales, aproximadamente el 11% se localizan en América del Norte, y un 15% de esta parte en México. La demanda

por el uso de robots se fundamenta en el potencial que poseen estas máquinas para realizar aplicaciones industriales donde se requiera alta precisión, repetibilidad y estabilidad de operaciones. La industria automotriz opera la mayor cantidad de robots y existe un gran interés en aplicarlos a operaciones de soldadura, montaje y manejo de materiales. En aras de la competitividad en las industrias modernas, la soldadura manual debe limitarse a períodos de tiempo más cortos debido al tiempo de preparación requerido, molestias al operador, consideraciones de seguridad y costos. Por lo tanto, la soldadura robótica es fundamental para la automatización de la soldadura en muchas industrias. Se estima que el 25% de todos los robots industriales se están utilizando para tareas de soldadura [2].

Cabe mencionar que la programación de un sistema robótico no es una labor sencilla. En este contexto, un enfoque que facilita esta labor mediante la planificación de los movimientos del robot usando gráficos de computadora del robot y de su entorno, es la programación fuera de línea (PFL). Este enfoque permite al usuario planear estrategias de manipulación y depurar los programas sin la necesidad de acceder al robot real o a los otros dispositivos de la estación de trabajo. Una vez establecidos los desplazamientos deseados del robot, el usuario puede visualizar una simulación gráfica de sus movimientos. Por lo tanto, la ejecución exitosa de la tarea se puede validar antes de ordenar la acción al robot real [3]. Esta simulación gráfica se lleva a cabo dentro de ambientes virtuales conformados por objetos tridimensionales que son modelados en paquetes de software de diseño asistido por computadora (CAD, por sus siglas en inglés). Estos paquetes son imprescindibles en la PFL, pues gracias a ellos es posible reproducir una presentación visual del robot al realizar sus tareas y eliminar los problemas de alcance, accesibilidad y colisiones que la etapa de planificación de trayectorias conlleva.

Por otro lado, la realidad virtual ha sido desarrollada en los últimos años gracias a tecnologías interesantes [4, 5]. Una de ellas es la háptica, la cual aplica el sentido del tacto para la comunicación con sistemas de cómputo. Una interfaz háptica es aplicada como un dispositivo de entrada/salida, de tal forma que mediante la

activación manual de esta interfaz el usuario puede proveer movimientos a los objetos en un ambiente virtual y recibir respuestas en forma de fuerzas direccionales que resultan de límites sólidos, del peso de los objetos virtuales, vibraciones y de la misma inercia.

1.2 Estado del arte

De acuerdo con Gan [6], en la actualidad existe una gran variedad de software de ambiente gráfico que soportan la PFL, mismos que proceden básicamente de tres fuentes: 1) fabricantes de software genérico de robótica, 2) fabricantes de robots industriales y, por último, 3) instituciones de investigación. Hablando de los fabricantes de software genérico de robótica, se tiene que éstos han desarrollado sistemas tales como Tecnomatix RobCAD [7], Delmia D5/v5 [8] y ACT-Weld [9], entre otros. Particularmente, estos paquetes de software proporcionan un conjunto de herramientas de modelado y simulación capaces de representar gráficamente las tareas ejecutadas por un manipulador robótico dentro de su estación de trabajo. Dichos sistemas tienen la capacidad para programar la mayoría de los tipos de robots, de distintos fabricantes. En el mismo sentido, los fabricantes de robots industriales ofrecen sistemas para PFL diseñados específicamente para sus robots, por ejemplo RobotStudio de la compañía ABB [10], KUKA.Sim Pro de la empresa KUKA [11] y ROBOGUIDE de la firma Fanuc [12]. La compatibilidad de este tipo de software CAD con el hardware del robot es una garantía. Posee funciones para importar modelos tridimensionales de talleres y/o piezas de trabajo para generar las trayectorias deseadas del robot dentro de un espacio de trabajo virtual.

Por su parte, dentro del ámbito científico, algunas instituciones dedicadas a la investigación han reportado un gran número de trabajos relacionados a los sistemas de PFL que sientan un precedente en nuestra investigación. En primer lugar, tenemos al sistema ARMS [13]. En éste se presentan y analizan matemáticamente un conjunto de primitivas de movimientos dentro de un paquete de simulación denominado

GRASP (General Robot Arm Simulation Program) (Figura 1.1) con el fin de diseñar nuevos robots, modificar diseños de robots existentes y evaluar el rendimiento de brazos robóticos a lo largo de posibles rutas, dentro de entornos de trabajo factibles. Las simulaciones se llevan a cabo mediante imágenes del manipulador con las líneas que componen las aristas de los objetos. Este trabajo es uno de los pioneros en la simulación de trayectorias de un brazo robótico, pero sólo se limita a esto, y no efectúa la programación en el robot real.

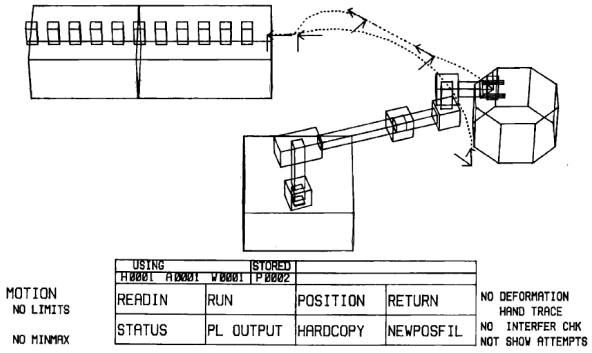


Figura 1.1. Escena de una estación robotizada en el programa de simulación GRASP por Derby (1983). [13]

En [14] se presentó el sistema para PFL llamado ROBOSIM, el cual introduce un punto de vista modular para la PFL de un robot manipulador para tareas de transferencia de piezas (*pick-and-place*). Asimismo, se tiene al paquete SMAR (Système de Modélisation et Animation des Robots) [15], cuya peculiaridad es la construcción tanto de modelos virtuales de múltiples robots como de su entorno, y facilita la verificación de la factibilidad de las tareas a ejecutar a través de funciones para la detección de colisiones, que se emplean en el mismo sistema como rutinas auxiliares en los algoritmos de cálculo del emplazamiento óptimo de robots y de planificación de trayectorias.

Otro enfoque utilizado en este ámbito es la construcción de sistemas para PFL basados en software CAD, es decir, utilizando el entorno operacional, las funciones y el motor gráfico que brinda este tipo de tecnología de ingeniería y modelado tridimensional. Bajo este criterio, en [16] se desarrolló un add-on (extensión) para

SolidWorks™, a través del cual el usuario es capaz de definir las trayectorias de trabajo del robot y organizarlas en la secuencia deseada, para luego generar automáticamente códigos para un robot de soldadura a partir de modelos CAD. En el mismo orden de ideas, en [17] y [18] se utilizó el paquete Autodesk® Inventor® como herramienta para la PFL. Este sistema funciona como una interfaz humano-robot, donde se requieren archivos CAD que contengan la definición y parametrización de la posición/orientación del robot, marcos de referencia y trayectorias de trabajo. Para después, probar y ajustar si es necesario, y convertir esta información en programas para el robot. Algo similar fue reportado por Yin [19], donde se utilizó un método eficiente para la PFL de robots basado en archivos DXF del software AutoCAD®. Aquí, estos archivos son decodificados para obtener la información sobre la posición tridimensional de elementos gráficos tales como puntos, líneas, arcos y curvas, para después transformar estos datos en valores cartesianos y posteriormente convertirlos en comandos robóticos. La trayectoria planeada es verificada con una simulación en OpenGL®. En el mismo orden de ideas, existen trabajos de investigación en PFL donde tanto la trayectoria del robot como los parámetros que definen la posición y orientación del órgano terminal son introducidos al sistema a través de distintos formatos de archivos CAD [20–22]. Para estos trabajos, se crearon programas de simulación que traducían a movimientos del robot los criterios implícitos de los mencionados archivos y de este modo visualizar las trayectorias generadas.

También, la comunidad científica ha explorado la aplicación de software libre para el desarrollo de sistemas para PFL. En [23] se presentó un sistema que permitía la programación y supervisión remota del robot Mitsubishi Movemaster RV-M1. La celda de trabajo se representaba como una sala virtual, donde se podían colocar robots con equipos. La parte de visualización del programa se basó en el estándar de gráficos OpenGL® debido a su implementación en diversas plataformas de sistemas operativos. De igual forma, Cai *et al* [24] presentaron una metodología para la planificación de una ruta continua de la cabeza del láser para un robot de soldadura en el proceso de intersección del corte de línea. La simulación en este sistema se logró utilizando también las librerías gráficas de OpenGL®. Asimismo, en el ya

citado trabajo de Yin [19] la trayectoria planificada se verificaba con la simulación basada en OpenGL®. En la investigación de Ferreira *et al.* [25], los puntos de la ruta de soldadura se extraen del modelo de pieza de trabajo CAD, y la pose de la pieza a soldar proviene de un sistema de visión artificial. Se creó un banco de trabajo personalizado en FreeCAD, un software CAD gratuito y de código abierto.

En México, particularmente en el Instituto Tecnológico de la Laguna (ITLag) se empezaron a desarrollar trabajos en el campo de la simulación gráfica y PFL de robots, manejando prototipos experimentales elaborados en el propio Instituto. El primer paquete desarrollado por esta institución estaba orientado a la operación de uno de tales prototipos: el robot Salviati, de 4 grados de libertad. Se crearon dos versiones de éste y se reportaron en [26] y [27]. En ese paquete, el robot se visualizaba a través de la representación de alambre, mientras se ejecutaban tareas *pick-and-place*. De igual manera, bajo la misma filosofía, se creó un software llamado SISPROS (SIstema de Simulación y Programación del RObot Salviati) orientado a tareas *pick-and-place*, que experimentaba con la programación de movimientos con rutas continuas [28]. También se creó el paquete denominado PLASIRS (PLanificación y SImulación del Robot Sagredo) para la PFL del robot señalado. Contenía funciones para la representación de objetos sólidos [29], además de rutinas que trataban la resolución de la redundancia cinemática mediante la optimización del desempeño del robot [30]. Debido a la modificación del robot Sagredo, el paquete PLASIRS fue actualizado y optimizado [31]. En la versión final de este sistema fueron incluidos procedimientos para el estudio del emplazamiento óptimo de robots redundantes [32].

Ahora bien, debido al carácter general de los sistemas anteriormente mencionados y las limitaciones que éstos presentaban, el ITLag desarrolló un software específicamente para PFL de robots de soldadura de arco cuyo costo resultara atractivo para pequeños usuarios, el cual se nombró como WEROP (acrónimo de WElding Robots Off-line Programming) (Figura 1.2). Este software facilitaba la ubicación del modelo virtual de la antorcha de soldadura del robot en un espacio

tridimensional durante el proceso de enseñanza de las rutas que éste debía seguir para realizar las tareas programadas. El WEROP era un paquete de software programado en Matlab© que integraba los modelos geométricos del robot y de su entorno, diseñados en el paquete SolidWorks™, y aplicaba los patrones simbólicos de robots generados en el paquete SYMORO (SYmbolic MOdeling of RObots) [33]. Cabe señalar, que como parte de los requerimientos del WEROP era necesario que el usuario definiera diversos parámetros referentes al proceso de soldadura; como las coordenadas que definen la posición y los ángulos que precisan la orientación recomendada de la antorcha con respecto a las placas a unir durante la aplicación de los cordones de soldadura, el tipo de cordón y la velocidad de avance de la antorcha [34].

En la primera versión del WEROP, estos parámetros se especificaban de forma numérica en una caja de diálogo (modo explícito). Mientras que en una versión posterior el usuario podía arrastrar esta antorcha en la escena virtual utilizando el mouse para colocarla en la pose deseada (modo interactivo). Resulta importante señalar que estas dos opciones que poseía el WEROP para la especificación de una pose deseada de la antorcha en la estación virtual del robot, implicaban un proceso de establecimiento en primer término de los parámetros de posición y orientación de la antorcha, es decir, tanto el modo explícito como el modo interactivo requerían la determinación previa de una serie de coordenadas operacionales de la antorcha.

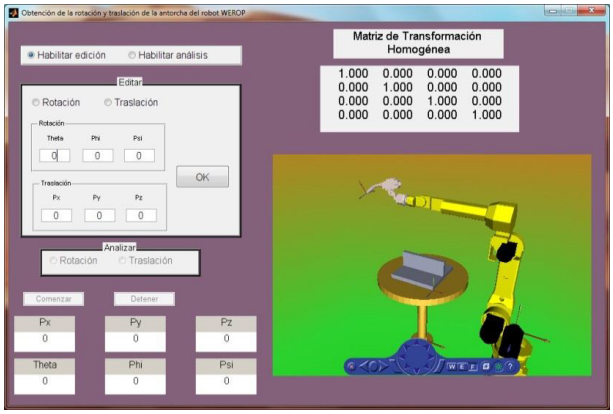


Figura 1.2. Ventana de la interfaz del programa WEROP por Alba (2012) para PFL de robots de soldadura. [34]

Del esfuerzo conjunto del ITLag con la Universidad Autónoma de Sinaloa (UAS), surgió un trabajo de investigación que trata sobre la PFL de robots de soldadura a través de una interfaz háptica [35]. En éste, se destacan las ventajas de utilizar un dispositivo háptico en comparación a las funciones originales que presentaba el WEROP en su primera versión. Se mostró un caso de estudio en el cual se manifestó la eficacia de dicho procedimiento experimental, esto se logró gracias a una interacción natural del usuario con el ambiente virtual. Además se consiguió una disminución en el tiempo de programación de trayectorias del robot. No obstante, quedó como un tema pendiente el análisis formal de la evaluación de la precisión de los puntos de soldadura. Aunado a lo anterior, cabe señalar que existía poco realismo visualmente hablando en la escena virtual, puesto que el usuario al manipular la antorcha virtual mediante la interfaz háptica existían interpenetraciones entre la antorcha y los demás objetos de la escena, cosa que en la realidad no sucede.

Asimismo, se han hecho trabajos incorporando comportamiento dinámico a aplicaciones con interfaces hápticas. Es entonces que el modelo resorte-amortiguador, ha sido utilizado en aplicaciones de realidad virtual, así como en el terreno de la robótica. En [36] se demuestra que el resorte lineal mantiene un comportamiento dinámico más estable que el torsional para el caso de un robot de 3 grados de libertad. En [37] se aplicó tal conocimiento generado para conseguir una interacción realista del robot virtual en un ambiente dinámico. La manipulación del robot por parte del usuario en ambos estudios se llevó a cabo por medio de una interfaz háptica. A su vez, en [38] se utiliza una perspectiva orientada a tareas de ensamble virtual. Cuando las partes entran en contacto, considerándose el modelo resorte-amortiguador, se calcula la fuerza que el usuario sentirá a través de un sistema háptico.

En retrospectiva, mientras que los proveedores de software comercial para PFL enfatizan en hacer que el paquete sea más potente, modular y flexible, los investigadores académicos han puesto atención en mejorar algoritmos de planificación de procesos y han desarrollado algunos paquetes de software PFL buscando hacer más accesibles para las industrias este tipo de sistemas, utilizando software CAD

y tecnología de código abierto. Empero, ante la gran variedad de sistemas para PFL y/o simulación presentados en esta sección, se infiere que persiste un alto grado de especialización por parte del usuario al momento de realizar la PFL de un robot industrial. Asimismo, la dificultad para efectuar tal proceso también se hace presente en los trabajos señalados hasta aquí. Quedando sólo por indicar que aunque existen muchos sistemas especializados en este ámbito, la mayoría de ellos son poco accesibles para pequeños y medianos usuarios de robots; principalmente aquellos desarrollados por fabricantes profesionales de software o por fabricantes de robots.

1.3 Planteamiento del problema

En las industrias, la soldadura sigue siendo en gran medida un proceso realizado por los operadores humanos. No obstante, el uso de robots de soldadura en los procedimientos de fabricación industrial ha tenido un incremento en los últimos tiempos [39]. En consecuencia, esta participación masiva de los robots dentro de las empresas manufactureras ha motivado a algunos investigadores a explorar métodos de programación que contribuyan en el proceso de enseñanza de las trayectorias de robots de soldadura, puesto que una adecuada planificación de dichas trayectorias permite al robot seguir una ruta efectiva y segura en su espacio de trabajo para completar una tarea dada.

Sin embargo, estos esfuerzos de investigación no han podido resolver la complejidad de la programación de un sistema robótico de soldadura. Por un lado, algunos investigadores se han enfocado esencialmente en la programación en línea, la cual se efectúa guiando al robot a través de la ruta deseada y guardando las posiciones deseadas, todo esto usando una consola portátil (teach-pendant). Típicamente, este tipo de programación involucra repeticiones manuales, que son arriesgadas, lentas y disruptivas para el proceso de producción [40]. Por otro lado, están aquellos estudios orientados a la PFL, donde el robot es modelado tridimensionalmente e introducido en un ambiente virtual, por medio del cual el usuario a través de la simulación genera

trayectorias de soldadura sin interferir en el proceso de producción, pero los usuarios deben poseer un alto nivel en habilidades para la programación y tener una base de conocimiento en el manejo de software CAD [7, 14, 16, 21, 41], tener dominio de un lenguaje de alto nivel [42] y la cinemática de manipuladores [16]. Normalmente, las personas que fungen como operadores de los robots en las plantas industriales no disponen de estas habilidades ni tal conocimiento. Resulta claro que hoy en día, la PFL de robots requiere de personal capacitado, sin embargo, la infraestructura para generar personal con las aptitudes necesarias para llevar a cabo esta labor, se encuentra restringida a unas pocas instituciones, por lo que es preciso encontrar nuevas formas de capacitar al personal que sorteen estos inconvenientes. Por otro lado, los paquetes de software del tipo industrial para PFL disponibles en el mercado son demasiado generales con relación a las necesidades específicas de estos mismos usuarios, no cumpliendo a cabalidad para lo que se requieren.

A pesar de las características atractivas de los sistemas para PFL, la interacción del usuario con los entornos virtuales no es lo suficientemente amigable si se aplican interfaces de hardware estándar como un mouse o un joystick. De hecho, la orientación y ubicación de los objetos tridimensionales en dichos entornos podría ser un problema complejo, esto es debido a que los dispositivos mencionados sólo se mueven en 2D. Ante esto, se han buscado alternativas para resolver este inconveniente. Tal es el caso del uso de interfaces hápticas, las cuales proveen una interacción fácil e instintiva del usuario en un ambiente tridimensional [35]. Aunque, los resultados han mostrado muy poco realismo visual en las escenas del mundo virtual, dado que se han observado algunas interpenetraciones entre la antorcha y los otros objetos en la escena virtual.

Un análisis del estado del arte nos ha arrojado que los paquetes de software para PFL ofrecidos por los fabricantes de robots requieren esfuerzo adicional, como la necesidad de que los usuarios estén familiarizados con los sistemas CAD y el modelado completo de los objetos de la estación de trabajo industrial. En el mismo sentido, los sistemas que han intentado incorporar la háptica en la PFL no han puesto atención en

la precisión de la trayectoria de la tarea de soldadura y ofrecen al usuario una perspectiva poco realista del comportamiento de los objetos dentro del ambiente virtual.

1.4 Justificación

La tecnología de la realidad virtual actualmente es empleada como una herramienta de enseñanza en un número de diferentes áreas de aplicación, como la medicina y la aviación [43]. A través de la simulación tridimensional, en un entorno virtual controlado, tanto cirujanos como pilotos son capaces de discernir y resolver, por ejemplo, las situaciones peligrosas y riesgosas, toda vez que las simulaciones son realistas y precisas gracias a su enfoque altamente visual y multisensorial. De ahí, que existan muchos dispositivos de los que echa mano la realidad virtual, como las interfaces hápticas. Éstas hacen experimentar sensaciones al usuario a través del tacto, cuando exista un choque con los objetos del mundo virtual [44]. Estas interfaces permiten una comunicación bimodal con cualquier sistema, esto es, el usuario puede no solamente enviar la información al sistema en cuestión, sino también recibir la información del sistema en forma de una sensación sobre alguna parte del cuerpo.

La utilización de una interfaz háptica en la PFL para robots de soldadura, facilitará la interacción del usuario con el ambiente virtual, haciendo sentir a éste físicamente la antorcha de soldadura en la mano, así como las posibles colisiones de la antorcha con los objetos del entorno virtual. A la par, la precisión de los dispositivos hápticos ha sido probada en el campo médico, a través de procedimientos quirúrgicos asistidos. En dichos procedimientos se han generado trayectorias que aseguran una maniobra segura y cuyos errores son mínimos [43, 45]. Por lo que, el uso de este tipo de retroalimentación, además de contribuir a una mejor percepción que el operador tiene de sus propios movimientos, favorece a una noción más realista del efecto que éstos tendrán en el mundo real. Aunado a esto, la incorporación del comportamiento

dinámico al ambiente virtual mejorará la retroalimentación visual dado que se incorporan factores físicos a los objetos como fricción y peso, de tal modo que existe una detección de colisiones y por ende se evita la interpenetración visual de objetos por lo que brinda un mejor realismo a la escena tridimensional [46].

A su vez, hay varias plataformas de programación para construir sistemas de simulación robótica en tres dimensiones, una de ellas es OpenGL®. En comparación con otras plataformas, OpenGL® es una biblioteca gráfica tridimensional abierta que es independiente del hardware y sistema operativo, y puede ser obtenida de forma gratuita en todas las arquitecturas importantes actuales [47]. Además, esta plataforma tiene capacidades gráficas abundantes y puede renderizar gráficos tridimensionales con una excelente asignación de texturas, efectos especiales y un rendimiento de visualización potente en tiempo real, los cuales facilitan la simulación de robots.

Aunque la incorporación de los robots en procesos industriales se ha vuelto muy atractivo para las empresas manufactureras en la era de la competición global, el carácter complejo y tardado del proceso de programación de un sistema robótico permanece como uno de los mayores desafíos de la implementación. En este contexto, la simulación tridimensional y las técnicas de PFL ofrecen diversas ventajas y surgen como una opción atractiva para definir un proceso integrado de producción industrial. Utilizando la retroalimentación por fuerza del dispositivo háptico, el sistema propuesto podrá hacer de la PFL una tarea menos compleja y menos especializada.

1.5 Objetivos

Objetivo General:

- Desarrollar un sistema de programación fuera de línea que permita la planificación de trayectorias para un robot industrial de soldadura a partir

de una interfaz háptica dentro de un ambiente virtual con comportamiento dinámico.

Objetivos Específicos:

- Crear un ambiente virtual con comportamiento dinámico para la planificación de trayectorias de soldadura a través de la interfaz háptica.
- Comparar precisión entre diferentes configuraciones de antorchas virtuales que consideren el comportamiento dinámico y/o la sensación háptica.
- Desarrollar una rutina para la simulación del robot virtual ejecutando las tareas de soldadura programadas.
- Transferir las consignas articulares generadas fuera de línea a la computadora del robot para la ejecución de las tareas de soldadura programadas en el robot real.

1.6 Hipótesis

La colocación de los puntos de soldadura durante la planificación del proceso de soldadura robótica en un sistema para programación fuera de línea es más precisa cuando se utiliza una antorcha virtual con sensación háptica y comportamiento dinámico a la vez, que cuando se usa una antorcha virtual que no posea al mismo tiempo estas dos características.

Capítulo 2

Marco teórico

El contenido de este capítulo estriba en los fundamentos teóricos necesarios para el desarrollo de la tesis. Comenzando con los conceptos básicos de robótica, pasando por un apartado sobre cinemática con el fin de estudiar la pose de un cuerpo en un espacio tridimensional en el tiempo. También, se trata la aplicación de los robots en la soldadura. De igual forma, se habla sobre los métodos de programación de un sistema robótico. Finalizando con la descripción de la háptica y el comportamiento dinámico de los objetos en un ambiente virtual.

2.1 Robótica

Hoy en día, los robots son los principales protagonistas de la automatización industrial. Estos mecanismos se han convertido en el reemplazo de operadores humanos al ejecutar tareas de una forma más eficiente y segura, reduciendo significativamente la necesidad física e intelectual del hombre. En este contexto un robot es un mecanismo articulado que desempeña labores como la soldadura, transporte de piezas, pintura o el corte de metales (Figura 2.1) e indistintamente toma nombres como robot industrial o manipulador industrial. Una definición más formal la dicta el RIA (Robot Institute of America) cuando expresa que un robot es "un manipulador multifuncional reprogramable diseñado para mover materiales, partes, herramientas o dispositivos especializados a través de movimientos

programados variables para el desempeño de varias tareas" [48].

Por su parte, la robótica se refiere a todas las tecnologías asociadas a los robots. Además, involucra diferentes áreas del conocimiento como la mecánica, matemáticas, electrónica, eléctrica y computación para analizar, investigar y desarrollar robots para el progreso tecnológico de la sociedad.

Figura 2.1. Robot industrial KUKA KR 1000
(Cortesía KUKA Roboter GmbH).

Con relación a los robots industriales, se tiene que éstos están conformados por componentes rígidos llamados eslabones, los cuales están conectados en serie a través de articulaciones. Las articulaciones más comunes en los manipuladores industriales son de revolución (tipo R) y prismática (tipo P). Las del primer tipo permiten un movimiento de rotación relativo entre los eslabones conectados, mientras que las del segundo tipo permiten un movimiento de traslación relativo. Debido a que el robot considerado en la presente tesis posee exclusivamente articulaciones de revolución, durante el desarrollo de la misma, al referir el término articulación se hará pensando en las del tipo R.

Comúnmente, el tipo de robot al que nos referimos está fijo sobre una base emplazada ya sea en el piso, en un muro o en el techo. Si se considera la configuración física de este tipo de robots como una cadena, se tiene que ésta inicia con el eslabón fijo y

termina en el extremo final con el órgano terminal, el cual puede ser una herramienta para manipulación de objetos o para la ejecución de alguna tarea.

Como ya se dijo, una articulación ofrece un movimiento relativo entre dos eslabones del robot, por tal motivo se dice que cada articulación proporciona un cierto grado de libertad (gdl) de movimiento. Generalmente un grado de libertad está asociado a una articulación, de tal forma que la complejidad de los robots puede clasificarse de acuerdo al número total de grados de libertad que poseen.

Se considera que un manipulador está compuesto por $n + 1$ eslabones idealmente rígidos conectados por n articulaciones para formar una cadena cinemática. Por otro lado, las variables articulares son las n variables $q_1, q_2, ..., q_n$ que definen la posición relativa de un eslabón respecto al precedente.

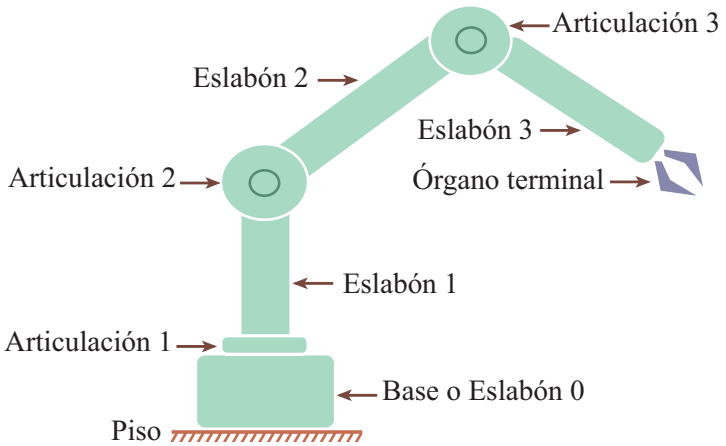


Figura 2.2. Esquema con eslabones y articulaciones de un robot de 3 gdl.

En la Figura 2.2 se muestra un robot manipulador de 3 gdl emplazado en el piso, con una pinza como órgano terminal. Dicha figura también ilustra que cada articulación está conectada a dos eslabones, uno de entrada y otro de salida. A partir de la base, se puede identificar un esquema numérico de eslabones conjuntos. De tal modo que, el eslabón de entrada de la articulación 1 es la base y su eslabón de salida es el eslabón 1, donde éste constituye el enlace de entrada para la articulación 2, mientras que su enlace de salida es el eslabón 2, mismo que continúa la secuencia hacia la articulación 3 conformando el enlace de entrada de ésta, mientras el eslabón 3 es su

enlace de salida.

2.1.1 Posición y orientación

Físicamente, el movimiento que se produce en las articulaciones resulta en un movimiento relativo de los distintos componentes del robot y por lo tanto, en el desplazamiento del órgano terminal. Generalmente los movimientos de los eslabones de un manipulador, incluido el órgano terminal, se componen de una combinación de traslaciones y rotaciones. A la posición y orientación de un objeto en el espacio tridimensional se le denomina pose. En el análisis cinemático de un manipulador se estudia la descripción de las poses de sus eslabones en función de los movimientos de sus articulaciones.

Un marco de referencia o sistema de coordenadas se representa a partir de un punto fijo llamado origen para definir un espacio tridimensional conformado por tres vectores unitarios mutuamente perpendiculares $\{\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}\}$ (Figura 2.3). Simbólicamente un marco de referencia se expresa con la letra griega sigma mayúscula: $\boldsymbol{\Sigma}$. Por ejemplo, un marco de referencia, digamos D se representa como $\boldsymbol{\Sigma_D}$. Para fines prácticos, cualquier descripción de la pose de un objeto se debe hacer con relación a algún marco de referencia. Dicho de otro modo, la posición de dicho objeto se especifica en términos de las coordenadas Cartesianas de un punto del objeto con respecto a los ejes de un marco de referencia. La orientación del mismo objeto se determina de acuerdo a la rotación del objeto con respecto a los ejes del marco de referencia en cuestión.

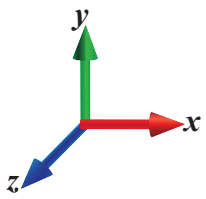


Figura 2.3. Ejemplo de un marco de referencia.

Típicamente se utilizan dos tipos de marcos de referencia: de mano derecha y de

mano izquierda. No existe una convención universal para la cual se deba usar uno o el otro, sin embargo, en robótica es muy común usar el marco de mano derecha. En este contexto se tiene que el marco de referencia de mano derecha determina la dirección de los ejes apuntando el dedo índice de la mano derecha a lo largo del eje $\boldsymbol{x}$ positivo, el dedo medio apuntando en dirección del eje $\boldsymbol{y}$ positivo, y el dedo pulgar apuntará entonces en dirección de $\boldsymbol{z}$ positivo como lo ilustra la Figura 2.4a. De igual forma, si se le aplica una o más rotaciones sucesivas a un marco de este tipo, seguirá manteniendo esta categoría sólo que con una orientación distinta.

También, se elige el marco de mano derecha para definir la dirección de las rotaciones. Convencionalmente, la dirección de la rotación alrededor de un eje en particular se consigue apuntando el dedo pulgar de la mano derecha a lo largo del lado positivo del eje en cuestión, si los dedos se curvan hacia la palma al realizar la rotación, entonces el ángulo θ es positivo, de lo contrario es negativo. Este procedimiento es llamado regla de la mano derecha (Figura 2.4b).

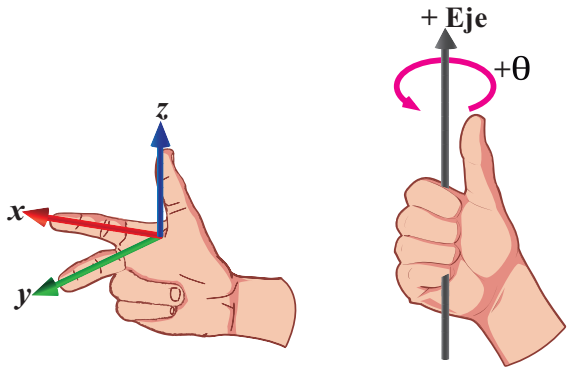


Figura 2.4. Usos de la mano derecha en la robótica

Como los eslabones de un manipulador pueden trasladarse y/o girar con respecto a un marco de referencia, será necesario establecer un marco de referencia ligado a cada eslabón, con su eje $\boldsymbol{z}$ coincidente con el eje de la articulación que conecta a ese eslabón con el eslabón precedente [49]. En la Figura 2.5 se observan dos marcos de referencia, cuyos orígenes coinciden en el punto O. El primero de ellos denominado marco de referencia A ($\boldsymbol{\Sigma_A}$) con $\boldsymbol{x_A}$, $\boldsymbol{y_A}$ y $\boldsymbol{z_A}$ como sus ejes de coordenadas, y el otro

denominado marco de referencia B (Σ_B) cuyos ejes de coordenadas son $\boldsymbol{x_B}$, $\boldsymbol{y_B}$ y $\boldsymbol{z_B}$. De éstos, el sistema de referencia Σ_A está fijo en la superficie de la tierra, mientras el sistema de referencia Σ_B es el ligado a un eslabón prismático cuyas aristas se indican con líneas punteadas.

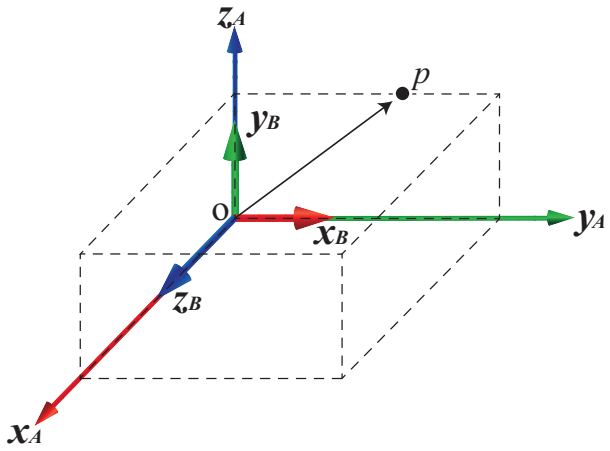


Figura 2.5. Sistemas de coordenadas de referencia fijo (Σ_A) y ligado a un eslabón (Σ_B).

Asimismo, se asume que la posición de un punto p en el espacio puede ser representado por las componentes Cartesianas de sus vectores de posición con respecto a los sistemas de coordenadas Σ_A y Σ_B, respectivamente, como:

$$ {}^{A}\boldsymbol{p} = \begin{bmatrix} p_{x_A} \\ p_{y_A} \\ p_{z_A} \end{bmatrix} \qquad {}^{B}\boldsymbol{p} = \begin{bmatrix} p_{x_B} \\ p_{y_B} \\ p_{z_B} \end{bmatrix} \tag{2.1} $$

(a) **(b)**

Donde ${}^{A}\boldsymbol{p}$ y ${}^{B}\boldsymbol{p}$ representan el mismo punto p en el espacio con respecto a distintos sistemas de coordenadas.

Ahora bien, suponiendo que Σ_B se gira un ángulo θ respecto al eje x_A, se tiene que la orientación de Σ_B con respecto a Σ_A se describe mediante la siguiente matriz:

$$ R(x,\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -sen\theta \\ 0 & sen\theta & cos\theta \end{bmatrix} \tag{2.2} $$

Análogamente, si Σ_B se gira un ángulo θ con respecto al eje y_A entonces la orientación de Σ_B con respecto a Σ_A vendrá representada por la siguiente matriz:

$$R(y,\theta) = \begin{bmatrix} cos\theta & 0 & sen\theta \\ 0 & 1 & 0 \\ -sen\theta & 0 & cos\theta \end{bmatrix} \tag{2.3}$$

En el mismo sentido, suponiendo que Σ_B se gira un ángulo θ con respecto al eje z_A se tiene que Σ_B poseerá una orientación con respecto a Σ_A que se simboliza en la siguiente matriz:

$$R(z,\theta) = \begin{bmatrix} cos\theta & -sen\theta & 0 \\ sen\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

Estas tres matrices de 3 x 3 (Ecuaciones 2.2, 2.3 y 2.4) se denominan matrices de rotación básicas de un espacio tridimensional y facilitan las transformaciones de las componentes de vectores del marco de referencia B al A. Una aplicación trascendental de este tipo de matrices radica en la representación de rotaciones sucesivas, las cuales implican que una rotación arbitraria puede ser representada mediante tres cantidades independientes, que en este caso, son los ángulos que giró de manera sucesiva el marco de referencia con respecto a cada uno de sus ejes.. Un orden en las rotaciones sucesivas muy utilizado es R_x - R_y - R_z, mismo que se le llama rotación de ángulos de Bryant.

Suponiendo que el marco B se gira un ángulo λ respecto a $\boldsymbol{x_B}$, luego se gira un ángulo μ respecto a $\boldsymbol{y_B}$ y finalmente se gira un ángulo ν respecto a $\boldsymbol{z_B}$. Las rotaciones sucesivas que se producen con estos giros se representan como sigue:

$$R_{xyz} = R(x,\lambda)\ R(y,\mu)\ R(z,\nu) \tag{2.5}$$

Aplicando las matrices de transformación básicas representadas por las Ecuaciones 2.2, 2.3 y 2.4, se tiene que la Ecuación 2.5 queda así:

$$R_{xyz} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\lambda & -s\lambda \\ 0 & s\lambda & c\lambda \end{bmatrix} \begin{bmatrix} c\mu & 0 & s\mu \\ 0 & 1 & 0 \\ -s\mu & 0 & c\mu \end{bmatrix} \begin{bmatrix} c\nu & -s\nu & 0 \\ s\nu & c\nu & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_{xyz} = \begin{bmatrix} c\mu c\nu & -c\mu s\nu & s\mu \\ s\lambda s\mu c\nu + c\lambda s\nu & -s\lambda s\mu s\nu + c\lambda c\nu & -s\alpha c\mu \\ -c\lambda s\mu c\nu + s\lambda s\nu & c\lambda s\mu s\nu + s\lambda c\nu & c\lambda c\mu \end{bmatrix} \tag{2.6}$$

Donde $c\lambda$ significa $cos\lambda$; $s\lambda$ representa a $sen\lambda$ y así sucesivamente.

Por su parte, a la matriz de la Ecuación 2.6 se le llama matriz de rotación de los ángulos de Bryant. Asimismo, si se identifica cada elemento de ésta con la nomenclatura t_{fc}, donde f es el número de fila y c el número de la columna tenemos:

$$R_{xyz} = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix} \tag{2.7}$$

A partir de la Ecuación 2.7 se pueden resolver los ángulos λ, μ y ν mediante las siguientes fórmulas:

$$\lambda = atan2(-t_{23}, t_{33}) \tag{2.8a}$$
$$\mu = atan2(t_{13}, t_{33}/c\lambda) \tag{2.8b}$$
$$\nu = atan2(-t_{12}, t_{11}) \tag{2.8c}$$

Es común que se presente un escenario donde $\lambda = 90°$, de tal modo que $cos\lambda = 0$, esto origina que al calcular μ con la Ecuación 2.8b nos lleve a una indeterminación. Con el objetivo de sortear este inconveniente, alternativamente, se puede aplicar la siguiente fórmula:

$$\mu = atan2\,(t_{13}, -t_{23}/s\lambda) \tag{2.8d}$$

Asimismo, se tiene:

$$\mu = atan2\,(t_{13}, t_{11}/c\nu) \tag{2.8e}$$

$$\mu = atan2\,(t_{13}, -t_{12}/s\nu) \tag{2.8f}$$

2.1.2 Transformaciones

Hasta el momento sólo se han propuesto formas de representación aisladas de la posición y la orientación de un sólido en el espacio, esto es, ninguno de los métodos explicados por sí solo permite una representación conjunta de la posición y orientación de objetos rígidos en el espacio. Ante esta situación, deben ser utilizadas las coordenadas homogéneas. Un espacio tridimensional se encuentra representado en coordenadas homogéneas por 4 dimensiones, de tal modo que un vector $p(x, y, z)$ se representa por $p(wx, wy, wz, w)$, donde w tiene un valor arbitrario y se puede ver como un factor de escala.

A partir de las coordenadas homogéneas surge el concepto de matriz de transformación homogénea, la cual para fines prácticos se representará con la letra T y se define como una matriz de 4×4 que representa la transformación de un vector de coordenadas homogéneas de un marco de referencia a otro, estando ambos marcos con orientaciones y posiciones distintas en el espacio. También es conocida como matriz de transformación o matriz homogénea. Por la manera como está definida una matriz homogénea, sus elementos describen la pose de un marco de referencia ortonormal con respecto a otro. Simbólicamente, una matriz homogénea se puede representar como ${}^{i}_{j}\boldsymbol{T}$, que se lee como "matriz T de i a j", lo cual significa que los elementos de dicha matriz describen la posición y orientación de un marco ortonormal j con respecto a un marco ortonormal i.

La composición de una matriz de transformación homogénea ${}^{i}_{j}\boldsymbol{T}$ (Ecuación 2.9) consta de cuatro submatrices de diversos tamaños: $R_{3\times3}$ que corresponde a la matriz de rotación del marco j con respecto al marco i; $p_{3\times1}$ que denota al vector de posición del origen del marco j con respecto al marco i; $f_{1\times3}$ representa la transformación de perspectiva y $w_{1\times1}$ representando el escalado global. En las aplicaciones de las matrices de transformación homogéneas a la cinemática de sólidos, sólo interesa

conocer los valores de $R_{3\times3}$ y de $p_{3\times1}$, dejando en valor nulo los elementos de $f_{1\times3}$, mientras que $w_{1\times1}$ siempre es igual a 1.

$$ {}^{i}_{j}T = \begin{bmatrix} R_{3\times3} & p_{3\times1} \\ f_{1\times3} & w_{1\times1} \end{bmatrix} = \begin{bmatrix} \text{Rotación} & \text{Posición} \\ \text{Perspectiva} & \text{Escalado} \end{bmatrix} \tag{2.9} $$

En este trabajo, las matrices de transformación homogéneas son aplicadas para representar las rotaciones y traslaciones relativas entre los distintos eslabones del manipulador. Para esto, a cada eslabón se le asocia un marco de referencia ligado a él mismo, de tal modo que la matriz de transformación homogénea que representa la relación entre los marcos asociados a dos eslabones, digamos $i-1$ e i, se expresa como ${}^{i-1}_{i}T$. Tomando en cuenta lo anterior, ${}^{0}_{1}T$ describe la posición y orientación del primer eslabón con respecto a la base del robot (eslabón 0); ${}^{1}_{2}T$ representa la posición y orientación del segundo eslabón con respecto al primero y así sucesivamente. Del mismo modo, la posición y orientación del marco de referencia ligado al segundo eslabón del robot con respecto al marco de referencia del eslabón 0 se puede expresar mediante ${}^{0}_{2}T$. Debido a que no corresponde a eslabones consecutivos, la matriz anterior se obtiene mediante el producto de las dos matrices sucesivas ${}^{0}_{1}T$ y ${}^{1}_{2}T$; es decir:

$$ {}^{0}_{2}T = {}^{0}_{1}T \ {}^{1}_{2}T \tag{2.10} $$

Análogamente, si se desea calcular la matriz ${}^{0}_{3}T$ se tiene que:

$$ {}^{0}_{3}T = {}^{0}_{1}T \ {}^{1}_{2}T \ {}^{2}_{3}T \tag{2.11} $$

Asimismo, la inversa de cualquier matriz de transformación homogénea cambia el orden de referencia de los marcos involucrados, por ejemplo, si la matriz ${}^{1}_{2}T$ describe la pose del marco 2 con respecto al marco 1, al aplicarle la inversa a tal matriz se obtiene ${}^{2}_{1}T$ que describe la pose del marco 1 con respecto al marco 2, esto queda expresado como sigue:

$$ {}^{1}_{2}T^{-1} = {}^{2}_{1}T \tag{2.12} $$

Hasta ahora, este capítulo se ha dedicado al estudio de los conceptos básicos de la robótica relativos a la descripción matemática de la pose de un sólido en el espacio tridimensional mediante matrices de transformación homogéneas. Dichas nociones, permitirán adentrarnos en el estudio de la cinemática de los manipuladores.

2.1.3 Cinemática de manipuladores

La cinemática es una rama de la mecánica que estudia el movimiento de los objetos sólidos sin considerar las fuerzas que lo producen, por tanto sólo se considera el análisis de las posiciones y orientaciones de los cuerpos en función del tiempo. Así, en el análisis cinemático de un manipulador robótico se estudia el comportamiento de las poses de los eslabones del manipulador, así como el desempeño de sus velocidades y aceleraciones lineales y angulares, en función del tiempo. Para proceder a dichos estudios, en robótica industrial se parte de la especificación del movimiento de las articulaciones del manipulador, o bien del movimiento deseado del órgano terminal del manipulador al ejecutar alguna tarea en particular.

Teniendo en cuenta lo anterior, en la cinemática de robots se consideran dos problemas fundamentales: la cinemática directa y la cinemática inversa. La resolución del problema de la cinemática directa permite obtener las coordenadas de posición y orientación (llamadas coordenadas operacionales) del órgano terminal del manipulador conociendo las variables articulares o coordenadas generalizadas $(q_1, q_2, ..., q_n)$. En la resolución de la cinemática inversa se obtienen las variables articulares $(q_1, q_2, ..., q_n)$ que permiten alcanzar las coordenadas operacionales especificadas para la ejecución de una tarea en particular. El número n de variables articulares del manipulador es igual al número de grados de libertad de éste. En la Figura 2.6 se representan esquemáticamente las funciones que se realizan tanto en la cinemática directa como en la inversa de un manipulador.

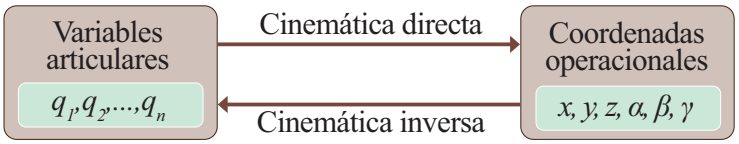


Figura 2.6. Relación entre cinemática directa y cinemática inversa.

Como se hace notar en la imagen previa, existe una relación implícita entre las variables articulares y las coordenadas operacionales. Esta relación depende de la estructura geométrica del robot, por lo que se requiere de una metodología que normalice la descripción de tal estructura, y de esta forma, simplificar la resolución de ambos tipos de cinemática. En esta sección se presenta un procedimiento común para tal propósito: la notación de Denavit-Hartenberg con los parámetros modificados de acuerdo a la propuesta de Khalil y Kleinfinger [50].

2.1.3.1 Convención modificada de Denavit-Hartenberg

En robótica de manipuladores, los parámetros descritos con la notación de Denavit-Hartenberg (D-H) describen con precisión la pose relativa de un eslabón con respecto al precedente en la cadena cinemática de n grados de libertad del manipulador. Los parámetros así definidos permiten obtener las n matrices de transformación homogéneas que describen la pose de cada eslabón móvil con respecto al precedente. El conocimiento de estas matrices facilita la resolución de los problemas de las cinemáticas directa e inversa.

La convención modificada de Denavit-Hartenberg considera un robot compuesto por $n+1$ eslabones, donde el eslabón 0 es la base fija, y el eslabón n es el órgano terminal. Asimismo, la articulación i conecta al eslabón $i-1$ con el eslabón i. Se requiere asignar un marco de referencia ortonormal Σ_i (Figura 2.3) a cada eslabón para definir los parámetros que especifican la pose de cada marco respecto al precedente. Los ejes del *i-ésimo* marco de referencia se definen bajo la siguiente convención:

- En primer término, el eje $\boldsymbol{z_i}$ se supone a lo largo del eje de la articulación i.

- Por otro lado, el eje $\boldsymbol{x_i}$ se define a lo largo de la perpendicular común a $\boldsymbol{z_i}$ y a $\boldsymbol{z_{i-1}}$. Es posible que se dé el caso donde estos dos ejes sean paralelos, por lo que $\boldsymbol{x_i}$ no puede definirse de manera única. Es entonces que se apela a un criterio de simetría o simplicidad para la definición de la ubicación de dicho eje.

- Finalmente, el eje $\boldsymbol{y_i}$ es definido a partir de los ejes $\boldsymbol{x_i}$ y $\boldsymbol{z_i}$ de tal modo que se atienda a los requisitos de un marco de mano derecha (Figura 2.4b).

Un inconveniente de la metodología anterior radica en la imposibilidad de definir en primera instancia los marcos de referencia unidos al eslabón 0 y al eslabón n. Para solucionar tal problema, se recomienda que el marco Σ_0 se defina de tal manera que esté alineado con el marco Σ_1 cuando $q_1 = 0$, de la misma forma, el marco Σ_n se escoge de modo que se encuentre alineado con el marco Σ_{n-1} cuando $q_n = 0$.

Con los marcos de referencia asignados a cada eslabón, se establecen los parámetros geométricos que definen la posición y orientación de cada marco respecto al precedente. Estos parámetros se definen de la manera indicada en la Tabla 1.

Tabla 1. Definición de parámetros de Denavit-Hartenberg modificados.

Parámetro	Definición
α_i	Es el ángulo de z_{i-1} a z_i medido respecto a x_{i-1} conforme a la regla de la mano derecha.
d_i	Es la distancia entre z_{i-1} y z_i a lo largo de x_{i-1}.
θ_i	Es el ángulo de x_{i-1} y x_i medido respecto a z_i conforme a la regla de la mano derecha.
r_i	Es la distancia entre x_{i-1} y x_i a lo largo de z_i.

A partir de los parámetros geométricos definidos de acuerdo a la Tabla 1, es posible establecer una matriz de transformación homogénea general que defina la pose

relativa de los marcos Σ_i y Σ_{i-1}. La conformación de dicha matriz es la siguiente:

$$ {}^{i-1}_{i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & d_i \\ s\theta_i c\alpha_i & c\theta_i c\alpha_i & -s\alpha_i & -r_i s\alpha_i \\ s\theta_i s\alpha_i & c\theta_i s\alpha_i & c\alpha_i & r_i c\alpha_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.13} $$

Donde $c\theta_i$ significa $cos\theta_i$; $s\theta_i$ representa a $sen\theta_i$ y así sucesivamente.

Cuando se sustituyen los parámetros correspondientes en esta matriz, se obtienen n matrices de transformación homogéneas, mismas que se conocen como las matrices elementales del robot.

2.1.3.2 Cinemática directa

Para resolver la cinemática directa de un robot es necesario determinar el vector de coordenadas operacionales $\boldsymbol{p}$ de su órgano terminal en función de sus variables articulares $\boldsymbol{q}$:

$$ \boldsymbol{p} = \boldsymbol{f}(\boldsymbol{q}) \tag{2.14} $$

Donde $\boldsymbol{f}$ es una función vectorial. Con base en la metodología D-H se obtienen las matrices elementales del robot aplicando la Ecuación 2.13; y considerando las propiedades de las matrices de transformación homogéneas, la matriz ${}^{0}_{n}\boldsymbol{T}$ define la pose del órgano terminal del robot con respecto a Σ_0. Esta matriz se obtiene mediante el producto de las n matrices elementales del manipulador, como se muestra en la Ecuación 2.15.

$$ {}^{0}_{n}T = {}^{0}_{1}T \ {}^{1}_{2}T .... \ {}^{n-2}_{n-1}T \ {}^{n-1}_{n}T \tag{2.15} $$

El desarrollo del producto de la Ecuación 2.15 equivale a la resolución de la cinemática directa del robot, que define una matriz de 4×4 que implícitamente contiene las coordenadas operacionales del órgano terminal. Las coordenadas Cartesianas que definen la posición de éste con respecto al marco Σ_0 se localizan en

los tres primeros renglones de la columna 4 (rectángulo verde de la Ecuación 2.16). Por otra parte, la orientación del órgano terminal se obtiene a partir de la submatriz de 3×3, resaltada de color azul en la Ecuación 2.16 utilizando las Ecuaciones 2.8 para determinar los ángulos de Bryant, o las correspondientes fórmulas para calcular los ángulos de Euler.

$$ {}^{0}_{n}T = \begin{bmatrix} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{24} \\ t_{31} & t_{32} & t_{33} & t_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.16} $$

2.1.3.3 Cinemática inversa

En esta sección se explica un método para la identificación del vector $\boldsymbol{f}^{-1}$ (Ecuación 2.17) de las funciones que permiten calcular las variables articulares $\boldsymbol{q}$ en términos de la posición y orientación ($\boldsymbol{p}$) del órgano terminal. Se trata de un método para la resolución del problema de la cinemática inversa y generalmente es más complejo que el problema de cinemática directa:

$$ \boldsymbol{q} = \boldsymbol{f}^{-1}(\boldsymbol{p}) \tag{2.17} $$

En otras palabras, asumiendo que se conoce la matriz ${}^{0}_{n}\boldsymbol{T}$, la cual especifica la pose deseada del órgano terminal del robot, se requiere calcular las variables articulares correspondientes a esa dicha pose. Existen distintos métodos para resolver el problema cinemático inverso, sin embargo, en este trabajo se aplicará el método de Paul [51], puesto que este método resulta apropiado para la resolución de la cinemática inversa de robots de 6 grados de libertad con muñecas esféricas, características que satisface el robot que se aplica en esta tesis.

En primer término, dicha metodología requiere de las matrices elementales, las cuales

permiten obtener sucesivamente las variables articulares a partir de la Ecuación 2.18.

$$U_0 = {}^{0}_{1}T \ {}^{1}_{2}T \ \ldots\ldots {}^{n-2}_{n-1}T \ {}^{n-1}_{n}T \tag{2.18}$$

Se observa que las matrices elementales del lado derecho de la Ecuación 2.18 contienen una variable articular (incógnita) cada una. Por otra parte, los elementos de la matriz U_0, que especifica la pose deseada del órgano terminal con respecto al marco Σ_0, se designan del siguiente modo para conformar la matriz conocida como matriz *snap*:

$$U_0 = \begin{bmatrix} s_x & n_x & a_x & p_x \\ s_y & n_y & a_y & p_y \\ s_z & n_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.19}$$

Si en esta matriz se agrupan los elementos de cada columna en vectores, se tiene que:

$$U_0 = \begin{bmatrix} \boldsymbol{s} & \boldsymbol{n} & \boldsymbol{a} & \boldsymbol{p} \end{bmatrix} \tag{2.20}$$

Mediante un manejo adecuado de la Ecuación 2.18, que se basa en el sucesivo despeje de las matrices elementales del lado derecho de esta ecuación, pasándolas al lado izquierdo, se pueden deducir ecuaciones escalares no triviales, conteniendo como incógnitas las variables articulares del manipulador. El enfoque propuesto por Paul busca simplificar el despeje de cada variable articular mediante el principio que se ilustra a continuación:

- Multiplicar ambos miembros de la Ecuación 2.18 por ${}^{1}_{0}\boldsymbol{T}$:

$${}^{1}_{0}T \ U_0 = {}^{1}_{2}T \ \ldots\ldots {}^{n-2}_{n-1}T \ {}^{n-1}_{n}T \tag{2.21}$$

$$U_1 = {}^{1}_{2}T \ \ldots\ldots {}^{n-2}_{n-1}T \ {}^{n-1}_{n}T \tag{2.22}$$

  En la Ecuación 2.21 se observa que el producto del lado derecho queda en función de las incógnitas q_2, q_3, ..., q_n, mientras que el producto del lado

izquierdo, que se convierte en U_1 (Ecuación 2.22), contiene únicamente a q_1, por lo que dicha incógnita queda "aislada". Entonces, de las 12 ecuaciones escalares no triviales que resultan, se busca una que facilite la determinación de q_1.

- Una vez que se obtuvo la ecuación para resolver la variable articular q_1, se multiplica la Ecuación 2.22 por la inversa de ${}^{1}_{2}\boldsymbol{T}$, de tal manera que se despeje del lado derecho, quedando como única incógnita en el lado izquierdo. De la ecuación matricial que resulte se busca el mínimo número de ecuaciones escalares que facilite la obtención de la función para el cálculo de q_2.

- Continuar con el proceso anterior despejando todas las variables articulares hasta que finalmente la variable q_n se obtiene mediante:

$$U_n = {}^{n-1}_{n}\boldsymbol{T} \tag{2.23}$$

2.2 Robots de soldadura

Dentro de los procesos que los robots industriales han vuelto automáticos, se encuentra la soldadura (Figura 2.7). Este método de fabricación une materiales, usualmente metales, mediante su fundición. Con el vertiginoso avance tanto de la tecnología robótica como de la tecnología de soldadura, el uso de los robots en la industria de soldadura ha crecido enormemente, y su aplicación es un clásico ejemplo de la automatización industrial [52–55].

Actualmente, los robots de soldadura son brazos articulados con una herramienta instalada a su extremo llamado antorcha, la cual actúa sobre los objetos de su entorno a fin de realizar un trabajo en beneficio para el usuario, en este caso, soldar piezas metálicas. A esta operación comúnmente se le denomina tarea de soldadura.

Desde la perspectiva de Pires *et al.*, [56], se considera que cualquier operación de

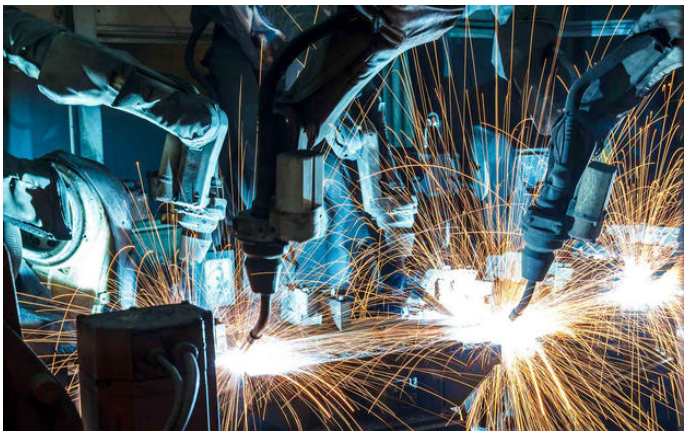


Figura 2.7. Soldadura con robots.

soldadura de forma automática está conformada por tres fases:

1. **Fase de preparación**.

   Donde el operador instala las partes a soldar, los aparatos de soldadura y los parámetros de soldadura.

2. **Fase de soldadura**.

   El sistema debe ser capaz de mantener la orientación de la antorcha mientras se sigue la trayectoria deseada, realice un seguimiento del cordón de soldadura y cambie los parámetros de soldadura en tiempo real.

3. **Fase de análisis**.

   Aquí el operador examina la soldadura obtenida y decide si es aceptable o si se necesitan hacer cambios en algunos parámetros de soldadura.

Un rasgo a considerar en este contexto es el tipo de soldadura por medio del cual el robot lleva a cabo sus tareas. La presente investigación hace referencia a un manipulador industrial con una configuración de soldadura de arco, la cual usa una fuente de poder para crear y mantener un arco eléctrico entre un electrodo y el material base ocasionando que el calor aumente y la punta del alambre de aporte se funda, surgiendo así los puntos de soldadura, que al fin de cuentas, son los que unen a los metales. La región de soldadura regularmente es protegida por algún tipo de

gas inerte, conocido como gas protector, normalmente es tungsteno. Para proveer la energía eléctrica necesaria para los procesos de soldadura de arco, se pueden usar diferentes fuentes de poder. Las más comunes son las fuentes de poder de corriente constante y las fuentes de poder de voltaje constante [57].

La automatización del proceso de soldadura se consigue gracias al manipulador; y al controlador, el cual actúa como el cerebro del robot procesando las consignas dadas al robot esencialmente a través de una botonadura externa (teach-pendant) conectada a éste. La Figura 2.8 muestra esquemáticamente la configuración básica de un sistema robótico de soldadura de arco.

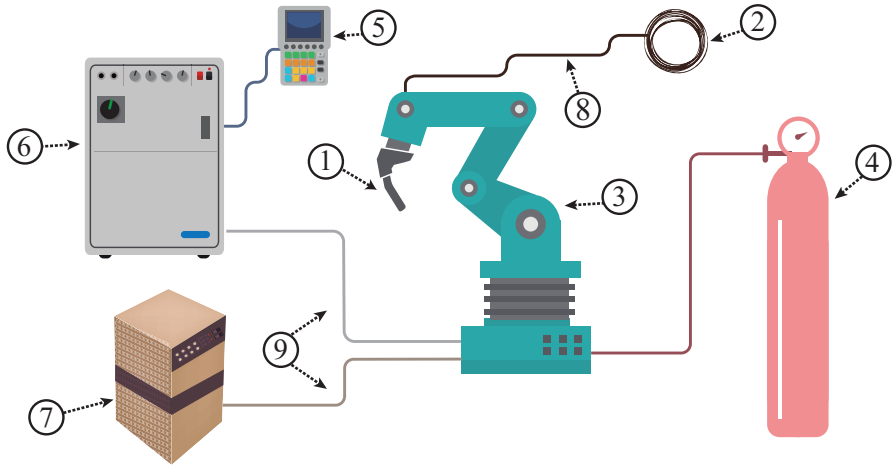


Figura 2.8. Configuración de un sistema robótico de soldadura de arco.
(1) Antorcha de soldadura, (2) Alambre de aporte, (3) Manipulador o robot, (4) Gas inerte, (5) Teach-pendant, (6) Controlador del robot, (7) Controlador del proceso de soldadura, (8) Conducto de alambre aislado, (9) Cables de conexión del robot

Existen estudios que proponen diversas metodologías para el mejoramiento del proceso de soldadura. Se ha experimentado con la integración de sensores al manipulador para obtener los parámetros necesarios y determinar la posición de los puntos de soldadura [53, 58, 59]. Asimismo, se han empleado técnicas de inteligencia artificial para optimizar la trayectoria de soldadura que debe seguir el robot [52, 60]. También, la simulación ha sido parte de los experimentos con robots de soldadura [55, 60, 61].

Adicionalmente a los esfuerzos que se han efectuado en los temas mencionados en

el párrafo precedente para contribuir al avance tecnológico en el área de soldadura robotizada, se busca contribuir a dicho desarrollo en el tema de la programación de ese tipo de robots, como se verá en la siguiente sección.

2.3 Programación de robots de soldadura

Cuando se habla de programación de robots de soldadura se trata de planificar una trayectoria para mover el manipulador desde una posición inicial a alguna posición final deseada según se determine por la antorcha en la parte que necesita ser soldada. Normalmente, el movimiento se especifica asignando una secuencia de puntos, llamados puntos nodo, entre el origen y el destino. Cada uno de estos puntos es un marco que especifica la posición y orientación de la antorcha con respecto al manipulador [54]. Con respecto a esto último, se tiene que, para localizar la antorcha en el espacio se requieren 3 coordenadas Cartesianas para especificar la posición, más 3 ángulos para especificar la orientación, es decir, las coordenadas operacionales. Esta información es utilizada posteriormente como valores de entrada para los cálculos de la cinemática inversa del robot, para determinar así sus configuraciones espaciales que satisfagan las posiciones y orientaciones que, a su vez definan la ruta deseada.

2.3.1 Programación en línea

En los sistemas robóticos de soldadura actuales, su controlador se usa para dirigir y mover al robot durante la soldadura. El método más común de programar una tarea de soldadura es orientar manualmente al robot mediante el teach-pendant, para alcanzar las coordenadas operacionales con las mejores condiciones de soldadura posibles (Figura 2.9). Este proceso de enseñanza, nos produce algunas configuraciones relevantes del robot, mismas que se guardan en un programa dentro del controlador, para posteriormente mandar al robot a moverse a

través de las poses guardadas de la antorcha.

Figura 2.9. *Teach-pendant* de manipulador industrial ABB.

A esta metodología de programación de sistemas robóticos se le denomina programación en línea (Figura 2.10); y aunque el concepto es simple y es ampliamente usada, tiene varios inconvenientes [40]:

1. Utilizar el teach-pendant para el movimiento del robot no es intuitivo, ya que se definen muchos sistemas de coordenadas en un sistema robótico, por lo que el operador siempre debe conocer en cuál marco de coordenadas está el robot cuando se mueva.

2. Guiar al robot a través de una ruta deseada, mientras se evita una colisión con un objeto en el espacio de trabajo, habitualmente es una tarea muy difícil y tardada, específicamente cuando la pieza a soldar tiene una geometría compleja o el proceso en sí es muy complicado.

3. Cuando se genera un programa, se tienen que hacer una gran cantidad de pruebas antes que éste sea satisfactorio en términos de confiabilidad y seguridad.

4. El programa que crea el operador carece de flexibilidad y reusabilidad, ya que una vez generado es muy difícil hacer correcciones posteriormente.

5. El proceso tedioso de programación se tiene que repetir una vez más para una pieza a soldar con una ligera diferencia.

6. El robot no se puede utilizar para producción durante el periodo de enseñanza.

Figura 2.10. Operador realiza programación en línea.

Como se ha dicho, los sistemas robóticos en los procesos de soldadura no sólo mejoran la eficiencia de la producción y las condiciones de trabajo, sino que también realizan la sistematización de la soldadura en el proceso de producción. Sin embargo, el método de programación para robots que se utilizan en la industria de la soldadura descrito previamente, requiere de un trabajo adicional para garantizar que funcionen con la calidad requerida.

En la actualidad, la programación en línea es ampliamente usada en líneas de producción, es un método sencillo y productivo, pero toma mucho tiempo a la hora de ejecutarse [62]. Asimismo, cuando se utiliza esta metodología, debido a la percepción que tiene el operador de la disposición física de los elementos de la estación de trabajo, éste toma sus decisiones a partir de una apreciación visual, y no realiza una evaluación objetiva del desempeño cinemático que tendrá el robot al ejecutar su tarea, por lo que se corre el riesgo de que, al ejecutar la tarea de soldadura, la productividad y la eficiencia de los movimientos del robot no sean los más convenientes. En un estudio efectuado en una aplicación industrial específica, en efecto, se observaron resultados en ese sentido [63].

2.3.2 Programación fuera de línea

Otra metodología empleada para la planificación de trayectorias en robots de soldadura es la programación fuera de línea (PFL), que es una tecnología clave para mejorar la flexibilidad de esta clase de robots y generar trayectorias más complejas [64].

La PFL ofrece algunas ventajas tales como código reusable, flexibilidad para modificaciones y disminución en el tiempo de funcionamiento del sistema durante la etapa de programación. Además, al reprogramar las tareas de un robot que forma parte de un proceso de fabricación, éste sólo tendría que interrumpirse brevemente, mientras los nuevos programas son descargados en las computadoras de control de las estaciones de trabajo. De esta manera se puede lograr un gran ahorro de tiempo y reducción en el costo de la operación.

En la PFL el usuario dispone en una computadora, de modelos virtuales del robot y de su entorno, geométricamente equivalentes al robot y su entorno reales. Después de que los movimientos del robot han sido establecidos y programados, es posible visualizar la animación del robot ejecutando la tarea, con lo que se comprueba en la computadora la realización exitosa de los movimientos proyectados antes de su ejecución en el robot real [14, 17, 18, 34, 61, 65, 66]. Todas estas operaciones se efectúan dentro de un ambiente interactivo, permitiendo la modificación del entorno virtual del robot y así facilitar la evaluación real y la eventual optimización del desempeño cinemático del manipulador.

Comparado con la metodología de la programación en línea, la PFL es un modo más efectivo y eficiente para planificar y controlar los movimientos del robot para muchas aplicaciones industriales, especialmente para aquellas tareas con superficies complejas o con trayectorias complicadas. Un sistema para PFL permite a los programadores simular la producción primero en un entorno gráfico visual y luego operar un robot real para implementar el movimiento físico mediante el controlador del mismo [6, 67, 68].

Llegados a este punto, la PFL provee una solución completa para la programación de un sistema robótico, a partir de procedimientos de simulación y optimización de trayectorias. Una metodología simple de PFL se muestra en la Figura 2.11, y los detalles de cada paso se expresan en los siguientes párrafos:

1. **Modelos CAD**.

   El primer paso es obtener el modelo geométrico tridimensional de cada objeto. Generalmente el enfoque más común para este tipo de modelos es usar software comercial CAD como Catia© y SolidWorks™ (Dassault Systèmes®), AutoCAD® (Autodesk®) o Pro/Engineer® (Parametric Technology Corporation®), sólo por nombrar algunos.

2. **Desarrollo de la escena virtual**.

   Ya que se cuenta con los modelos CAD de los objetos, es necesario realizar una representación gráfica tridimensional a partir de una disposición virtual de tales objetos que contenga toda la información importante para el proceso de programación, por ejemplo, la longitud de la herramienta del robot, así como su emplazamiento dentro de la escena y el posicionamiento relativo de todos los objetos tienen que representar con precisión el entorno real.

3. **Planificación de trayectorias**.

   Consiste en la creación de puntos nodo que definen las trayectorias deseadas del robot para ejecutar tareas. Aún más, constituye una serie de instrucciones que definen poses del órgano terminal y especifican acciones que toman lugar en cada uno de los puntos nodo creados. La idea básica de planificar trayectorias es calcular los movimientos que el robot debe realizar partiendo de un punto origen y terminando en un punto destino, evitando colisiones con objetos.

4. **Simulación**.

   Tiene como objetivo una verificación visual de los movimientos del robot generados por la PFL. Los resultados de la simulación pueden indicar anomalías.

Esto permite probar medios alternativos para realizar la tarea prevista. Cambios frecuentes en el plan, y posibilidades de un enrutamiento alternativo y aspectos de seguridad operacional se pueden verificar a través de la simulación. Aquí, la tarea de simulación gráfica se hace construyendo cuadros visuales correspondiente al cambio secuencial en la configuración del robot, y desplegando los cuadros en la misma secuencia. Así un efecto de animación se crea.

5. **Calibración**.

   Las simulaciones pueden funcionar bien como tales, pero a menudo se comportan de manera indeseada en las condiciones inciertas, impredecibles y generalmente diferentes que se encuentran en la estación de trabajo. Para resolver estas situaciones, se debe completar un proceso de calibración para tomar en cuenta los errores de ubicación de los objetos reales y reubicar los objetos virtuales antes de comenzar la ejecución de la tarea [69].

6. **Programa del robot**.

   Una vez que se verificó que la tarea programada es accesible es tiempo de construir el código del programa de movimientos del robot. Usando la información obtenida de las fases anteriores, el sistema es capaz de crear secuencias de control para el programa del robot. La generación automática del programa del robot no es más que escribir los comandos del robot en un archivo de texto, línea por línea.

7. **Ejecución de la tarea**.

   Después de la calibración, el programa del robot puede ser probado en el manipulador real, pues será usado para el proceso de soldadura.

Es justo señalar que el proceso de la Figura 2.11 regirá el curso del documento para poder cumplir los objetivos que se plantean en este trabajo. Para fines prácticos en la redacción, este proceso en lo sucesivo será nombrado como proceso de PFL.

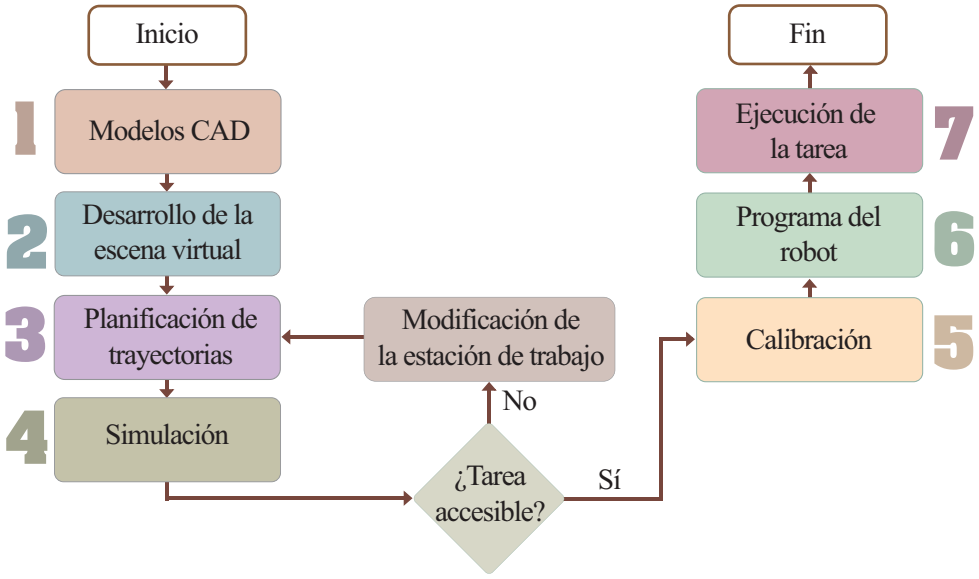


Figura 2.11. Proceso general de PFL.

2.4 Háptica

La realidad virtual, como una tecnología de interacción en ambientes tridimensionales, en los años recientes, ha evolucionado rápidamente en muchas clases de aplicaciones. Una aplicación específica de la realidad virtual es la háptica, que ha sido objeto de estudio en varios ámbitos, desde la ingeniería automotriz a la industria aeroespacial [70], de la medicina [43, 45] al servicio [71], de la educación [72] al adiestramiento [73, 74], del entretenimiento [75] a la industria [76], etc.

La háptica es una tecnología que busca aplicar el sentido del tacto a la interacción humana con sistemas informáticos, permitiéndoles sentir de forma real en sus manos la superficie de los objetos virtuales. Esta tecnología en evolución ofrece un enfoque revolucionario para una interacción realista en ambientes virtuales, puesto que puede incluir simultáneamente dispositivos tanto de retroalimentación de fuerza como de retroalimentación táctil. La retroalimentación de fuerza considera situaciones como simular la dureza de un objeto, el peso y la inercia; mientras la retroalimentación táctil, en circunstancias como simular la superficie geométrica, suavidad, deslizamiento y temperatura [44]. Un usuario puede sorprenderse de cuánto más satisfactoria y atractiva puede ser la interacción cuando más de un

sentido es involucrado.

Un dispositivo háptico es el que involucra el contacto físico entre la computadora y el usuario, por lo general mediante un dispositivo de entrada/salida, como una palanca de mando o guantes, que permiten transmitir los movimientos del cuerpo [77].

Históricamente, la interacción entre los usuarios y la computadora ha sido unidireccional. Por ejemplo, la información visual y auditiva se envían de la computadora al usuario. Con el teclado y el mouse, el usuario introduce información a la máquina. Por su parte la interacción háptica es diferente en el sentido que la energía física fluye bidireccionalmente desde y hacia el operador. El usuario puede no sólo enviar la información a la computadora, sino recibir la información de la computadora en forma de una sensación sobre alguna parte del cuerpo, bien en un único punto con los dispositivos manipuladores, en una mano, con los ciberguantes o bien en todo el cuerpo con los exoesqueletos.

Las interfaces hápticas son esencialmente pequeños robots que intercambian energía mecánica con los usuarios. Desde una perspectiva de hardware, un dispositivo háptico tiene uno o más transductores de entrada (sensores que miden la posición y/o las fuerzas de contacto de cualquier parte del cuerpo humano) y al menos un transductor de salida (actuador que despliega las fuerzas de contacto y las posiciones en una coordinación temporal y espacial apropiada para el usuario) [78]. Ejemplos de interfaces hápticas se pueden ver en la Figura 2.12.

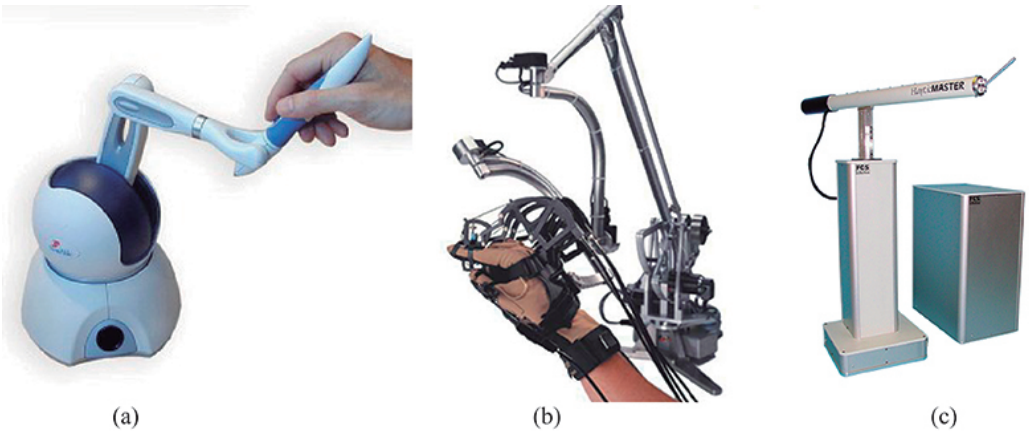


Figura 2.12. Ejemplos de interfaces hápticas.
(a) Phantom OMNI™, (b) Immersion's CyberForce®, (c) FCS HapticMASTER

2.5 Comportamiento dinámico

Uno de los principales inconvenientes en un ambiente virtual, es el hecho de que los objetos que lo componen no poseen las mismas propiedades físicas (peso, inercia, fricción, etc.) que tienen los objetos en la vida real, en consecuencia, un entorno virtual no puede restringir físicamente el movimiento real de los objetos como lo haría un entorno real. Lo anterior ocasiona un movimiento de los objetos poco realista y distrae visualmente al usuario en el momento que los modelos de los objetos traspasan otros objetos virtuales. La retroalimentación visual se puede mejorar manteniendo los modelos de los objetos virtuales fuera de los objetos que contactan. El uso de comportamiento dinámico en un ambiente virtual permite una simulación más apegada a la realidad, evitando la interpenetración visual entre objetos.

El comportamiento dinámico se obtiene por la integración de un motor de modelado basado en física o motor basado en física. Los más comunes en el mercado son AGEIA PhysX©, Open Dynamics Engine™(ODE), Havok® y Bullet© entre otros. Un motor basado en física es un programa de computadora que simula los modelos de física newtonianos, para predecir lo que sucederá en el mundo real en una situación específica.

Como una forma de obtener un comportamiento dinámico en un entorno virtual, se tiene a una técnica basada en el modelo resorte-amortiguador (MRA), la cual fue propuesta originalmente por Colgate en [79]. Esta técnica se fundamenta en la creación de dos modelos virtuales para cada objeto: el objeto cinemático (OC) y el objeto dinámico (OD). Estos modelos manifiestan un acoplamiento artificial entre sí mediante el uso de resortes-amortiguadores virtuales, de tal forma que durante los movimientos del objeto, el OD sigue la posición y orientación del OC. El OD es un modelo tridimensional del objeto renderizado en la escena virtual, el cual imita el comportamiento dinámico del objeto real durante su actividad. El OC también es un modelo renderizado, pero "fuera de pantalla", utilizado para calcular la fuerza de los resortes-amortiguadores [46, 80].

Las características del comportamiento dinámico en un ambiente virtual son las siguientes:

- Detección de colisiones entre objetos en tiempo real.
- No permite la interpenetración visual entre los objetos.
- Interacción dinámica entre las partes, el usuario y otros objetos en el entorno.
- Restricciones físicas para objetos.
- Simulación de movimientos reales de los objetos.

Capítulo 3

Modelado del ambiente virtual

La primer parte de este capítulo gira en torno al desarrollo del ambiente virtual y la incorporación de objetos al mismo; mientras que la parte final trata sobre la integración de la interfaz háptica al sistema para maniobrar la antorcha virtual; la metodología para enviar fuerzas a tal interfaz en caso de contacto de la antorcha con los demás objetos virtuales y la definición de los puntos nodo que conforman la trayectoria a seguir por el robot.

3.1 Arquitecturas de hardware y software

Para esta investigación el hardware utilizado está constituido por una computadora con un procesador Intel Core i5 con 2 núcleos, 2.30 GHz, 7.84 GB de memoria RAM y una tarjeta de video Nvidia® GeForce GTX 950M, con 2 GB de RAM para video. Para la interacción del usuario con el ambiente virtual, se propone aplicar una interfaz háptica llamada Geomagic® Touch™.

Por su parte, la arquitectura de software es presentada en la Figura 3.1, dentro de la cual se advierte que para el modelado del ambiente virtual se requiere de las bibliotecas de renderizado gráfico OpenGL®, mismas que serán utilizadas en la plataforma de programación Visual C++. El modelado geométrico de los objetos virtuales es realizado con un software CAD, en este caso SolidWorks™. Mientras que los archivos generados en este software son convertidos a un formato de mallas poligonales a través del paquete Deep Exploration™ para que posteriormente sean

importados al ambiente virtual.

Para manejar la antorcha virtual dentro de la escena, se logra a través de la operación del dispositivo háptico señalado previamente y su programación se efectúa mediante el software OpenHaptics®, puesto que este programa dispone de los modelos cinemáticos del dispositivo, de esta manera se pueden obtener las coordenadas geométricas del órgano terminal del dispositivo háptico; es decir, su posición y orientación. Dichas coordenadas son aplicadas como una transformación al modelo virtual de la antorcha, emparejando así su movimiento con el del dispositivo háptico.

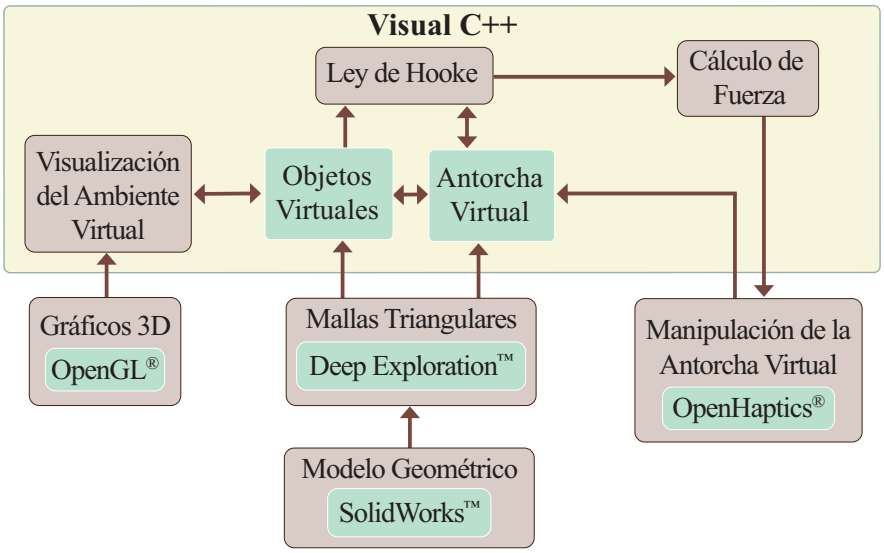


Figura 3.1. Arquitectura de software del ambiente virtual.

3.2 Modelado de objetos virtuales

La PFL principia con el diseño de los objetos tridimensionales que forman parte de la escena virtual, esto implica la creación de modelos geométricos de dichos objetos. El enfoque utilizado en esta investigación para la generación de tales modelos, nos conduce al manejo del software CAD llamado SolidWorks™. Este software es una herramienta de modelado sólido, que facilita el diseño mecánico a través de una interfaz gráfica de usuario, para el sistema operativo Windows®, y para nuestros fines, permite que el objeto en turno sea construido virtualmente, y que se genere

un archivo en formato propio de dicho software. Específicamente, en esta sección da principio el proceso de PFL de esta tesis, puesto que los archivos generados en SolidWorks™ corresponden a los modelos CAD referidos en dicho proceso.

A efecto de entender al modelado geométrico, es necesario ver al modelo como un conjunto de formas geométricas cuya representación matemática es almacenada en la memoria de la computadora. Estos modelos tridimensionales están constituidos por un determinado número de puntos que pueden definirse mediante coordenadas Cartesianas $\{x, y, z\}$.

Vale la pena advertir que los archivos CAD que resultan de esta sección cumplen fielmente con las medidas que se indican en los distintos planos que aparecen a lo largo de la misma. De igual forma, la posición y orientación de los sistemas de coordenadas de cada uno de los modelos de los objetos virtuales son determinados conforme a lo establecido en estos planos, mediante la edición de los archivos en SolidWorks™. Para los intereses de esta tesis, existen tres objetos que primeramente se requieren modelar en este software:

1. **Antorcha de soldadura**.

   Ésta se moverá de acuerdo al movimiento del dispositivo háptico y representa la presencia del usuario en la escena virtual (Figura 3.2).

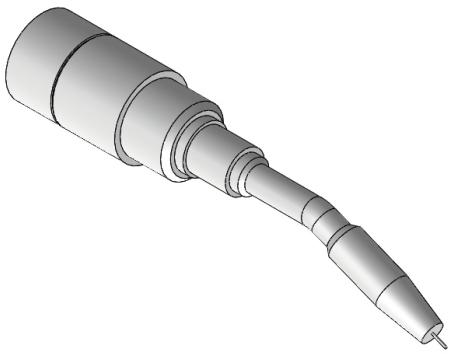


Figura 3.2. Modelo de la antorcha hecho en SolidWorks™.
(*Vista isométrica*)

2. Placas a soldar.

Son dos placas que permanecen estáticas y que deberán soldarse mediante una junta en T (Figura 3.3).

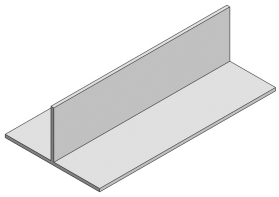


Figura 3.3. Modelo de las placas a soldar hecho en SolidWorks™.
(*Vista isométrica*)

3. Mesa de trabajo.

Es estática y es donde se pondrán las placas a soldar (Figura 3.4).

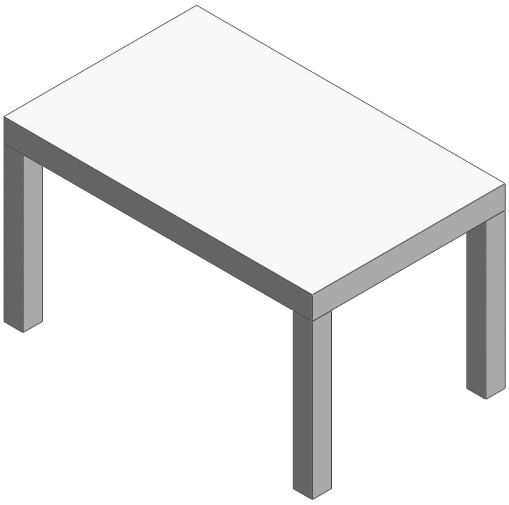


Figura 3.4. Modelo de la mesa de trabajo hecho en SolidWorks™.
(*Vista isométrica*)

En el diseño de estos objetos tridimensionales se ha considerado que sus dimensiones sean iguales a los objetos reales, de esta forma existe una noción más apegada a la apariencia real de la estación de trabajo. En las Figuras 3.5, 3.7 y 3.8 se esquematizan las dimensiones generales de la antorcha, las placas y la mesa respectivamente. En tales figuras se visualizan las aristas de los modelos

correspondientes. Por otro lado, en la Tabla 2 se muestran las medidas de estos objetos virtuales con base en los esquemas de las mencionadas figuras.

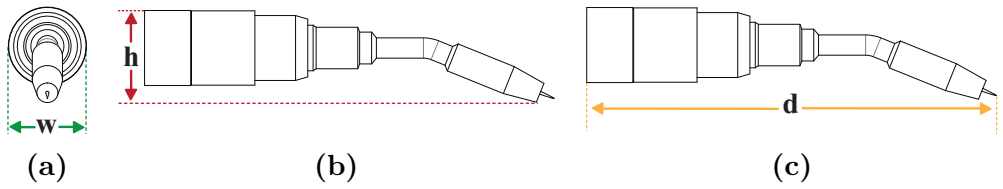


Figura 3.5. Planos con las dimensiones generales de la antorcha virtual ((a) en *vista derecha*; (b) y (c) en *vista frontal*).

El modelo de antorcha empleado es el Abirob R Serie A 500 22° de la marca Abicor Binzel®, cuyo croquis de dimensiones se presenta en la Fig. 3.6, mismo que fue obtenido de la ficha técnica provista por el fabricante.

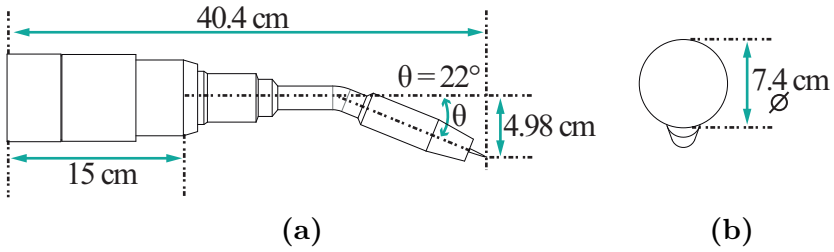


Figura 3.6. Planos con dimensiones específicas de la antorcha virtual ((a) en *vista frontal* y (b) en *vista izquierda*).

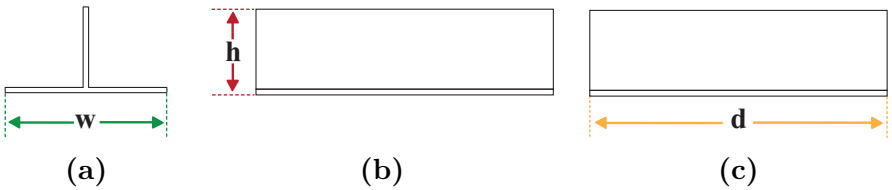


Figura 3.7. Planos con las dimensiones generales de las placas virtuales ((a) en *vista frontal*; (b) y (c) en *vista derecha*).

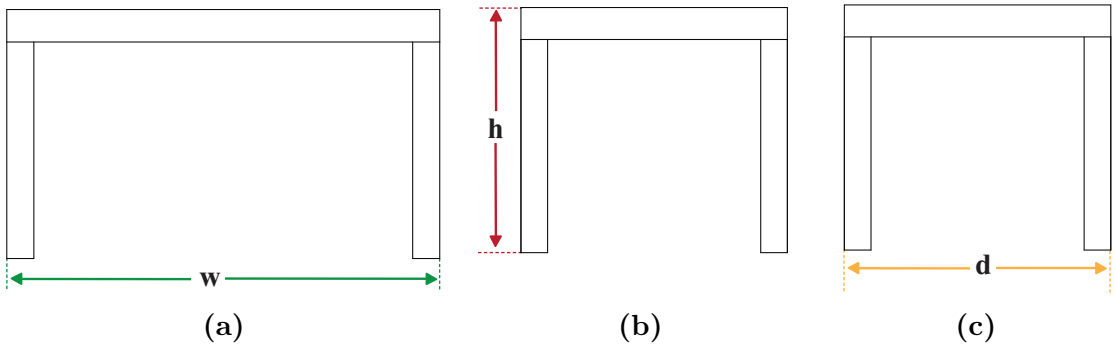


Figura 3.8. Planos con las dimensiones generales de la mesa virtual ((a) en *vista frontal*; (b) y (c) en *vista derecha*).

Tabla 2. Dimensiones en *cm* de los objetos virtuales.

Objeto	w	h	d
Antorcha	7.40	9.05	40.40
Placas	10.20	5.43	19.00
Mesa	80.00	46.00	50.00

También, a cada objeto se le asignó un marco de referencia ligado al propio objeto, el cual representa el origen del mismo, de tal forma que, la posición relativa de un objeto con respecto a otro dentro del ambiente virtual es posible calcularla en cualquier momento. Es así, que la definición de la posición y la orientación de los marcos de referencia Σ_P y Σ_M, correspondientes a las placas y la mesa respectivamente, se hizo de forma arbitraria, mientras que en el caso de la antorcha, la posición de su marco de referencia Σ_H se eligió de acuerdo a los movimientos que ésta presentará dentro de la escena virtual basado en la manipulación de la interfaz háptica. Por su parte, la orientación de este marco corresponde a la definida en la metodología de Denavit-Hartenberg para el robot (Sección 4.1.1). Los marcos de todos los modelos geométricos requeridos en el sistema respetan las pautas de los sistemas de coordenadas de mano derecha.

En primera instancia, se tiene que, para ilustrar la posición de Σ_H, Σ_P y Σ_M dentro de sus correspondientes modelos geométricos se ha considerado un sistema

de coordenadas auxiliar para cada modelo: Σ_{OH}, Σ_{OP} y Σ_{OM} respectivamente. Cada uno de estos marcos alternos es establecido en el límite izquierdo de su respectivo modelo cuando éste se visualiza en vista derecha, definiendo arbitrariamente a x_{OH}, x_{OP} y x_{OM} apuntando a la derecha y a y_{OH}, y_{OP} e y_{OM} hacia arriba. Ahora bien, si se observa al modelo en vista frontal, se considera la posición de este marco alternativo en el límite izquierdo de cada modelo de tal forma que z_{OH}, z_{OP} y z_{OM} apunte a la derecha, quedando conformado un marco de mano derecha. Con esta configuración de marco de referencia, las distancias de algún punto dentro del modelo correspondiente con respecto a Σ_{OH}, Σ_{OP} y Σ_{OM} son siempre mayor o igual a cero.

A partir de lo descrito anterior, se tiene que SolidWorks™ brinda facilidades para hacer una medición más precisa de las dimensiones implicadas para la localización de los marcos auxiliares mencionados. En primer lugar, se procedió a encontrar la posición de la punta de la antorcha, puesto que ésta fungirá como el punto de manipulación de la misma dentro de la escena virtual y por ende ahí se pretende localizar el marco de referencia de la antorcha, llamado Σ_H. Las Figuras 3.9 y 3.10 ilustran la posición de dicho marco con respecto a Σ_{OH}. Es pertinente mencionar que en el caso de la antorcha, el marco Σ_H ha de fungir como punto de pivote al momento de que el usuario realice las maniobras con la interfaz háptica, es decir, las rotaciones aplicadas a la interfaz háptica se verán reflejadas a partir de la punta de la antorcha virtual.

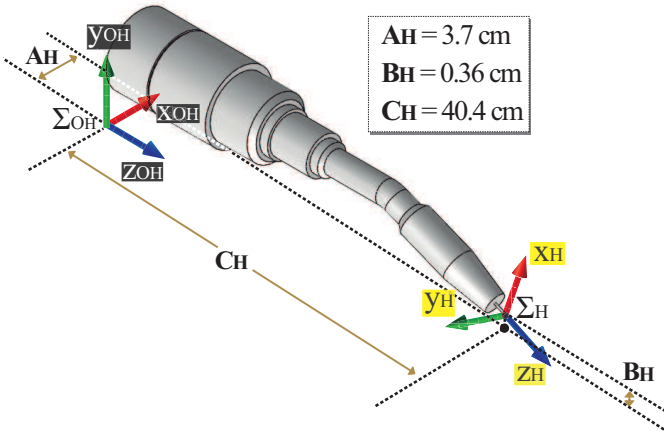


Figura 3.9. Σ_H en el modelo de la antorcha.
(*Vista isométrica*)

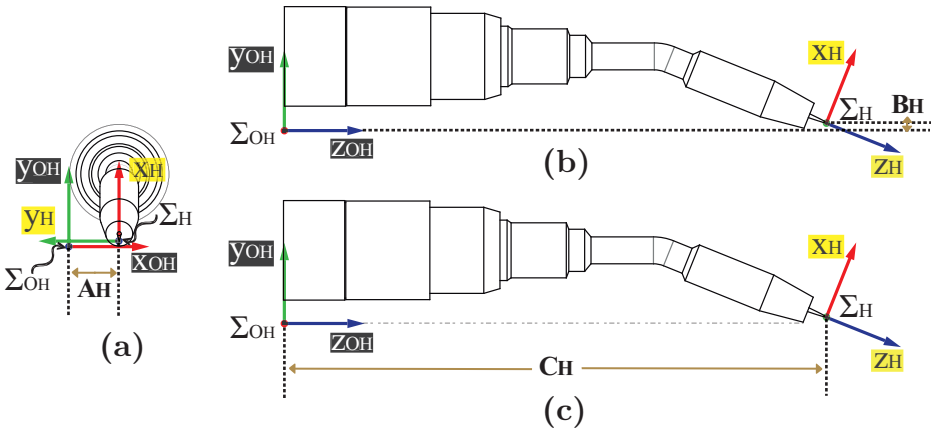


Figura 3.10. Distancias de Σ_H con respecto al marco auxiliar del modelo. ((a) en *vista derecha*; (b) y (c) en *vista frontal*)

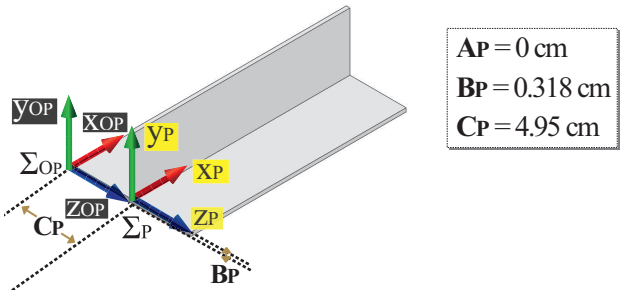


Figura 3.11. Σ_P en el modelo de las placas a soldar. (*Vista isométrica*)

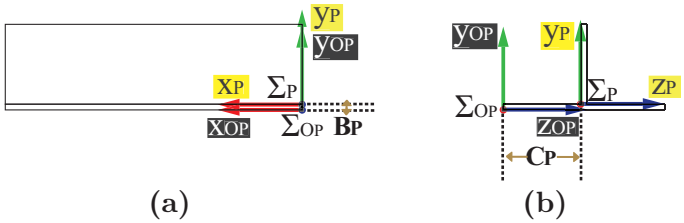


Figura 3.12. Distancias de Σ_P con respecto al marco auxiliar del modelo. ((a) en *vista izquierda*; (b) en *vista frontal*)

Por su parte, en la Figura 3.11 se aprecia la ubicación del marco de referencia Σ_P y se especifican las medidas de este marco con respecto al marco auxiliar del modelo (Σ_{OP}). En la misma tónica, en la Figura 3.12 se representa la posición del marco de referencia de las placas con respecto al marco auxiliar del modelo en planos de

dos dimensiones. Para finalizar, el modelo de la mesa con su respectivo marco de referencia Σ_M y la distancia de éste con respecto al marco auxiliar del modelo se exponen en las Figuras 3.13 y 3.14.

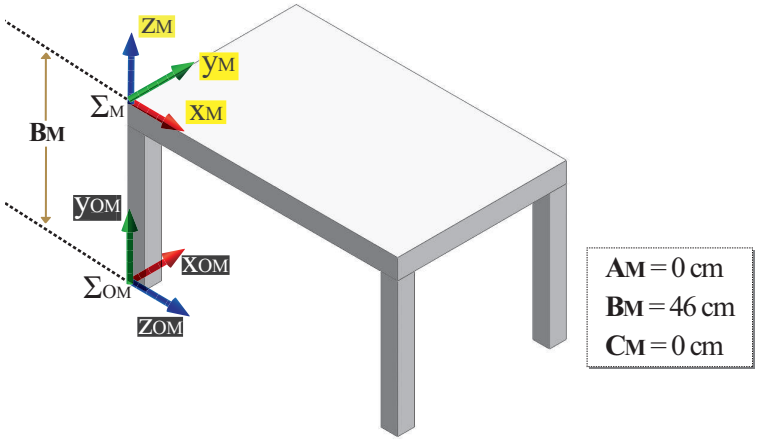


Figura 3.13. Σ_M en el modelo de la mesa de trabajo.
(*Vista isométrica*)

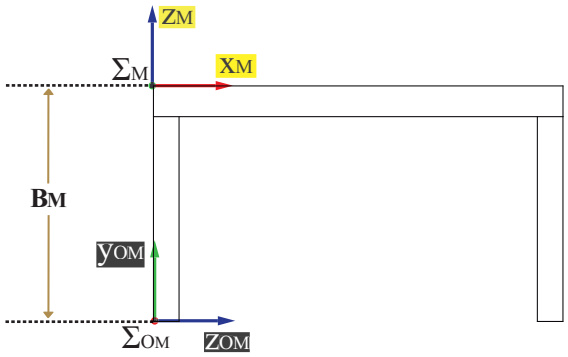


Figura 3.14. Distancia de Σ_M con respecto al marco auxiliar del modelo.
(*Vista frontal*)

3.3 Creación del ambiente virtual

El ambiente virtual o escena virtual es un mundo tridimensional generado por computadora en donde el usuario reside e interactúa a través de distintas interfaces [76]. Asimismo, la utilización de ambientes virtuales representa un área de la

computación que se combina con otras para la creación de presentaciones realistas del mundo real [81].

En lo particular, en la presente sección se detalla la segunda etapa del proceso de PFL, la cual gira en torno del desarrollo de la escena virtual. En esta etapa, se creó un ambiente virtual que facilite la PFL de robots de soldadura. Tal ambiente está desarrollado en Visual C++, con la integración de las bibliotecas de renderizado gráfico OpenGL®. La renderización tiene como finalidad proveer una visión en dos dimensiones de un objeto en tres dimensiones, esto es, transforma un objeto diseñado en tres dimensiones y calcula una representación lo más realista posible en las dos dimensiones de la pantalla.

En las versiones actuales de Windows®, se encuentra una opción viable para la gestión de gráficos por computadora, ésta es la interfaz de programación de aplicaciones (API) de OpenGL®, la cual nos brinda el control tanto de cómo los objetos aparecen en la pantalla, como del flujo del programa [82]. Ahondando más sobre esta API, es importante decir, que contiene cientos de instrucciones individuales agrupadas en librerías, las cuales abren un sinfín de posibilidades para explotar este motor gráfico.

El ambiente virtual presentado en esta tesis se concibió con la idea de una manipulación directa de la antorcha virtual utilizando la interfaz háptica, dado que ésta da pie a que los usuarios actúen con más libertad y se confronten a sensaciones más cercanas a la realidad, de tal modo que, el usuario sea capaz de ejecutar acciones mediante dicha interfaz, con un control preciso y que muestre una respuesta inmediatamente en el monitor, con el fin de provocar una sensación de causalidad. Se tomó como base el sistema de ensamble virtual VEDAP-II [83]. Este ambiente representa una estación de trabajo industrial de soldadura, cuyos componentes principales son los objetos virtuales descritos en la sección anterior. Además se incluyen las paredes, el piso y la generación de sombras provocada por una fuente de luz que ilumina la escena. También se utilizaron distintas texturas para forrar todos estos elementos.

Aún más, este ambiente alberga también a las simulaciones provistas para verificar al robot virtual ejecutar las tareas de soldadura antes de que las lleve a cabo el robot real.

3.3.1 Definición de matrices de transformación

Como referencia, la escena virtual tiene al sistema de coordenadas del mundo o global, llamado Σ_G, el cual es aquel sobre el que se define todo el ambiente tridimensional, pues representa el origen del mismo y que por default es el que maneja OpenGL®. En la Figura 3.15 se ilustra la apariencia general del ambiente virtual así como su marco de referencia Σ_G, el cual se encuentra localizado sobre el piso. La vista que presenta esta imagen muestra de frente al usuario a la mesa, sobre la cual se disponen las placas a soldar. También se observa a la antorcha de soldadura, con la cual el usuario interactuará en la escena mediante su manipulación con la interfaz háptica. A esta vista se le considera frontal y es la que aparece por default al iniciar el sistema.

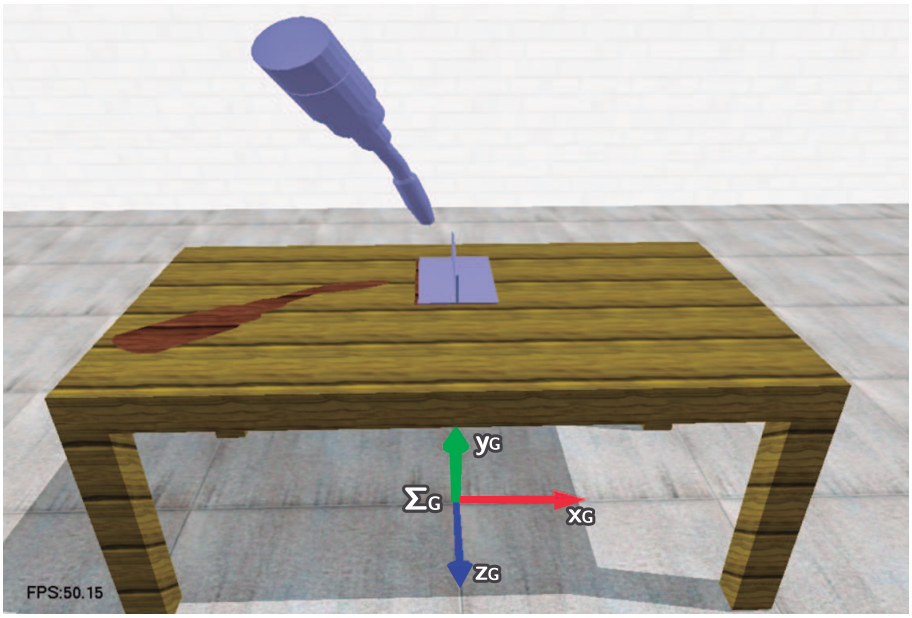


Figura 3.15. Ambiente virtual del sistema.

Todos los objetos son dibujados en el ambiente virtual con respecto a Σ_G. En ese

sentido, la matriz de transformación de las placas a soldar con respecto al marco global está representada por la Ecuación 3.1.

$$ {}^{G}_{P}T = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & ry_P \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1} $$

Donde $ry_P = 46.32\ cm$, que representa la suma de la altura de la mesa (46 cm) más el calibre de las placas (1/8 de pulgada $\approx$ 0.32 cm). De tal forma que el origen de Σ_P siempre estará alineado con el origen de Σ_G, únicamente separados por ry_P.

En el caso de la mesa, la Ecuación 3.2 expone la matriz de transformación que define la pose de la mesa con respecto a las placas dentro de la escena virtual.

$$ {}^{P}_{M}T = \begin{bmatrix} 0 & 1 & 0 & rx_M \\ 0 & 0 & 1 & ry_M \\ 1 & 0 & 0 & rz_M \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2} $$

Donde $ry_M = 0.32\ cm$, valor que es constante, mientras que rx_M y rz_M pueden variar dependiendo la posición requerida de la mesa con respecto a las placas. Se asignaron los valores $rx_M = -19\ cm$, $rz_M = -23\ cm$ considerando la posición de la mesa con respecto a unas placas de prueba en el mundo real.

Para el renderizado en el ambiente virtual de la mesa se debe de calcular la matriz de transformación que defina la pose de ésta con respecto a Σ_G. La Ecuación 3.3 efectúa tal operación.

$$ {}^{G}_{M}T = {}^{G}_{P}T \ {}^{P}_{M}T \tag{3.3} $$

Donde ${}^{G}_{P}T$ es la matriz de las placas con respecto a Σ_G (Ecuación 3.1) y ${}^{P}_{M}T$ es la matriz de la mesa con respecto a las placas (Ecuación 3.2).

Cuando se lleva a cabo la manipulación de la antorcha virtual en el sistema mediante la interfaz háptica, la matriz de transformación del órgano terminal de la interfaz con respecto a Σ_G es provista por OpenHaptics®, denominada ${}^{G}_{H}T$. Sin embargo, se requiere también la definición de una matriz auxiliar ${}^{H}_{A}T$ (Ecuación 3.4) para alinear el marco de la antorcha (Σ_H) con Σ_G y de este modo estar en posibilidades de calcular la pose de la antorcha con respecto al marco global del ambiente virtual (Ecuación 3.5).

$$ {}^{H}_{A}T = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4} $$

$$ {}^{G}_{A}T = {}^{G}_{H}T \ {}^{H}_{A}T \tag{3.5} $$

Donde ${}^{G}_{H}T$ es la matriz del órgano terminal de la interfaz háptica con respecto a Σ_G y ${}^{H}_{A}T$ es la matriz para emparejar el marco de la antorcha con Σ_G (Ecuación 3.4).

En la siguiente sección se explica el procedimiento que se efectuó para incorporar los objetos a la escena virtual.

3.3.2 Exportación de modelos al ambiente virtual

Cualquier objeto tridimensional se puede representar como una red de superficies poligonales, en lo que comúnmente se le denomina malla poligonal. Una malla poligonal es "un conjunto de polígonos que comparten vértices y lados" [82]. En ese contexto, se tiene que un vértice es un punto en el espacio tridimensional de la forma $\{x, y, z\}$. Mientras un lado se define como un segmento de línea que conecta dos vértices; y una cara es un conjunto cerrado de lados, conformando generalmente un triángulo o cuadrilátero.

Para que los objetos previamente modelados fueran insertados en el ambiente virtual

debieron pasar por un proceso que hiciera de ellos una malla poligonal de triángulos, la cual también es conocida como malla triangular, que es una clase de malla poligonal que está compuesta por una serie de caras triangulares interconectadas por sus lados o vértices comunes y definen una superficie, dentro de un entorno tridimensional (Figura 3.16). Una de las razones por la que las mallas triangulares son muy utilizadas en aplicaciones relacionadas con gráficos por computadora es que se fundamenta en el tipo de polígono más simple. Además, que este tipo de mallas permite el cálculo de los vectores normales a las superficies que se requieren para la simulación del proceso de reflexión de la luz sobre éstas [84].

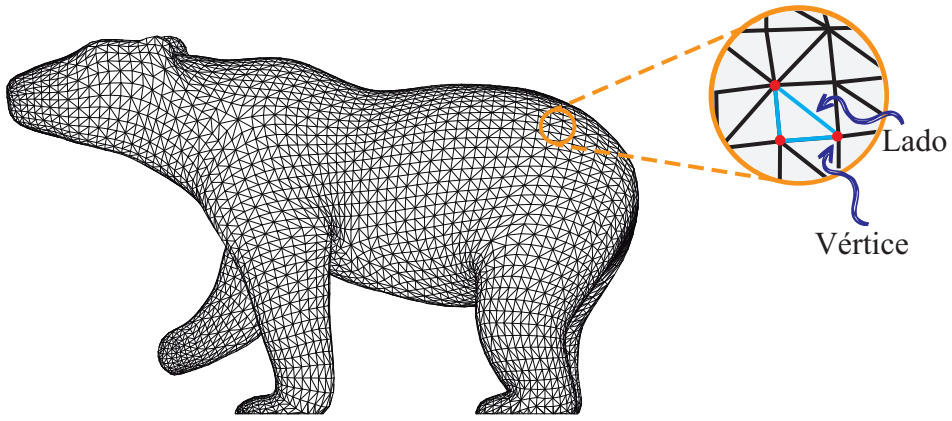


Figura 3.16. Malla poligonal de triángulos representando un oso.

Entonces, para la exportación al ambiente virtual de los objetos mencionados se utilizó el proceso de la Figura 3.17, cuyos detalles se mencionan en los siguientes párrafos.

1. **Exportar el modelo a formato STL**.

   Con el diseño tridimensional del objeto modelado en SolidWorks™, se prosigue a exportar este modelo a un formato puramente geométrico, en este caso un formato llamado STL o de estereolitografía. Este tipo de formato reconoce solamente el aspecto geométrico del modelo y deja a un lado características comunes como información acerca del color y textura. Asimismo, el formato STL se conforma

por una serie de caras triangulares que definen una aproximación poliédrica de las superficies del modelo tridimensional.

2. **Convertir el archivo STL a formato CPP**.

   Una vez teniendo el archivo STL de cada objeto, se procede a generar la malla poligonal mediante la herramienta Deep Exploration™. Este software transforma el formato de estereolitografía a un archivo CPP, el cual respeta las reglas de la estructura del lenguaje de programación Visual C++. En el citado archivo, están almacenadas en una matriz, las coordenadas Cartesianas de los vértices que conforman las caras de la malla poligonal, con respecto al marco de referencia ligado al propio objeto. Asimismo, otra matriz se correlaciona con la primera para conformar los lados de las caras triangulares.

3. **Generar un archivo TRI a partir del archivo CPP**.

   El archivo CPP generado por el paquete Deep Exploration™ es sometido a un procedimiento para extraer exclusivamente los datos relativos a la malla triangular del objeto, como el número de caras triangulares y sus vértices. La información obtenida en este procedimiento es escrita en un archivo de texto que se le asigna la extensión TRI, el cual consta básicamente de tres secciones:

   a) Encabezado

      La información relativa al número de vértices (nv) y al número de caras triangulares (nt) que posee el objeto, se encuentra en este apartado.

   b) Vértices

      En esta sección, cada uno de sus nv renglones está compuesto por tres valores, mismos que representan las coordenadas Cartesianas de un vértice con respecto al marco de referencia ligado al objeto, de tal suerte que cada terceto de éstos posee un índice identificador que va de 1 a nv.

   c) Caras

      Esta sección se caracteriza por agrupar en cada uno de sus nt renglones a los tres índices de los vértices que se requieren para unir los lados y así dar forma

a una cara triangular en particular.

La Tabla 3 concentra la información sobre las mallas triangulares de los modelos de los objetos virtuales.

Tabla 3. Mallas triangulares de los modelos de los objetos virtuales.

Modelo	N° de Vértices (nv)	N° de Caras Triangulares (nt)
Antorcha	6,156	2,052
Placas	84	28
Mesa	228	76

4. Renderizar el objeto en el ambiente virtual.

Finalmente, la información del archivo TRI es almacenada en una matriz durante la ejecución del sistema y ésta es utilizada para renderizar el objeto mediante OpenGL® en la escena virtual. El funcionamiento básico de este procedimiento consiste en dibujar triángulos con las funciones propias de este paquete, a partir de las nt caras triangulares vinculadas con los nv vértices extraidos del archivo TRI.

Las librerías incluidas en OpenGL® permiten la visualización de objetos geométricos tridimensionales y el uso de texturas, iluminación y sombreado de objetos para ofrecer un mayor realismo visual en la pantalla.

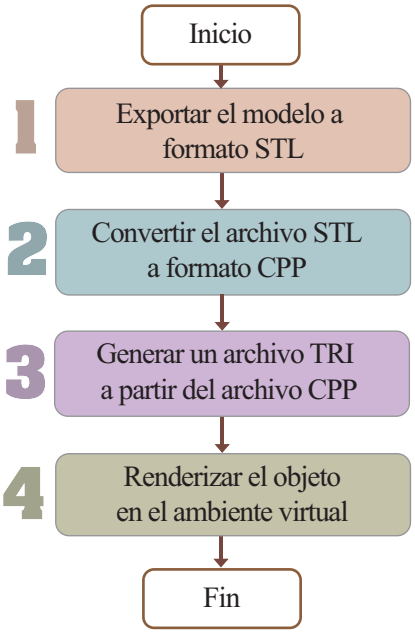


Figura 3.17. Proceso para exportar los modelos al ambiente virtual.

3.4 Manipulación de la antorcha virtual

Se ha planteado que en el sistema para PFL presentado en esta tesis, el usuario tiene que operar la interfaz háptica moviendo su órgano terminal con la mano para localizar la antorcha de soldadura en el ambiente virtual. El movimiento de dicha herramienta se logra en tiempo real durante la manipulación del dispositivo háptico. Las maniobras ejecutadas por el usuario se basan en la percepción visual del mundo virtual desplegado en la computadora.

De acuerdo al enfoque propuesto, cualquier colisión de la antorcha con otros objetos virtuales puede ser detectada por el sistema. Entonces, las señales de control son enviadas a la interfaz háptica de tal forma que el usuario siente en la mano el impacto en tiempo real de la antorcha virtual. Así, una rectificación del movimiento de la interfaz debe guiar al usuario hasta la ubicación correcta de su órgano terminal. El software OpenHaptics® soporta la comunicación entre la computadora y el dispositivo háptico.

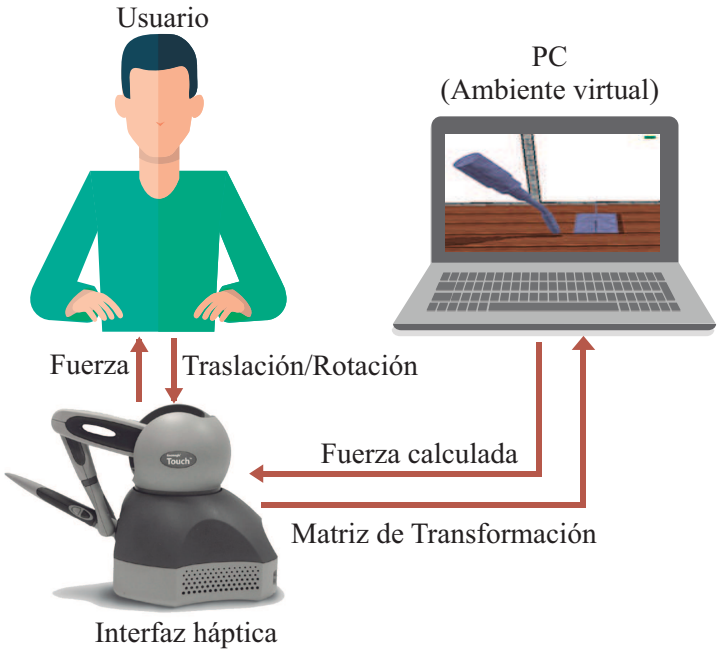


Figura 3.18. Manipulación de la antorcha virtual con una interfaz háptica.

En la Figura 3.18 se ilustra un esquema del proceso que permite una interacción con el ambiente virtual a través de una interfaz háptica. La utilización de este tipo de dispositivos permite al usuario tener el control en tiempo real de la posición y orientación de la antorcha virtual. En todo momento, el usuario se basa en la ubicación que él percibe del entorno virtual que se despliega en la pantalla de su computadora. El uso de esta clase de interfaces además de mejorar la percepción que el usuario tiene de sus propios movimientos, le permite distinguir de forma más real el efecto que éstos tendrían en el mundo real.

3.4.1 Integración de la interfaz háptica

La interfaz háptica utilizada en este trabajo es el modelo mostrado en la Figura 3.19 llamada Geomagic® Touch™, producida por 3D Systems Inc. Las especificaciones técnicas son dadas en la Tabla 4. Es un dispositivo para trabajos de investigación del Grupo de Investigación de Realidad Virtual y Robótica de la UAS, que es considerado un manipulándum de 6 grados de libertad con una pluma como órgano terminal que

puede tomar orientaciones arbitrarias dentro de su espacio de trabajo.

Figura 3.19. Interfaz háptica Geomagic® Touch™.

Tabla 4. Especificaciones de la interfaz Geomagic® Touch™.

Parámetro	Especificación
Espacio de trabajo con retroalimentación de fuerzas	160 mm de ancho, 120 mm de altura y 70 mm de profundidad
Resolución nominal de posición	>450 dpi
Capacidad de fuerza máxima en la postura nominal	3.3 N
Rigidez	Eje x > 1.26 N/mm Eje y > 2.31 N/mm Eje z > 1.02 N/mm
Retroalimentación de fuerza	x, y, z
Interfaz	Puerto Ethernet conforme a RJ45 o puerto USB

El espacio de trabajo del dispositivo háptico puede ser descrito por un marco de referencia. Por default, el eje positivo x apunta a la derecha del dispositivo; el eje positivo y apunta hacia arriba; y el eje positivo z apunta hacia el usuario cuando usa el dispositivo en forma normal (Figura 3.20). Este marco de referencia, el cual se denomina Σ_I, coincide con Σ_G, es decir, con el correspondiente al ambiente virtual. Para fines prácticos, la posición de Σ_I se ubica en el área donde la punta del órgano

terminal del Geomagic® Touch™ permanece en reposo, puesto que a partir de esta posición comienza la traslación de la pluma de la interfaz háptica una vez que el usuario la pone en operación.

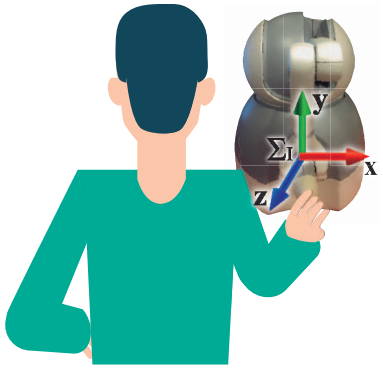


Figura 3.20. Marco de referencia de la interfaz háptica.

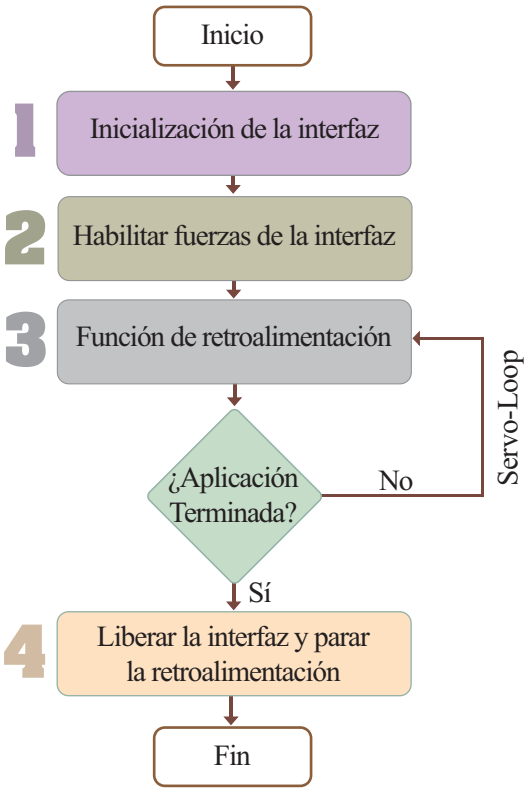


Figura 3.21. Proceso de la integración de la háptica al sistema.

Existe un área de investigación emergente llamada háptica computacional, que está relacionada con el desarrollo de algoritmos para generar y renderizar el toque de

entornos y objetos virtuales, justo como los gráficos de computadora lo hacen con la generación y renderizado de imágenes. La háptica computacional tiene dos componentes, renderizado háptico y renderizado visual, que comunica los gráficos del ambiente virtual, sonido, y respuestas de fuerza para el usuario. El renderizado háptico es considerado el núcleo de cualquier aplicación basada en háptica, maneja algoritmos para detectar y reportar cuándo y dónde el contacto geométrico ha ocurrido (detección de colisión) y calcula la fuerza de interacción correcta entre un dispositivo háptico y su entorno virtual (respuesta de colisión) [78, 85].

Para activar la háptica dentro del sistema fue necesario cumplir el proceso ilustrado en la Figura 3.21 [86] y que enseguida se explica:

1. **Inicialización de la interfaz**.

   Incluye todo lo necesario para comunicarse con la interfaz. Esto generalmente involucra crear un manejador de la interfaz y calibrar el dispositivo. También se configura el grado de dificultad con el que éste se mueve a través del espacio.

2. **Habilitar fuerzas de la interfaz**.

   Este tipo de dispositivos son inicializados con las fuerzas deshabilitadas por seguridad; por lo que se requiere ejecutar una rutina para habilitarlas.

3. **Función de retroalimentación**.

   Esta función se ejecuta en un ciclo para hacer consultas y cambiar estados. La llamada a esta función es asíncrona porque regresa inmediatamente después de que se completa la operación sin esperar. Una devolución de una llamada asíncrona puede persistir en la función para representar un efecto háptico. Durante cada iteración, la devolución de esta llamada aplica el efecto al dispositivo.

   Servo-loop. Se refiere al ciclo cerrado de control utilizado para calcular las fuerzas que se envían al dispositivo háptico. Para hacer una retroalimentación háptica estable, este ciclo debe ejecutarse a una

velocidad constante de 1 KHz o más, mientras la aplicación no sea terminada. Para mantener una tasa de actualización tan alta, el servo-loop se ejecuta generalmente en un hilo separado de alta prioridad, nombrado hilo del servo-loop.

4. Liberar la interfaz y parar retroalimentación

Se deshabilita el dispositivo háptico y se detiene la función de retroalimentación.

Por otro lado, el kit de herramientas de OpenHaptics® permite añadir háptica y habilitar una navegación tridimensional a las aplicaciones. Asimismo, facilita la integración con aplicaciones de OpenGL®. Este paquete maneja cálculos complejos, proporciona el control del dispositivo háptico a los desarrolladores y admite objetos poligonales, propiedades de materiales y efectos de fuerza.

Existen bibliotecas de OpenHaptics® que son utilizadas dentro de Visual C++ que brindan la oportunidad de establecer una intercomunicación entre la interfaz háptica y el ambiente virtual. Estas bibliotecas contienen la información cinemática de la interfaz, esto es, poseen funciones por medio de las cuales se obtienen la posición y orientación de los eslabones, de las articulaciones y del órgano terminal del dispositivo háptico en cualquier instante, facilitando de esta forma la programación del sistema. Evidentemente, con la incorporación de la interfaz háptica para la manipulación de la antorcha virtual en el sistema, se recurrió a las funciones de las bibliotecas provistas por OpenHaptics®.

A diferencia de los otros objetos virtuales de la escena, la antorcha manifiesta movimiento durante su manipulación con la interfaz háptica, por lo que es justo señalar que para que en la escena virtual exista una percepción de movimiento continuo, se emplea una función que declara un ciclo infinito para la actualización de los gráficos en pantalla. Dentro de este ciclo se dibuja una y otra vez el contenido de la escena, donde todos los objetos virtuales se redibujan a una frecuencia de 33 Hz. Se utilizaron funciones de traslación y rotación de OpenGL® para lograr tal objetivo.

3.4.2 Cálculo de la fuerza de contacto

La simulación del contacto con un objeto virtual equivale a un cálculo de fuerzas que resiste el órgano terminal de la interfaz háptica para penetrar en la superficie del objeto virtual. Un enfoque para simular esta interacción es a través del concepto de un proxy que sigue la transformación del órgano terminal del dispositivo en el entorno virtual [86].

Una fuerza de resorte es probablemente el cálculo de fuerza más común utilizado en el renderizado háptico porque es muy versátil y fácil de usar. Se puede calcular una fuerza elástica aplicando la Ley de Hooke con amortiguamiento:

$$F = kx - b\vartheta \tag{3.6}$$

Donde k es una constante de rigidez; x es un vector de desplazamiento; b es una constante de amortiguamiento y ϑ es la velocidad lineal de desplazamiento de la antorcha. El resorte está unido entre una posición fija de anclaje p_0 y la posición del órgano terminal de la interfaz p_1. La posición fija de anclaje generalmente se coloca en la superficie del objeto que el usuario está tocando. El vector de desplazamiento $x = p_0 - p_1$ es tal que la fuerza del resorte siempre se dirige hacia la posición fija de anclaje. La fuerza que se siente se llama la fuerza de restauración del resorte, ya que el resorte está tratando de restablecerse a su longitud de descanso, la cual en este caso es cero. La constante de rigidez k dicta cuán agresivamente el resorte intentará restablecerse a su longitud de descanso. Una constante de baja rigidez hará que el resorte se sienta flojo, mientras que una constante de alta rigidez se sentirá tenso.

A la posición fija de anclaje referida previamente se le denomina punto de contacto en la superficie (PCS), el cual es un punto que intenta seguir la posición del órgano terminal de la interfaz háptica pero está limitado a estar en la superficie del objeto. En espacio libre, el PCS coincide con la posición del órgano terminal de la interfaz, como se muestra en t_0. Al tocar un objeto, el PCS puede calcularse moviendo

el último PCS hacia la posición del órgano terminal de la interfaz sin penetrar la superficie. La fuerza se calcula simulando un resorte estirado desde la posición del órgano terminal de la interfaz al PCS. El círculo t_1 muestra la penetración en el objeto. Asimismo t_2 muestra una mayor penetración: el resorte se alarga más y, por lo tanto, el usuario sentirá una mayor resistencia. La Figura 3.22 ilustra lo recién explicado.

Debido al modelado de la antorcha virtual, el órgano terminal de la interfaz háptica controla la punta de la misma, permitiendo el cálculo del vector de desplazamiento x al penetrar ésta las placas o la mesa dentro de la estación de trabajo industrial representada en el ambiente virtual, y así determinar la fuerza de contacto de acuerdo a la Ecuación 3.6 [35].

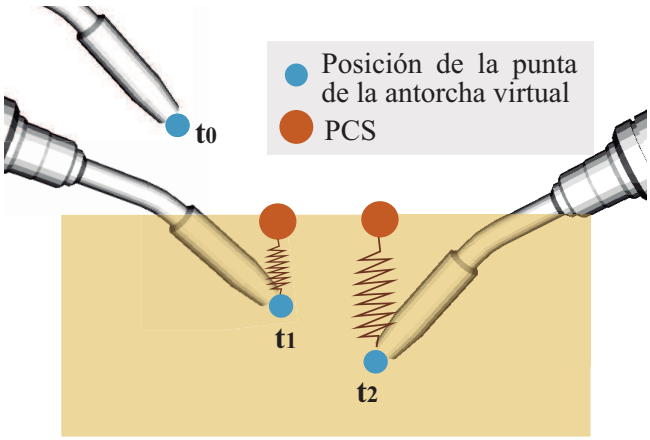


Figura 3.22. Punto de contacto en la superficie.

En esta parte del sistema, es de gran utilidad la función de retroalimentación explicada en la sección anterior y que se ilustra en la Figura 3.21. Vale la pena señalar que en esta función se establecen los umbrales de los PCS, esto es, se hacen las validaciones correspondientes para determinar si la punta de la antorcha está haciendo contacto o penetrando algún otro objeto del ambiente virtual. Para el caso de la soldadura de las placas en T, es necesario establecer límites en los ejes del sistema de coordenadas global.

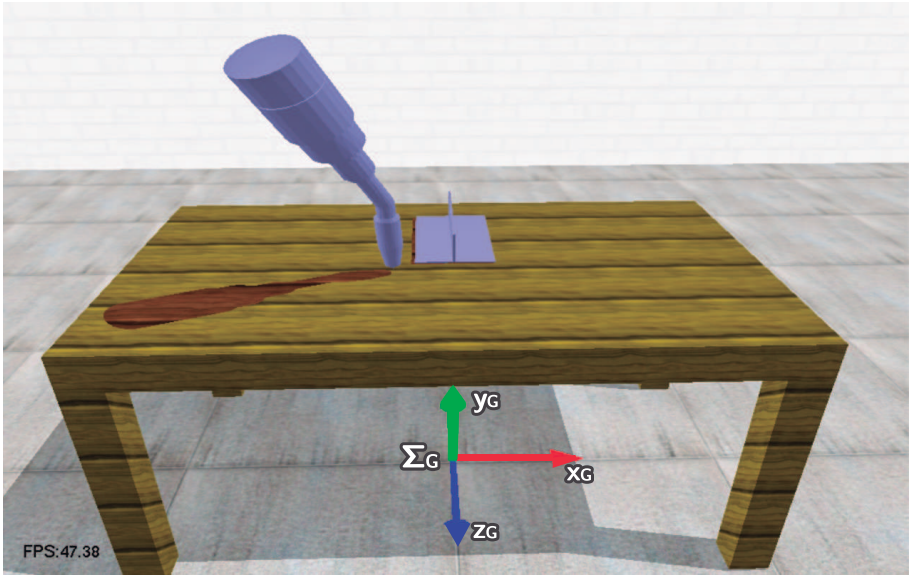


Figura 3.23. Antorcha virtual haciendo contacto la mesa.

En la Figura 3.23 se observa que la punta de la antorcha hace contacto con la mesa, donde ésta se despliega sobre el plano x_G-z_G; sin embargo, en este caso la validación recae sobre el eje y_G, estableciendo un límite L_y, esto se debe a que este eje determina la altura a la que se posiciona la punta de la antorcha; si ésta se posiciona a una altura menor que este límite y además se encuentra dentro de los límites de la mesa (L_{Mx_1}, L_{Mx_2}, L_{Mz_1} y L_{Mz_2}) entonces se utiliza la Ecuación 3.6 para calcular la fuerza que se le envía a la interfaz háptica, una vez que ha transgredido L_y. Estos límites corresponden a posiciones con respecto a Σ_G. La Figura 3.24 ilustra estos límites.

Algo similar ocurre cuando la punta de la antorcha se mueve sobre el eje x_G y se topa con la placa dispuesta verticalmente, en este caso si se mueve de x_G- a x_G+ existe un valor límite en x_G denominado L_x. A esta condicionante la acompañan dos valores límites en y_G correspondiente a la altura de las placas, denominados L_{y1} y L_{y2}; más dos valores límites en z_G que conciernen al tamaño de las placas sobre ese plano (L_{z1} y L_{z2}). Si la punta de la antorcha supera L_x y además se encuentra localizada dentro de estos límites se utiliza la Ecuación 3.6 para calcular la fuerza para transmitirla al dispositivo háptico. En la Figura 3.25 se representan estos límites.

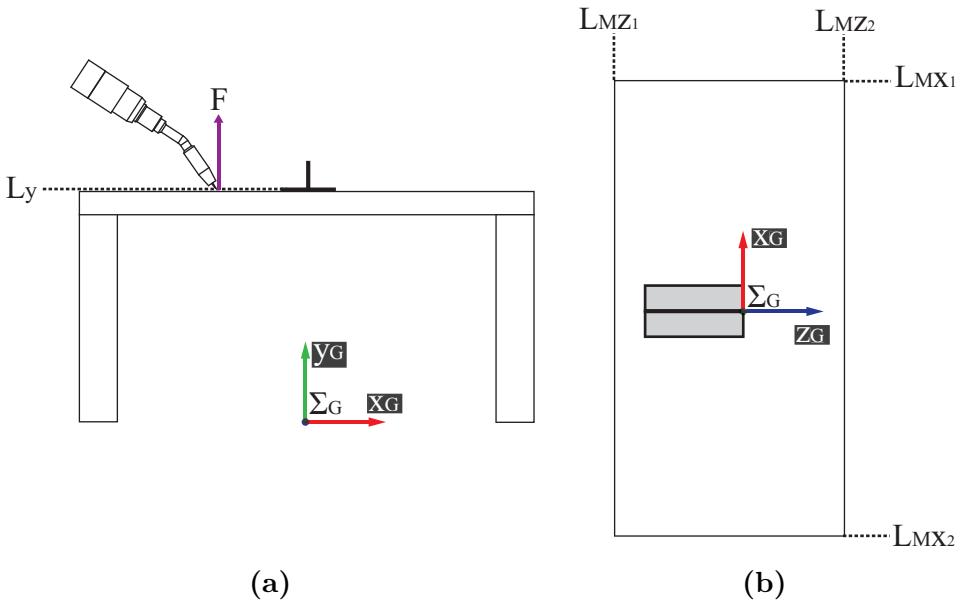


Figura 3.24. Límites para cálculo de fuerza vertical.
((a) en *vista frontal*; (b) en *vista superior*)

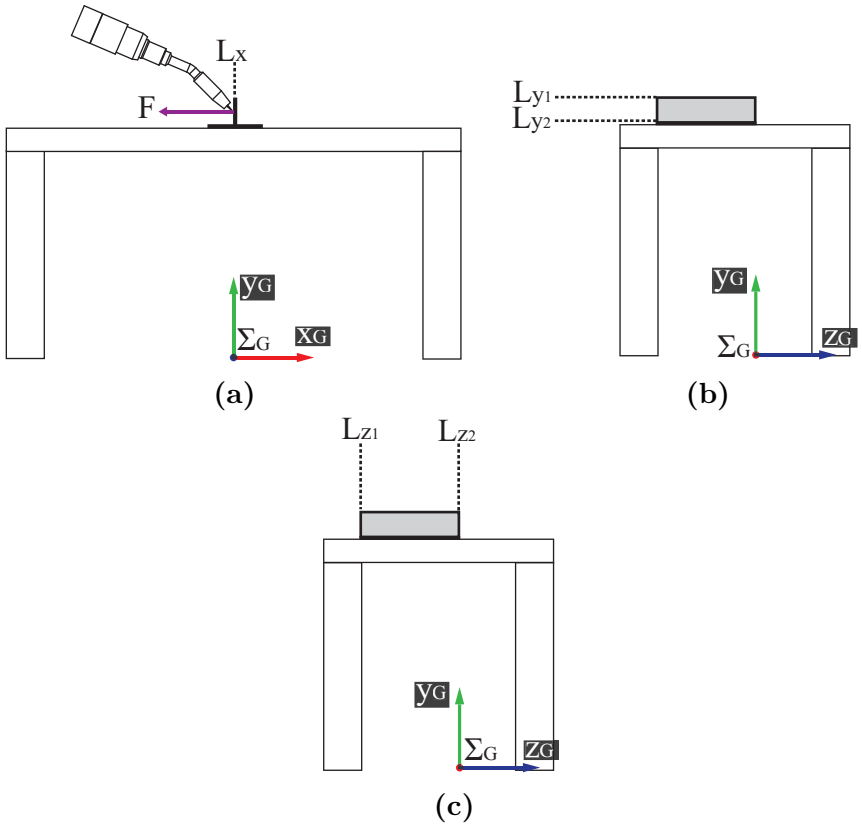


Figura 3.25. Límites para cálculo de fuerza horizontal.
((a) en *vista frontal*; (b) y (c) en *vista izquierda*)

3.4.3 Generación de puntos nodo

Hay que recordar que la PFL es una alternativa para la especificación de los movimientos deseados de los robots industriales. A esta actividad se le denomina planificación de trayectorias y es justamente en esta sección donde se detalla tal actividad, la cual invoca a la tercera fase del proceso de PFL planteado en este documento. En la Figura 3.26 se esquematiza el proceso a seguir para cumplir con esta función y en los siguientes párrafos se explica a detalle tal proceso.

Cabe señalar que en el enfoque propuesto en esta tesis, la interfaz háptica Geomagic® Touch™ se aplica en un sistema de PFL para la planeación de tareas de soldadura para un manipulador industrial. En el proceso de enseñanza de dichas tareas, el usuario debe especificar muestras de poses (posiciones y orientaciones) de la antorcha que debe alcanzar el robot cuando éste ejecute las tareas en cuestión. Cada pose corresponde a un punto nodo de la trayectoria deseada y para enseñar las poses al sistema, el usuario mueve la antorcha virtual accionando el dispositivo háptico para dibujar un punto en el ambiente virtual con la apariencia de una esfera.

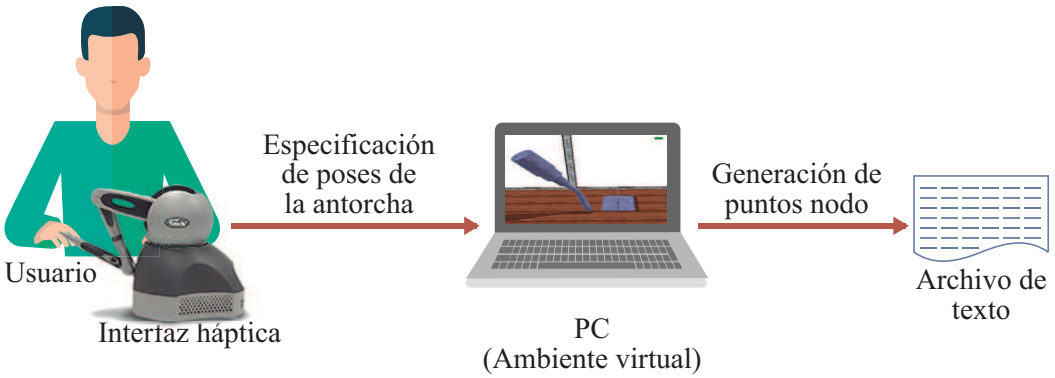


Figura 3.26. Proceso para la generación de puntos nodo.

Las coordenadas operacionales de estos puntos nodo son generadas acorde a la posición de la punta de la antorcha virtual, en otras palabras, el usuario mueve la antorcha dentro de la escena virtual al manejar la interfaz háptica y cuando él lo desee generará un punto nodo en la posición donde se localiza la punta de la antorcha con su respectiva orientación. Se planteó que tales coordenadas operacionales deben

ser con respecto al marco de las placas (Σ_P), para lo cual se tiene que efectuar la operación indicada en la Ecuación 3.7.

$$ {}^{P}_{A}T = {}^{P}_{G}T \; {}^{G}_{A}T \tag{3.7} $$

Donde ${}^{P}_{G}T$ es la matriz de Σ_G con respecto a las placas o bien la matriz inversa de ${}^{G}_{P}T$ (Ecuación 3.1); ${}^{G}_{A}T$ es la matriz de la antorcha con respecto a Σ_G (Ecuación 3.5).

$$ {}^{P}_{A}T = \begin{bmatrix} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{24} \\ t_{31} & t_{32} & t_{33} & t_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.8} $$

Considerando la matriz de la Ecuación 3.8, primeramente, las coordenadas Cartesianas, que definen la posición de la punta de la antorcha virtual con respecto a las placas son los tres primeros elementos de la última columna de la matriz (color amarillo), donde:

$$ x_{GT} = t_{14} \tag{3.9a} $$

$$ y_{GT} = t_{24} \tag{3.9b} $$

$$ z_{GT} = t_{34} \tag{3.9c} $$

Por su parte, es necesario echar mano de las operaciones de las Ecuaciones 2.8 (ángulos de Bryant) para calcular la orientación de la punta de la antorcha virtual con respecto a las placas mediante los ángulos λ, μ y ν, tomando en cuenta la matriz ${}^{P}_{A}T$ (Ecuación 3.8).

Cuando se alcanza la pose deseada de la antorcha virtual, se almacenan las coordenadas operacionales que representan a dicha pose en un archivo de texto. En ese sentido, en dicho archivo se guarda la fecha y hora de cada experimento, así como la información referente a los puntos nodo. Esta información se describe en la Tabla 5.

Ahora bien, en la generación de las trayectorias deseadas definiremos dos tipos de puntos nodo: puntos flotantes y puntos de soldadura. Los primeros toman ese nombre porque se encuentran flotando dentro de la escena virtual. El segundo tipo se refiere a aquellos puntos que representan el cordón de soldadura *per se* y que se encuentran en la(s) junta(s) que se desee(n) soldar.

Tabla 5. Parámetros de los puntos nodo que se almacenan en cada trayectoria.

Parámetro	Descripción
Punto	Número consecutivo que identifica a cada punto nodo.
Px	Posición en cm de la punta de la antorcha con respecto al eje x de Σ_P.
Py	Posición en cm de la punta de la antorcha con respecto al eje y de Σ_P.
Pz	Posición en cm de la punta de la antorcha con respecto al eje z de Σ_P.
Rx	Ángulo en $^\circ$ de la antorcha con respecto al eje x de Σ_P.
Ry	Ángulo en $^\circ$ de la antorcha con respecto al eje y de Σ_P.
Rz	Ángulo en $^\circ$ de la antorcha con respecto al eje z de Σ_P.

Dado que los puntos nodo definidos conformarán la trayectoria que deberá seguir el robot para ejecutar la tarea de soldadura, es justo señalar que los puntos flotantes fungen el papel de acercamiento o alejamiento a dicha tarea, esto es, los primeros puntos nodo permitirán al robot acercarse al área adonde generará el cordón de soldadura, mientras que los puntos finales regresarán al robot a su pose inicial.

Con base en la Figura 3.27 se pueden identificar los dos tipos de puntos nodo descritos con antelación. Los puntos flotantes son las esferas rojas y las esferas amarillas. De éstas, las primeras son los puntos que permiten acercarse a la tarea de soldadura (puntos de ataque); y las segundas representan aquellos puntos destinados a alejarse

del cordón de soldadura para retornar al estado original del manipulador o posición de *home* (puntos de escape). Los puntos de soldadura aparecen en color verde y se encuentran asentados en las placas sobre la mesa (puntos de cordón). En los primeros tres incisos de dicha figura se observan distintas perspectivas del ambiente virtual donde se aprecia la totalidad de los puntos nodo. Mientras que el inciso *d* es un acercamiento a los puntos de soldadura en la placa.

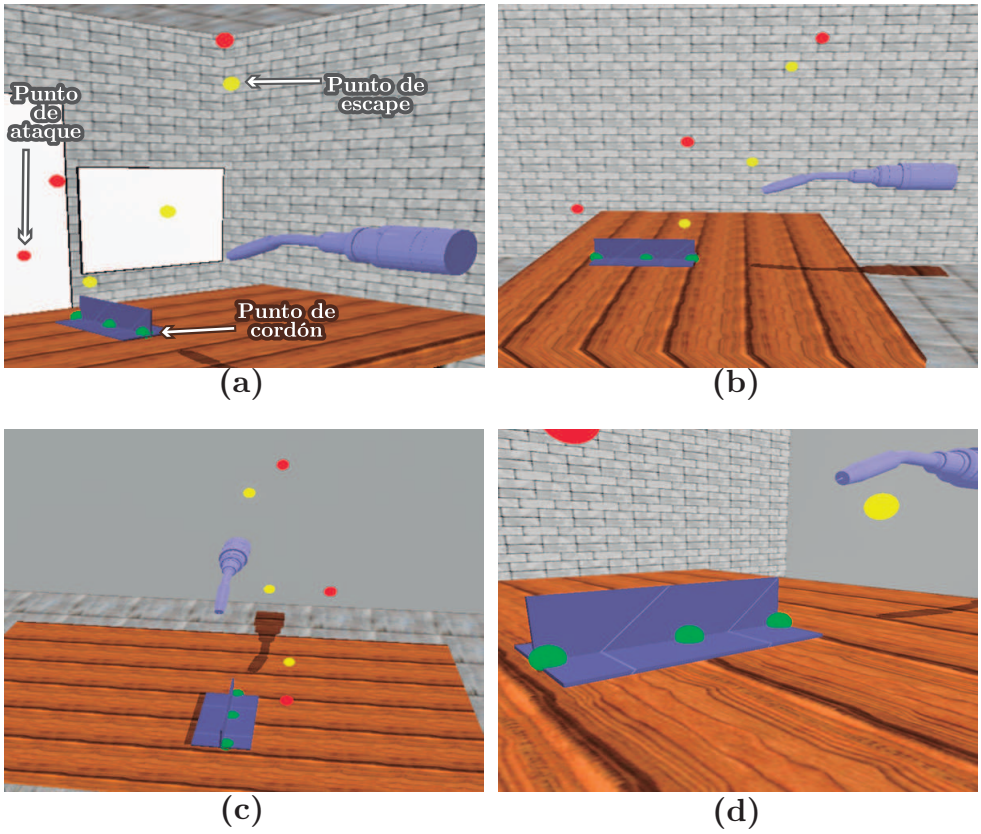


Figura 3.27. Puntos nodo en el ambiente virtual.

Al finalizar esta etapa de la PFL se pretende habilitar al robot virtual para ejecutar una secuencia de movimientos seguros y sin colisiones para llevar a cabo la tarea de soldadura en una simulación.

En la Figura 3.28 se observa la información de una tarea de soldadura que está conformada por cinco puntos nodo con las respectivas coordenadas operacionales de la antorcha de soldadura. En dicha figura se expone la estructura que presenta el archivo de texto donde se han almacenado particularmente los parámetros de los

puntos nodo de una trayectoria deseada acorde a los datos proporcionados por la Tabla 5.

```
*****************************************************************
*       Experiment started on 16/07/2018 - 10:40:54             *
*****************************************************************
        Punto   Px      Py      Pz      Rx      Ry      Rz
        P1      11.83   19.76   -21.24  55.40   15.45   18.80
        P2      17.10   3.25    -7.96   45.43   11.83   16.22
        P3      -3.34   0.40    -4.11   47.14   8.58    13.84
        P4      -3.16   0.14    -9.46   36.79   6.21    8.74
        P5      -0.60   29.57   -9.68   44.39   20.87   43.05
```

Figura 3.28. Parámetros de los puntos nodo en el archivo de texto.

Como se demostrará más delante, estos parámetros serán empleados para calcular la cinemática inversa del robot Fanuc ARC Mate 100iC y estar en posibilidad de ejecutar las simulaciones requeridas durante este trabajo.

Capítulo 4

Modelado del robot virtual

La peculiaridad de este capítulo radica en la descripción del robot utilizado para la validación de la programación fuera de línea. Aunado a lo anterior se detallan los modelos de las piezas que conforman al robot virtual. Para concluir, se dan los pormenores de la simulación desarrollada en la cual el robot virtual ejecuta la(s) tarea(s) planificada(s) mediante la interfaz háptica.

4.1 Características del robot Fanuc ARC Mate 100iC

El robot Fanuc ARC Mate 100iC ("robot Fanuc" en lo sucesivo) es un robot articulado con 6 grados de libertad, que posee alta precisión y velocidad. Es empleado específicamente para tareas de soldadura de arco. El Laboratorio de Mecatrónica y Control del Instituto Tecnológico de La Laguna dispone de un robot industrial de soldadura de este tipo (Figura 4.1), que sirve para la validación experimental de los procesos propuestos en esta tesis. En la Figura 4.2 se muestra sus dimensiones físicas extraídas de la ficha técnica del robot provista por el fabricante [87].

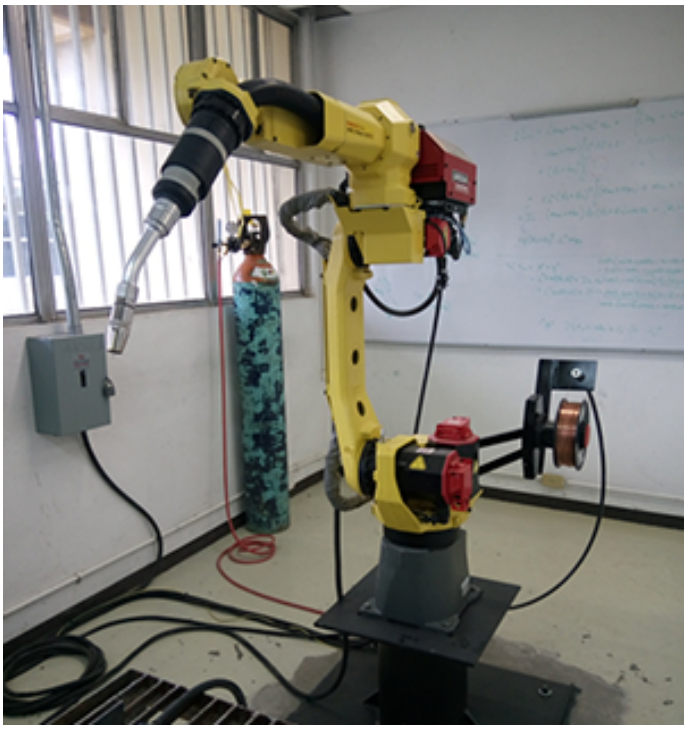


Figura 4.1. Robot industrial de soldadura Fanuc ARC Mate 100iC.

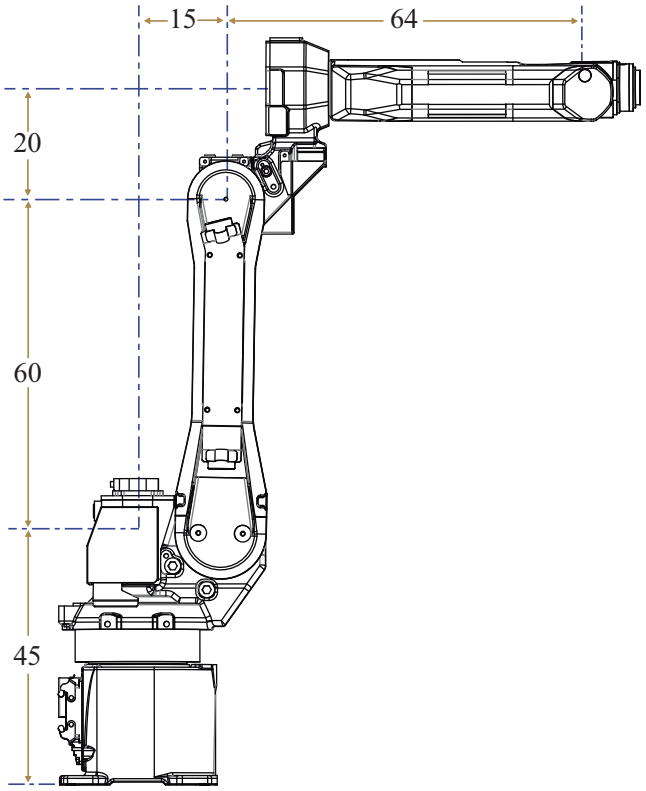


Figura 4.2. Dimensiones físicas en *cm* del robot Fanuc.
(*Vista frontal*)

La capacidad de movimiento de este robot se ilustra en la Figura 4.3 y permite el control de sus articulaciones. Concretamente, este robot tiene 6 articulaciones en su estructura. Es justo señalar que cada una de estas articulaciones presenta límites físicos en lo que se refiere a su rotación, en otras palabras, existe un tope en términos de valores angulares al momento de rotar estos ejes. Nombrando a estas articulaciones como J_i donde i es el número de articulación, la Tabla 6 indica estos valores.

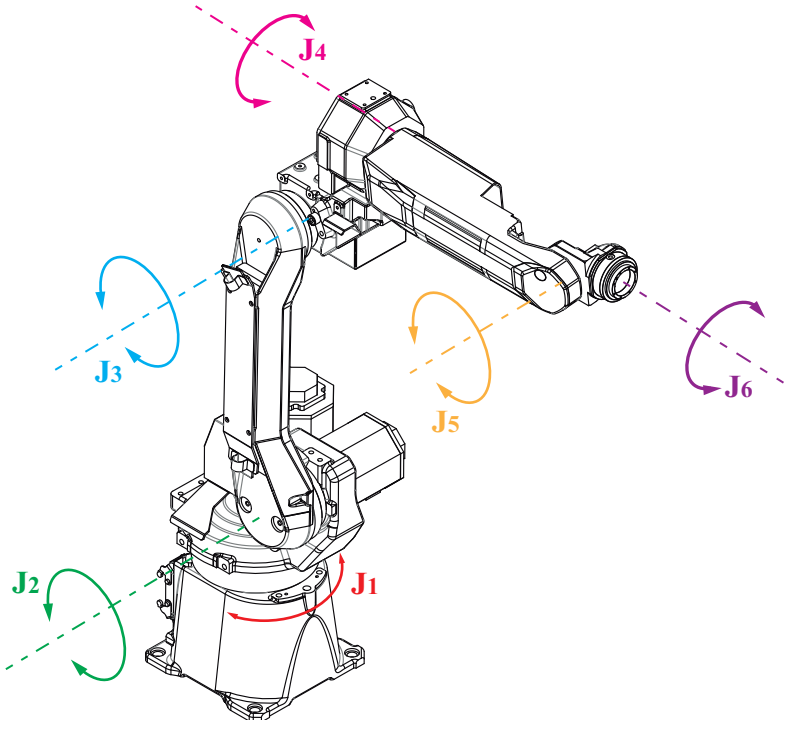


Figura 4.3. Grados de libertad del robot Fanuc.
(*Vista isométrica*)

Tabla 6. Límites articulares del robot Fanuc.

Articulación	Límite Inferior (°)	Límite Superior (°)
J_1	- 170	+ 170
J_2	- 90	+ 160
J_3	-180	+ 264.5
J_4	- 190	+ 190
J_5	- 140	+ 140
J_6	- 270	+ 270

Es preciso señalar que tal robot está compuesto por 7 eslabones y en la Sección 4.2 se indicarán las características más importantes de tales piezas.

4.1.1 Parámetros de Denavit-Hartenberg del robot

Acorde a las dimensiones físicas del robot y mediante la asignación de marcos de referencia en los eslabones aplicando la convención modificada de Denavit-Hartenberg (Sección 2.1.3.1) se obtuvo el esquema cinemático ilustrado en la Figura 4.4. De igual modo, siguiendo las especificaciones para la definición de los parámetros de tal convención (Tabla 1), se conformó la Tabla 7.

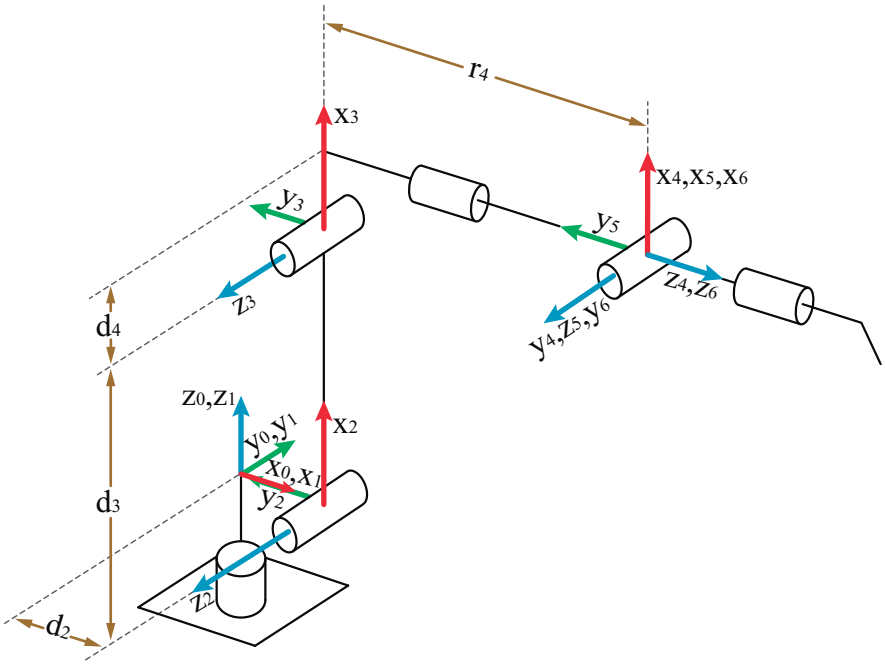


Figura 4.4. Esquema cinemático del robot Fanuc.

Tabla 7. Parámetros de Denavit-Hartenberg modificados del robot Fanuc.

i	$\alpha_i(^\circ)$	d_i	θ_i	r_i
1	0	0	θ_1	0
2	90	d_2	θ_2	0
3	0	d_3	θ_3	0
4	90	d_4	θ_4	r_4
5	-90	0	θ_5	0
6	90	0	θ_6	0

Donde de acuerdo a las dimensiones físicas del robot dadas en la Figura 4.2 se tiene que d_2 = 15 cm, d_3 = 60 cm, d_4 = 20 cm y r_4 = 64 cm.

4.1.2 Matrices elementales del robot

La matriz de transformación homogénea general es una matriz de 4×4 que relaciona el marco de referencia *i-ésimo* con el marco de referencia *(i-1)-ésimo* (Ecuación 2.13). Dada esta matriz se describen a las matrices elementales del robot en

términos de los parámetros de Denavit-Hartenberg. De tal forma que sustituyendo los parámetros geométricos de la Tabla 7 en dicha matriz se obtienen las siguientes matrices elementales:

$$ {}^{0}_{1}T = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{1}_{2}T = \begin{bmatrix} c_2 & -s_2 & 0 & d_2 \\ 0 & 0 & -1 & 0 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1a} $$

$$ {}^{2}_{3}T = \begin{bmatrix} c_3 & -s_3 & 0 & d_3 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{3}_{4}T = \begin{bmatrix} c_4 & -s_4 & 0 & d_4 \\ 0 & 0 & -1 & -r_4 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1b} $$

$$ {}^{4}_{5}T = \begin{bmatrix} c_5 & -s_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_5 & -c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{5}_{6}T = \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1c} $$

Donde c_i significa $cos\theta_i$ y s_i representa a $sen\theta_i$ para $i = 1, 2, 3, 4, 5, 6$.

4.1.3 Modelo cinemático directo del robot

Considerando la esencia de la cinemática directa, que consiste en determinar la posición y orientación del órgano terminal del robot, con respecto a un sistema de coordenadas que se toma como referencia, conocidos los valores de las articulaciones y los parámetros geométricos de los elementos del robot; estando definidas las matrices elementales que describen individualmente la pose de un marco de referencia con respecto a su antecesor; y aprovechando las propiedades de dichas matrices al multiplicarse entre sí, se tiene que la matriz que describe al marco de referencia de la última articulación (Σ_6) respecto al marco Σ_0 se obtiene con la siguiente ecuación:

$$ {}^{0}_{6}T = {}^{0}_{1}T \ {}^{1}_{2}T \ {}^{2}_{3}T \ {}^{3}_{4}T \ {}^{4}_{5}T \ {}^{5}_{6}T \tag{4.2} $$

Dando como resultado la siguiente matriz:

$$ {}^{0}_{6}T = \begin{bmatrix} \begin{matrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{matrix} & \begin{matrix} t_{14} \\ t_{24} \\ t_{34} \end{matrix} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 1 \end{bmatrix} \tag{4.3} $$

Donde:

$$ t_{11} = c_1c_{23}c_4c_5c_6 + c_5c_6s_1s_4 - c_1c_6s_{23}s_5c_4s_1s_6 + c_1c_{23}s_4s_6 \tag{4.4a} $$

$$ t_{21} = c_{23}c_4c_5c_6s_1 - c_1c_5c_6s_4 - c_6s_1s_{23}s_5 - c_1c_4s_6 - c_{23}s_1s_4s_6 \tag{4.4b} $$

$$ t_{31} = c_4c_5c_6s_{23} + c_{23}c_6s_5 - s_{23}s_4s_6 \tag{4.4c} $$

$$ t_{12} = c_4c_6s_1 - c_1c_{23}c_6s_4 - c_1c_{23}c_4c_5s_6 - c_5s_1s_4s_6 + c_1s_{23}s_5s_6 \tag{4.4d} $$

$$ t_{22} = -c_1c_4c_6 - c_{23}c_6s_1s_4 - c_{23}c_4c_5s_1s_6 + c_1c_5s_4s_6 + s_1s_{23}s_5s_6 \tag{4.4e} $$

$$ t_{32} = -c_6s_{23}s_4 - c_4c_5s_{23}s_6 - c_{23}s_5s_6 \tag{4.4f} $$

$$ t_{13} = c_1c_5s_{23} + c_1c_{23}c_4s_5 + s_1s_4s_5 \tag{4.4g} $$

$$ t_{23} = c_5s_1s_{23} + c_{23}c_4s_1s_5 - c_1s_4s_5 \tag{4.4h} $$

$$ t_{33} = -c_{23}c_5 + c_4s_{23}s_5 \tag{4.4i} $$

$$ t_{14} = d_2c_1 + d_3c_1c_2 + d_4c_1c_{23} + r_4c_1s_{23} \tag{4.4j} $$

$$ t_{24} = d_2s_1 + d_3c_2s_1 + d_4c_{23}s_1 + r_4s_1s_{23} \tag{4.4k} $$

$$ t_{34} = -r_4c_{23} + d_3s_2 + d_4s_{23} \tag{4.4l} $$

Donde c_i equivale a $cos\theta_i$, s_i simboliza a $sen\theta_i$ para $i = 1,2,3,4,5,6$; c_{23} representa a $cos(\theta_2 + \theta_3)$ y s_{23} significa $sen(\theta_2 + \theta_3)$.

No obstante que aquí se presenta la metodología para obtener las matrices elementales del robot, cabe señalar que la resolución de estas matrices se efectuó mediante el software SYMORO (SYmbolic Modelling of RObots) [88]. Este sistema genera automáticamente modelos simbólicos como el modelo cinemático directo y el modelo cinemático inverso.

Ahora bien, con la matriz ${}^{0}_{6}T$ resuelta (Ecuación 4.3), se procede a calcular las coordenadas operacionales de la última articulación del robot. Para esto, se sigue el procedimiento utilizado a lo largo de este documento, es decir, la posición de Σ_6 con respecto a Σ_0 en función de coordenadas Cartesianas se encuentra en los tres primeros renglones de la última columna (parte cian de la matriz). Por otro lado, si se desea obtener su orientación se toman las Ecuaciones 2.8 y se calculan λ, μ y ν.

4.1.4 Modelo cinemático inverso del robot

Al igual que el modelo cinemático directo, el modelo cinemático inverso del robot Fanuc fue resuelto por el sistema SYMORO, a través del método de Paul, el cual dicta que es necesario definir la matriz *snap* (Ecuación 4.5), misma que se define a partir de la matriz ${}^{0}_{6}T$ y cuyos elementos quedan dispuestos como sigue:

$$ {}^{0}_{6}T = \begin{bmatrix} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{24} \\ t_{31} & t_{32} & t_{33} & t_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_x & n_x & a_x & p_x \\ s_y & n_y & a_y & p_y \\ s_z & n_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.5} $$

Según la cinemática inversa, en un manipulador de seis grados de libertad, dada la posición y orientación del órgano terminal en la matriz *snap*, se requiere encontrar los ángulos de articulación correspondientes $q = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)^T$ para lograr una posición y orientación deseadas en el órgano terminal. Este proceso se siguió empleando el programa de cálculo simbólico mediante el paquete SYMORO y las ecuaciones generadas por este software en función de los ángulos articulares son las siguientes:

$$ \theta_1 = atan2(E_1\ p_y, E_1\ p_z) \tag{4.6a} $$

$$ z_1 = -d_2 \ + \ p_x\ c_1 \ + \ p_y\ s_1 $$

$$ b_1 = 2(-(d_4\ p_z) \ - \ r_4\ z_1) $$

$$ b_2 = 2(p_z\ r_4 \ - \ d_4\ z_1) $$

$$b_3 = d_3^2 \; - \; d_4^2 \; - \; p_z^2 \; - \; r_4^2 \; - \; z_1^2$$

$$sq = \left(b_1 \; b_3 \; + \; E_2 \; b_2 \sqrt{b_1^2 \; + \; b_2^2 \; - \; b_3^2} \right) \; / \; (b_1^2 \; + \; b_2^2)$$

$$cq = \left(b_2 \; b_3 \; - \; E_2 \; b_1 \sqrt{b_1^2 \; + \; b_2^2 \; - \; b_3^2} \right) \; / \; (b_1^2 \; + \; b_2^2)$$

$$\theta_2 = atan2(-((-p_z \; - \; r_4 \; cq \; + \; d_4 \; sq) \; / \; d_3) \, , \, (z_1 \; - \; d_4 \; cq \; - \; r_4 \; sq) \; / \; d_3) \tag{4.6b}$$

$$\theta_3 = atan2(sq, cq) \; - \; \theta_2 \tag{4.6c}$$

$$x_1 = -(a_y \; c_1) \; + \; a_x \; s_1$$

$$y_1 = -(a_x \; c_1 \; c_{23}) \; - \; a_y \; c_{23} \; s_1 \; - \; a_z \; s_{23}$$

$$\theta_4 = atan2(E_4 \; x_1, E_4(-y_1)) \tag{4.6d}$$

$$y_2 = -(c_4(-(s_y \; c_1) \; + \; s_x \; s_1)) \; + \; (s_4(s_x \; c_1 \; c_{23} \; + \; s_y \; c_{23} \; s_1 \; + \; s_z \; s_{23}))$$

$$y_3 = a_z \; c_{23} \; - \; a_x \; c_1 \; s_{23} \; - \; a_y \; s_1 \; s_{23}$$

$$\theta_5 = atan2(-y_2, -y_3) \tag{4.6e}$$

$$y_4 = -(c_4(-(s_y \; c_1) \; + \; s_x \; s_1)) \; + \; (s_4(s_x \; c_1 \; c_{23} \; + \; s_y \; c_{23} \; s_1 \; + \; s_z \; s_{23}))$$

$$y_5 = -(c_4(-(n_y \; c_1) \; + \; n_x \; s_1)) \; + \; (s_4(n_x \; c_1 \; c_{23} \; + \; n_y \; c_{23} \; s_1 \; + \; n_z \; s_{23}))$$

$$\theta_6 = atan2(-y_4, -y_5) \tag{4.6f}$$

Donde $c_i = cos\theta_i$, $s_i = sen\theta_i$, $c_{23} = cos(\theta_2 + \theta_3)$ y $s_{23} = sen(\theta_2 + \theta_3)$. Además s_x, s_y, s_z, n_x, n_y, n_z, a_x, a_y, a_z, p_x, p_y y p_z son elementos de la matriz *snap* (Ecuación 4.5). En el caso de E_1, E_2 y E_4 sus valores pueden ser 1 o -1, puesto que al resolver las ecuaciones escalares por el método de Paul el resultado está en función de una raíz cuadrada, por consecuencia, existen dos alternativas para el cálculo de θ_1, θ_2 y θ_4. Asimismo, si se calculan las posibles combinaciones de soluciones considerando estas variables, son ocho soluciones del modelo cinemático inverso para el robot en cuestión.

4.2 Modelado de piezas virtuales del robot

En el modelado geométrico en software CAD, la representación de un objeto físico es convertida a formato digital, lo cual genera información matemática y analítica inherente al propio objeto, que repercute de manera directa en la exactitud de cálculos así como en la forma de visualización del objeto en pantalla. Dicho esto, es justo señalar que la precisión en las medidas de los modelos de las piezas que conforman al robot virtual es trascendental en los sistemas de PFL, pues una diferencia entre el robot virtual y el robot físico ocasionaría discordancias en su implementación.

Cabe añadir que en el proceso de PFL, está planteada una simulación del robot ejecutando tareas de soldadura. Esta simulación se efectuará en el ambiente virtual de OpenGL® indicado en la Sección 3.3. Por lo que se requiere realizar el diseño virtual del robot Fanuc.

Como en el caso de los objetos virtuales de la Sección 3.2, los archivos CAD recién referidos son editados en el software de modelado SolidWorks™ de tal forma que cumplan a cabalidad con las especificaciones en cuanto a las medidas de los modelos y de la pose de sus marcos de referencia, precisiones que se hacen en algunos planos durante esta sección. Es preciso señalar que tal robot virtual está compuesto por 7 piezas. Es entonces que a continuación se indican las características más importantes de tales piezas y se presentan también sus modelos fabricados en SolidWorks™:

a) Base o eslabón 0.

A partir de esta pieza se emplaza al robot como tal y es la única pieza que no admite rotaciones una vez emplazado el robot.

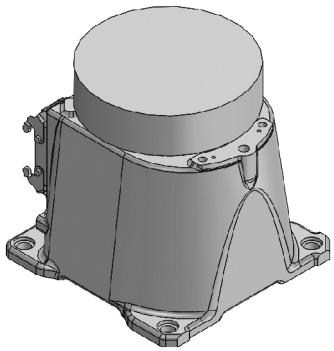


Figura 4.5. Modelo de la base del robot en SolidWorks™.
(*Vista isométrica*)

b) Eslabón 1.

Dado que el diseño del robot Fanuc se asemeja al cuerpo humano, análogamente se tiene que esta pieza corresponde a la cintura de una persona y permite el giro del tronco de la misma; por lo que en el robot al girarse esta pieza se giran todas las piezas subsecuentes.

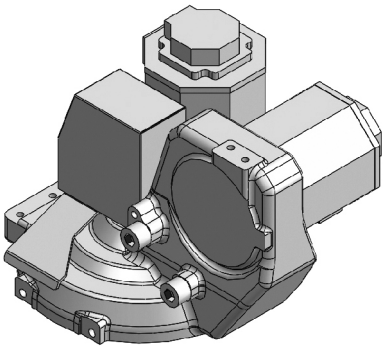


Figura 4.6. Modelo del eslabón 1 del robot en SolidWorks™.
(*Vista isométrica*)

c) Eslabón 2.

Debido al carácter antropomórfico del robot Fanuc, esta pieza corresponde al brazo humano y su rotación al hombro. Esta pieza corresponde al eslabón de mayor longitud del robot y al rotar su articulación correspondiente permite al manipulador alcanzar varios puntos dentro de un amplio radio.

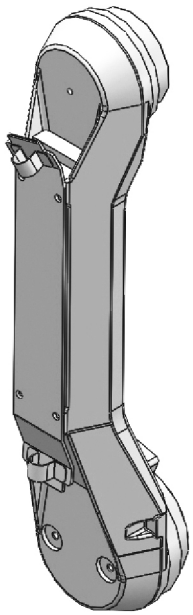


Figura 4.7. Modelo del eslabón 2 del robot en SolidWorks™.
(*Vista isométrica*)

d) Eslabón 3.

Considerando la correspondencia existente entre las piezas del robot Fanuc con las partes del cuerpo humano, la pieza en cuestión atañe al codo.

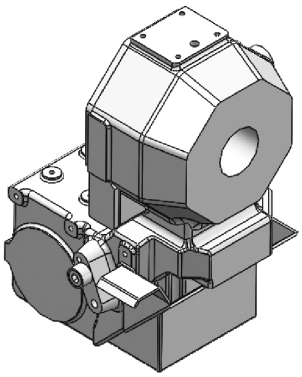


Figura 4.8. Modelo del eslabón 3 del robot en SolidWorks™.
(*Vista isométrica*)

e) Eslabón 4.

Representa al antebrazo de acuerdo a la correlación del robot Fanuc y la anatomía humana y su giro afecta a la muñeca. Dentro de la estructura del robot, a la

articulación que gira a esta pieza se le considera parte de la muñeca del robot que permite el movimiento del órgano terminal en el espacio de trabajo.

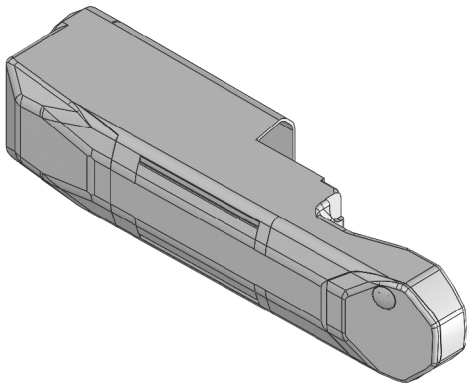


Figura 4.9. Modelo del eslabón 4 del robot en SolidWorks™.
(*Vista isométrica*)

f) Eslabón 5.

También forma parte de la muñeca y hace el movimiento de elevación del órgano terminal. Haciendo la analogía propia en este ámbito, permite la flexión de la muñeca.

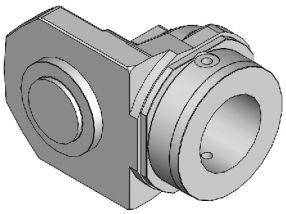


Figura 4.10. Modelo del eslabón 5 del robot en SolidWorks™.
(*Vista isométrica*)

g) Eslabón 6.

Por similitud con la muñeca humana, a esta parte le corresponde directamente los movimientos de rotación del órgano terminal sobre su propio eje. En esta pieza va adherida la antorcha de soldadura.

Figura 4.11. Modelo del eslabón 6 del robot en SolidWorks™.
(*Vista isométrica*)

En el mismo sentido que los objetos que conforman la escena virtual, los modelos geométricos de las piezas del robot virtual poseen las mismas dimensiones que las del robot real. Para explicitar lo anterior, la Tabla 8 y de la Figura 4.12 a la Figura 4.18 despliegan información referente a las dimensiones de los modelos que representan las piezas virtuales, pretendiendo identificar claramente sus dimensiones generales.

Tabla 8. Dimensiones en *cm* de los modelos de las piezas del robot virtual.

Modelo	*w*	*h*	*d*
Base	28.30	27.80	28.52
Eslabón 1	31.57	27.17	37.10
Eslabón 2	18.68	75.10	16.43
Eslabón 3	24.96	35.18	24.97
Eslabón 4	15.80	10.00	51.09
Eslabón 5	10.20	10.00	13.93
Eslabón 6	7.30	7.30	1.25

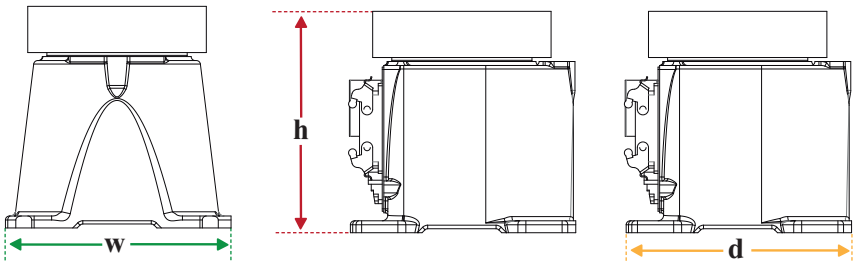


Figura 4.12. Planos con las dimensiones generales de la base del robot.
((a) en *vista derecha*; (b) y (c) en *vista frontal*)

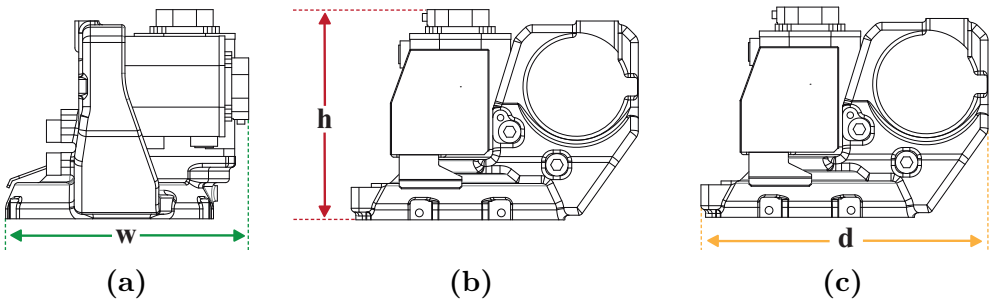


Figura 4.13. Planos con las dimensiones generales del eslabón 1 del robot. ((a) en *vista derecha*; (b) y (c) en *vista frontal*)

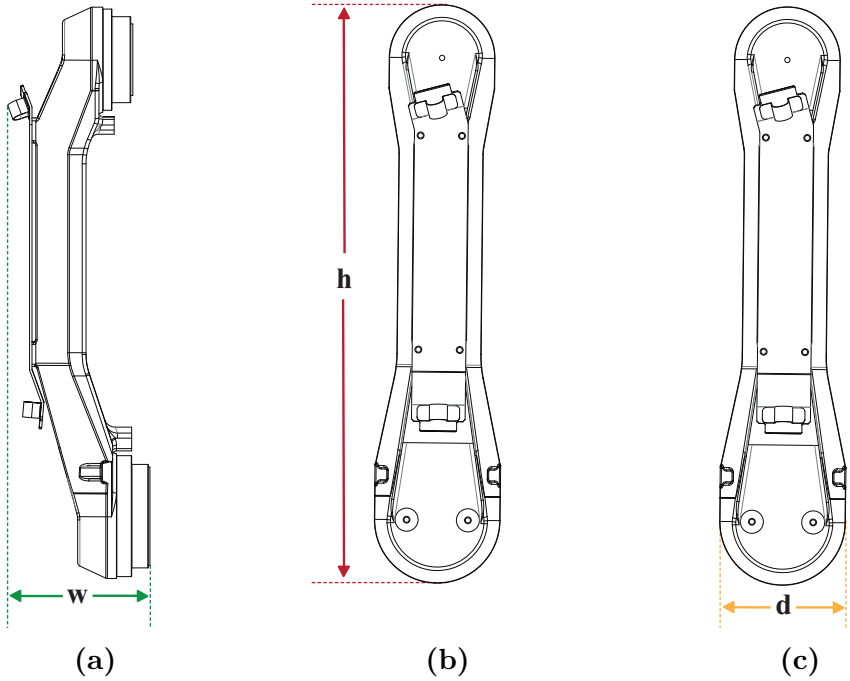


Figura 4.14. Planos con las dimensiones generales del eslabón 2 del robot. ((a) en *vista derecha*; (b) y (c) en *vista frontal*)

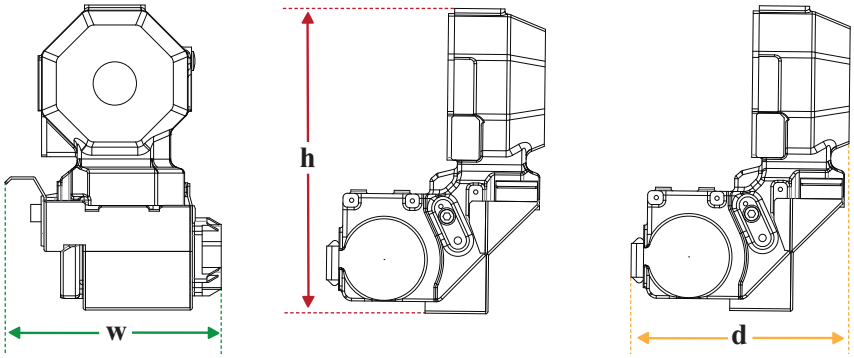


Figura 4.15. Planos con las dimensiones generales del eslabón 3 del robot. ((a) en *vista derecha*; (b) y (c) en *vista frontal*)

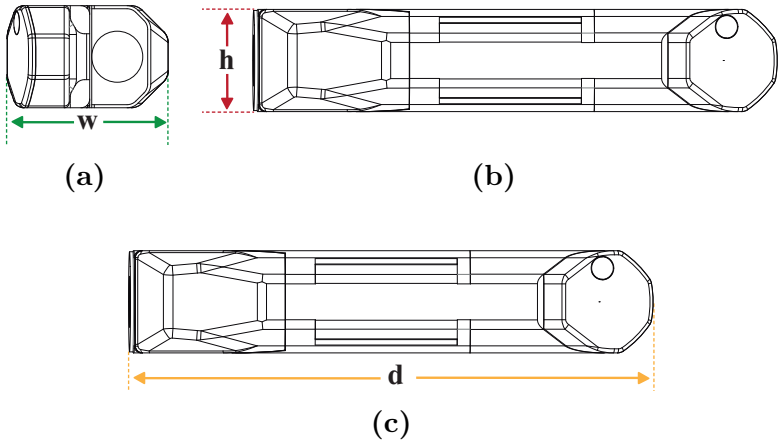


Figura 4.16. Planos con las dimensiones generales del eslabón 4 del robot. ((a) en *vista derecha*; (b) y (c) en *vista frontal*)

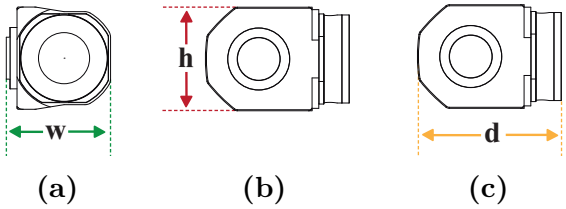


Figura 4.17. Planos con las dimensiones generales del eslabón 5 del robot. ((a) en *vista derecha*; (b) y (c) en *vista frontal*)

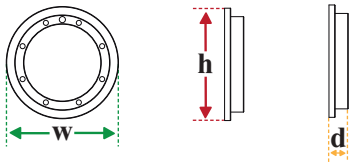


Figura 4.18. Planos con las dimensiones generales del eslabón 6 del robot. ((a) en *vista derecha*; (b) y (c) en *vista frontal*)

Debido a que los modelos geométricos en cuestión son exportados al ambiente virtual considerando el proceso indicado en la Figura 3.17, el formato en el cual han sido generados tales modelos es STL, el cual es un requisito indispensable en esta metodología, ya que a partir de este tipo de formato se trabaja para incluir los modelos tridimensionales dentro del sistema. En la Tabla 9 se proporciona información relativa a las mallas triangulares que dan forma a las piezas.

Tabla 9. Mallas triangulares de los modelos de las piezas del robot virtual.

Pieza	N° de vértices	N° de triángulos
Base	3,354	1,118
Eslabón 1	8,382	2,794
Eslabón 2	9,456	3,152
Eslabón 3	7,344	2,448
Eslabón 4	8,022	2,674
Eslabón 5	4,260	1,420
Eslabón 6	4,875	1,625

Para articular convenientemente los modelos de estas piezas virtuales y dar forma al robot Fanuc como tal, es necesario echar mano de la metodología de Denavit-Hartenberg para asignar un sistema de coordenadas ligado a cada modelo, por lo que tanto la posición como la orientación de los marcos de referencia de los modelos se determinó mediante esta metodología. En la Figura 4.19 se observa al robot Fanuc articulado con los modelos de las piezas virtuales, indicando también los marcos de referencia de acuerdo al esquema cinemático validado por la convención

de Denavit-Hartenberg ilustrado en la Figura 4.4. La posición de tales marcos se ha identificado con un círculo amarillo. Cabe aclarar que el modelado de estas piezas es individual y exclusivo, por tal motivo, la posición de los marcos de referencia calculados están en función del modelo geométrico en cuestión.

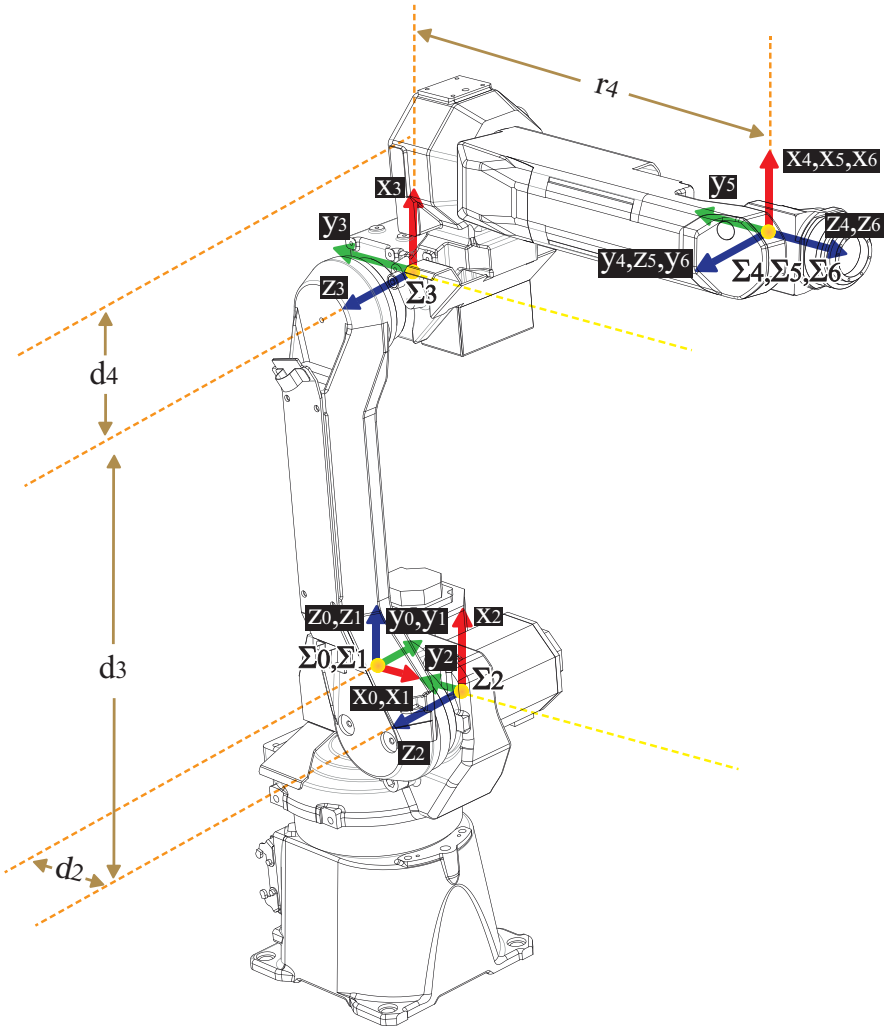


Figura 4.19. Esquema cinemático en el robot Fanuc articulado.
(*Vista dimétrica*)

Donde d_2 = 15 cm, d_3 = 60 cm, d_4 = 20 cm y r_4 = 64 cm.

De forma análoga al procedimiento que se usa para determinar la posición de los marcos de referencia de los objetos virtuales de la Sección 3.2, se calcula la posición de los sistemas de coordenadas de los modelos geométricos de estas piezas. Es entonces,

que la posición de los marcos de referencia deseados de cada modelo, nombrados como Σ_i, donde i es el número de eslabón, se definen con respecto a su correspondiente marco auxiliar, denominado Σ_{Oi}, cuya posición y orientación se define de igual manera que los marcos alternos de los objetos virtuales previamente señalados.

Cabe señalar que las posiciones de Σ_i se determinaron unas veces atendiendo a las dimensiones físicas del robot (Figura 4.2), otras veces con la ayuda de SolidWorks™ y otras basándose en el esquema cinemático (Figura 4.19).

4.2.1 Base

El primer modelo a tratar es el de la pieza denominada base o eslabón 0. Para este modelo, su sistema de coordenadas es Σ_0. Con base en las dimensiones físicas del robot se tiene que la distancia desde Σ_{O0} a la posición de Σ_0 en términos de y_{O0} es de B_0 = 45 cm. Por su parte, los valores de las distancias de Σ_0 con respecto al marco auxiliar del modelo en función de x_{O0} y z_{O0} se calcularon utilizando SolidWorks™ una vez que se ubicó Σ_0 en el centro de la tapa circular ubicada en la parte superior de la base, dando como resultado A_0 = 14 cm y C_0 = 14.12 cm respectivamente. Con las medidas calculadas previamente, es evidente mencionar que Σ_0 queda fuera de la base como tal. Las Figuras 4.20 y 4.21 ilustran lo explicado aquí.

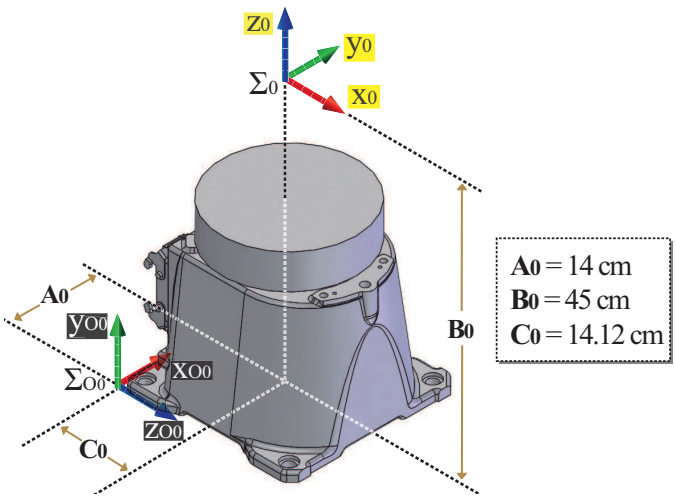


Figura 4.20. Σ_0 con respecto al marco auxiliar del modelo de la base. (*Vista isométrica*)

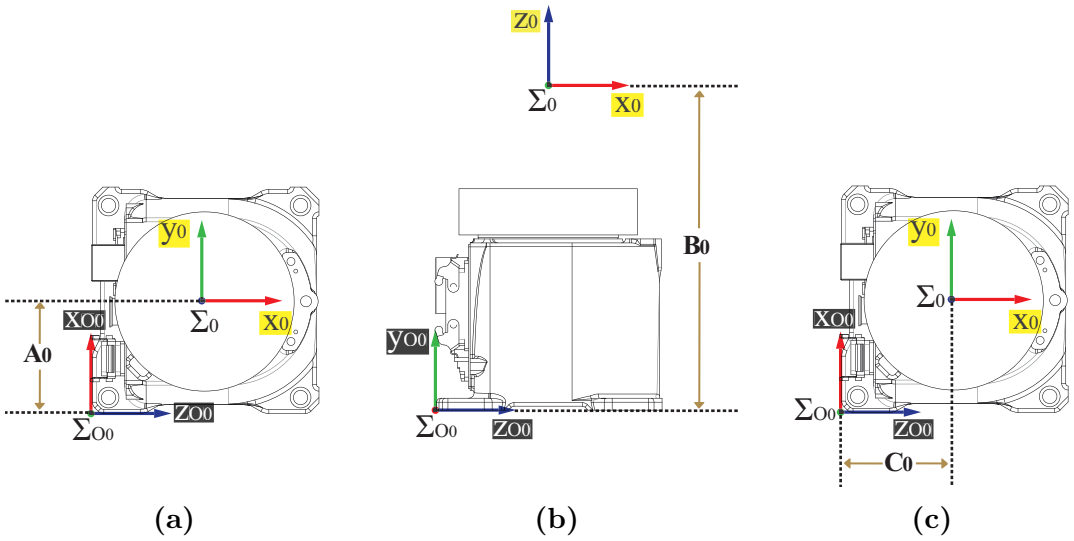


Figura 4.21. Distancias de Σ_0 con respecto al marco auxiliar del modelo. ((a) y (c) en *vista superior*; (b) en *vista frontal*)

4.2.2 Eslabón 1

Tomando en consideración a la Figura 4.19 se puede ver que el marco de referencia Σ_1 coincide tanto en posición como en orientación con el marco Σ_0. A partir de las dimensiones físicas del robot se tiene que la distancia desde el pie de la base del robot a la ubicación de Σ_1 es de 45 cm y dado que la altura (h) de la base es 27.8 cm, entonces la distancia de Σ_1 con respecto a Σ_{O1} en función de y_{O1} es de B_1 = 17.2 cm. Mientras que las posiciones de Σ_1 con respecto a Σ_{O1} en términos de x_{O1} y z_{O1} surgen cuando se localiza el centro de la parte cuasicircular del eslabón 1 que está en contacto con la base y que permite la rotación del mismo con respecto al eje z_1, tal procedimiento de localización se hizo con SolidWorks™, resultando A_1 = 13.78 cm y C_1 = 14.1 cm respectivamente (Figura 4.22). En los incisos (a) y (c) de la Figura 4.23 se observa la parte del eslabón 1 señalada recientemente, donde en su centro se ubica a Σ_1.

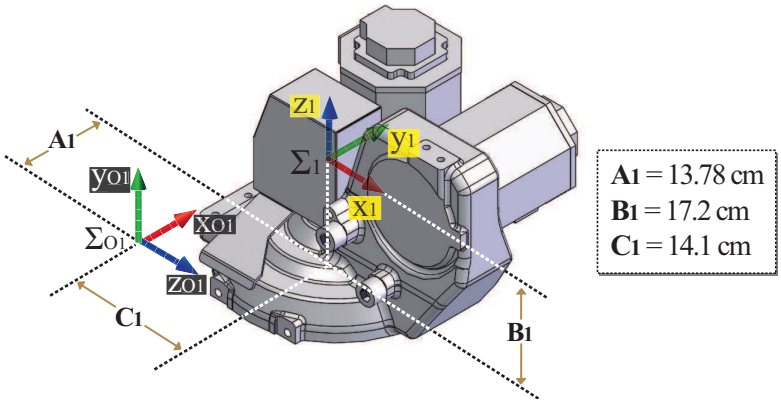


Figura 4.22. Σ_1 con respecto al marco auxiliar del modelo del eslabón 1. (*Vista isométrica*)

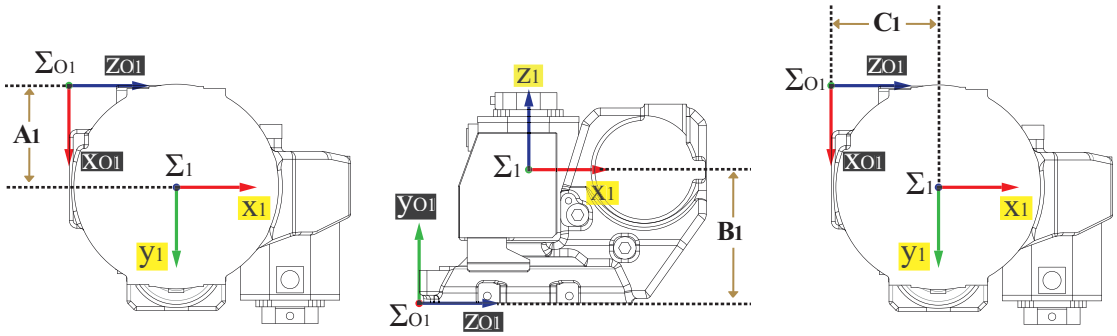


Figura 4.23. Distancias de Σ_1 con respecto al marco auxiliar del modelo. ((a) y (c) en *vista inferior*; (b) en *vista frontal*)

Comenzando con este modelo y prosiguiendo con los que aún faltan por tratar en este documento, es importante señalar que dichos modelos pueden rotar con respecto al eje z_i de cada marco de referencia Σ_i, razón por la cual, estos marcos son nombrados también como puntos de pivote.

4.2.3 Eslabón 2

Siguiendo con la metodología para calcular los marcos de referencia deseados en los modelos las piezas del robot virtual, se tiene ahora que para el eslabón 2 se ha calculado la posición de su sistema de coordenadas (Σ_2) mediante SolidWorks™.

Las Figuras 4.24 y 4.25 dan constancia de la posición de Σ_2 con respecto a Σ_{O2}. Para esto, ha sido necesario determinar el lugar donde se ubicaría este punto de pivote, mismo que de acuerdo al esquema cinemático del robot se encuentra alineado con Σ_1, pero separado de éste por una distancia d_2 sobre el eje x_1.

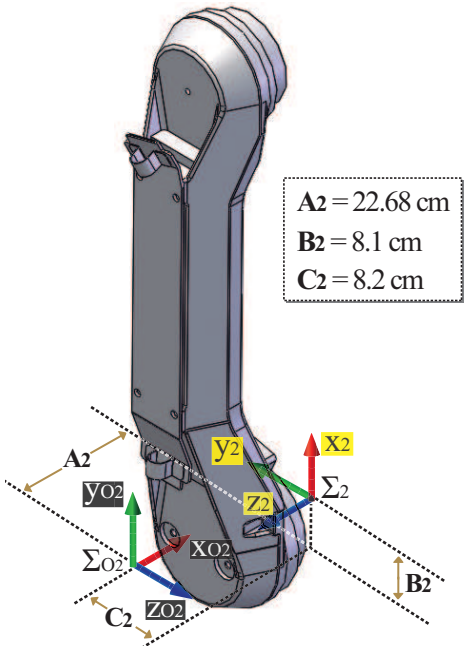


Figura 4.24. Σ_2 con respecto al marco auxiliar del modelo del eslabón 2. (*Vista isométrica*)

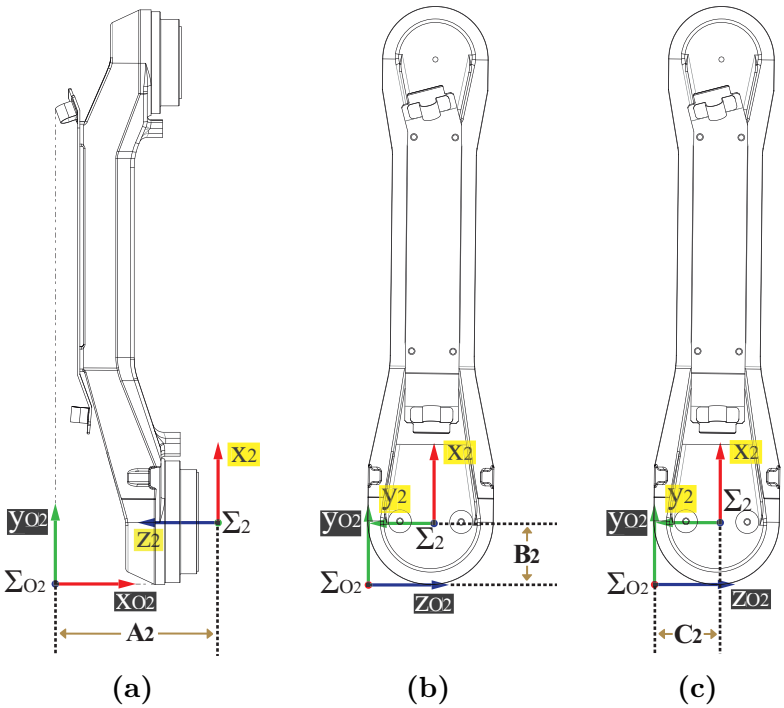


Figura 4.25. Distancias de Σ_2 con respecto al marco auxiliar del modelo.
((a) en *vista derecha*; (b) y (c) en *vista frontal*)

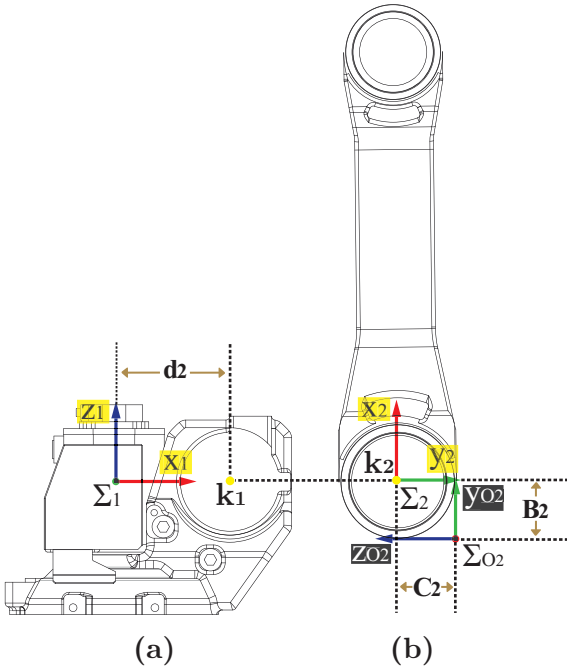


Figura 4.26. Distancias B_2 y C_2.
((a) en *vista frontal* y (b) en *vista posterior*)

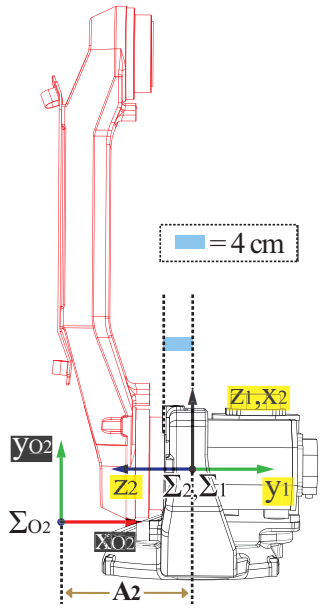


Figura 4.27. Distancia A_2.
(*Vista derecha*)

Para poder determinar la posición de Σ_2, en primer lugar se ha considerado la parte del eslabón 2 (k_2 en la Figura 4.26b) que contacta con una parte del eslabón 1 (k_1 en la Figura 4.26a) sobre la cual efectúa rotaciones el eslabón 2 en el eje z_2. Ambas partes tienen forma cuasicircular y se dibuja en ellas sendos puntos amarillos que simbolizan sus centros. Estos centros deben estar alineados al articularse adecuadamente el robot. Al medir con SolidWorks™ la posición del punto de pivote del eslabón 2 tenemos las distancias de Σ_2 con respecto a Σ_{O2} en función de y_{O2} y z_{O2}, las cuales son B_2 = 8.1 cm y C_2 = 8.2 cm.

En cuanto a la longitud existente entre Σ_2 y el marco auxiliar del modelo del modelo en términos de x_{O2} se tiene que considerar que sobre el plano de la Figura 4.27, Σ_2 coincide en posición con Σ_1. Asimismo, para calcular esta longitud se toma en cuenta la dimensión w del eslabón 2 indicada en la Tabla 8, igual a 18.68 cm y sumarle la distancia encontrada mediante SolidWorks™ sobre el propio eje x_{O2} entre el extremo del eslabón 2 que roza con el eslabón 1, cuya magnitud es igual a 4 cm y que en dicha figura se distingue con color cian, dando como resultado el valor de A_2 = 22.68 cm. Se le ha dado coloración diferente al eslabón 2 para diferenciarlo del eslabón 1 y

poder establecer límites visuales entre las piezas. Como se observa, la ubicación de Σ_2 se encuentra al exterior del eslabón 2.

4.2.4 Eslabón 3

En el caso de la definición del sistema de referencia del modelo del eslabón 3, también se ha requerido de SolidWorks™ para este fin. Con el apoyo de este software se ha localizado el marco de referencia Σ_3. Cabe mencionar que este marco se encuentra alineado con Σ_2 conforme a lo establecido en el esquema cinemático del robot, únicamente separados por una longitud d_3. Las Figuras 4.28 y 4.29 ilustran la posición de Σ_3 dentro del modelo.

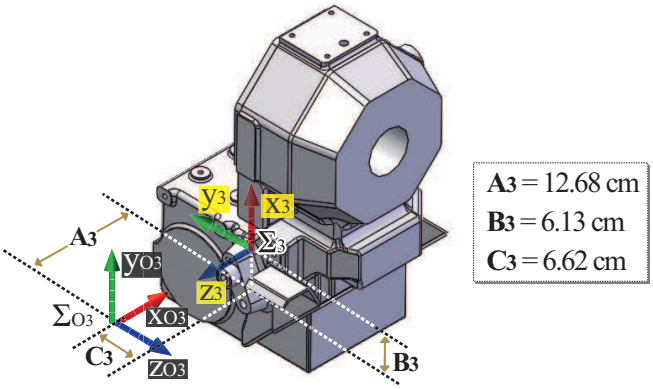


Figura 4.28. Σ_3 con respecto al origen del modelo del eslabón 3. (*Vista isométrica*)

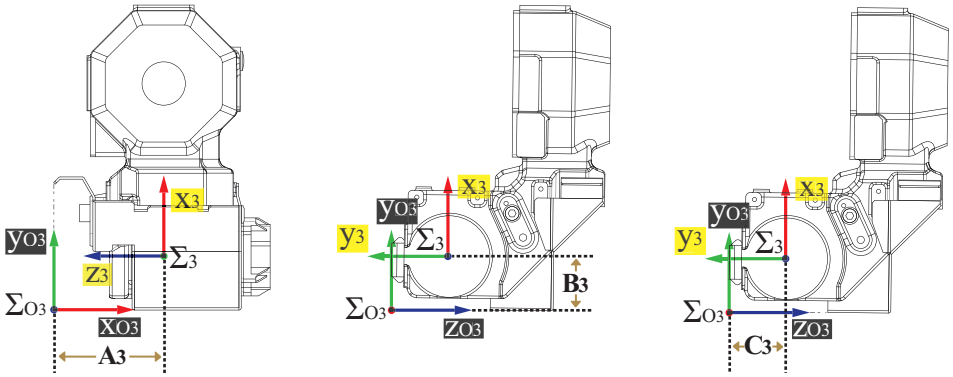


Figura 4.29. Distancias de Σ_3 con respecto al marco auxiliar del modelo. ((a) en *vista derecha*; (b) y (c) en *vista frontal*)

Para determinar la posición de Σ_3 con respecto al marco auxiliar del modelo (Σ_{O3}), se tiene en cuenta que tal posición se localiza en la parte del eslabón 3 por medio de la cual efectúa sus movimientos de rotación sobre su eje z_3 (k_3 en la Figura 4.30b) y que está en fricción con una porción del eslabón 2 (k_2 en la Figura 4.30a). En la citada figura, se identifican estos sectores de ambas piezas, cuya forma es cuasicircular y se indica con sendos puntos amarillos el centro de tales sectores. Dichos centros se alinean una vez que el robot está articulado convenientemente. Por lo anterior expuesto y realizando las mediciones correspondientes dentro de SolidWorks™, se tiene que B_3 = 6.13 cm es la distancia de Σ_3 con respecto a Σ_{O3} sobre el eje y_{O3} mientras que C_3 = 6.62 cm la longitud sobre el eje z_{O3}.

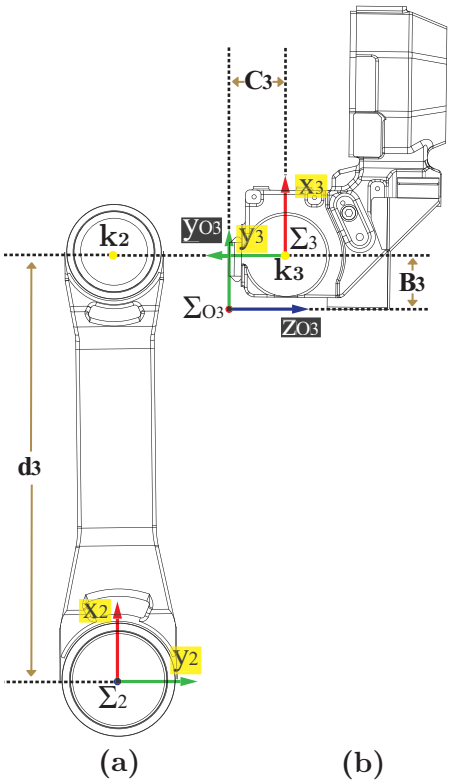


Figura 4.30. Distancias B_3 y C_3.
((a) en *vista posterior* y (b) en *vista frontal*)

Se ha tomado en cuenta la Figura 4.31 para el cálculo de la distancia de Σ_3 con relación a Σ_{O3} en lo que respecta al eje x_{O3}, llamada A_3. En esta figura, además de desplegar una vista lateral derecha del robot con los eslabones 1, 2 y 3 articulados

pertinentemente, se indica que los marcos de referencia Σ_2 y Σ_3 se encuentran alineados, respetando el esquema cinemático del robot. A efecto de una mayor claridad visual, el eslabón 2 se ha pigmentado de un color diferente a los eslabones 1 y 3. Ahora bien, el valor de A_3 calculado es igual a 12.68 cm, resultado de la suma acumulada de varias distancias, determinadas con SolidWorks™, que van sobre el eje x_{O3} desde Σ_{O3} hasta Σ_3. La primera de estas distancias es la que comprende desde el origen de Σ_{O3} hasta la zona de contacto entre el eslabón 2 y el eslabón 3, cuyo valor es de 6.5 cm y se distingue en la figura con color rosa. La siguiente distancia corresponde a la parte faltante del eslabón 2 para alcanzar la parte del eslabón 1 con la que hace contacto, pintada de color naranja en la figura y que es igual a 2.18 cm. Finalmente, la distancia de 4 cm, teñida de color cian, previamente señalada en la Figura 4.26 y explicada también en la sección a la que pertenece. Definido lo anterior se está en la posibilidad de cumplir con la alineación requerida de Σ_3 con respecto a Σ_2.

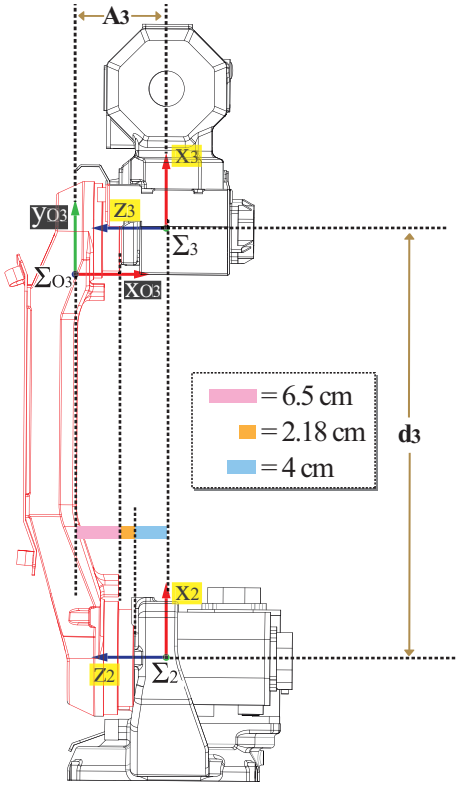


Figura 4.31. Distancia A_3.
(*Vista derecha*)

4.2.5 Eslabón 4

En lo particular, el sistema de coordenadas ligado al modelo que representa al eslabón 4, llamado Σ_4, es el que más se aleja de su marco auxiliar (Σ_{O4}), esto debido a que el robot Fanuc posee una muñeca de ejes concurrentes y por tal, los últimos 3 marcos de referencia coinciden en una posición más cercana al órgano terminal, pero en este caso, más distante de Σ_{O4}. En las Figuras 4.32 y 4.33 se muestra a Σ_4 con respecto a Σ_{O4}.

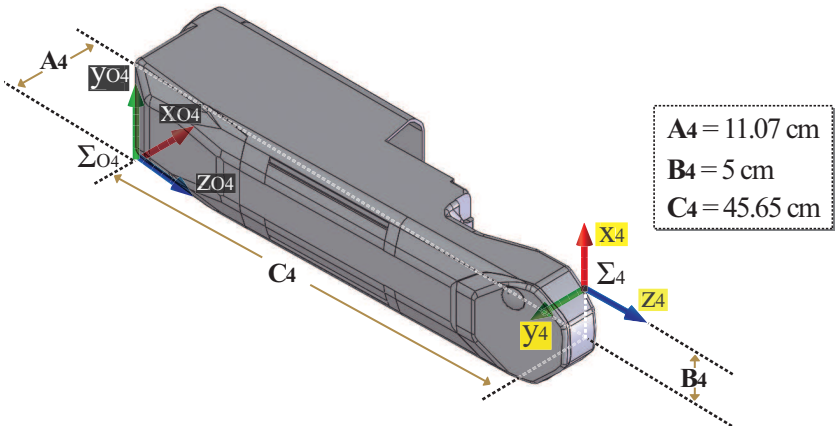


Figura 4.32. Σ_4 con respecto al marco auxiliar del modelo del eslabón 4. (*Vista isométrica*)

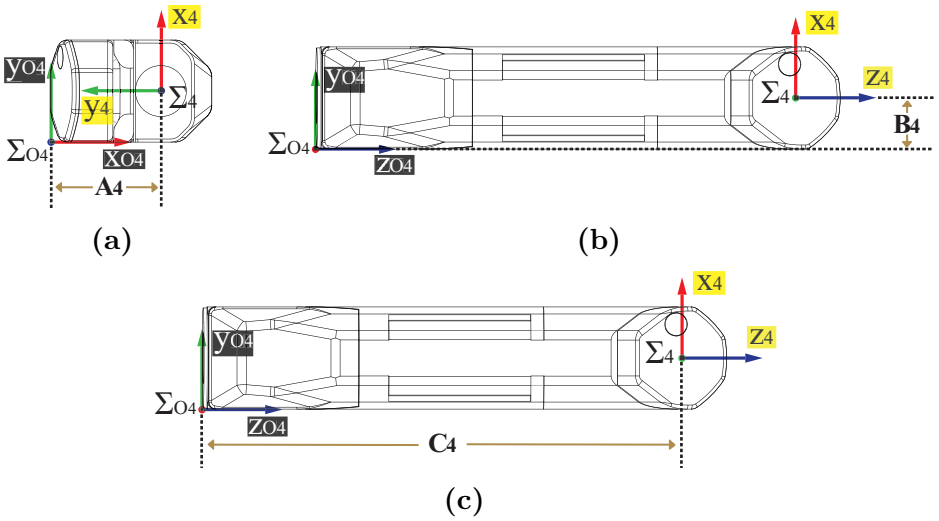


Figura 4.33. Distancias de Σ_4 con respecto al marco auxiliar del modelo. ((a) en *vista derecha*; (b) y (c) en *vista frontal*)

El cálculo de las distancias comprendidas entre Σ_4 y a Σ_{O4} sobre sus ejes x_{O4} e y_{O4} conlleva la definición del punto de pivote del eslabón 4, mismo que se ilustra en la Figura 4.34 y que se ha posicionado en la parte de tal eslabón que toca con su predecesor, en otras palabras, en la figura citada se observa que la posición de Σ_4 se ha establecido a una distancia d_4 con respecto a Σ_3 sobre su eje x_3. La distancia d_4 proviene del esquema cinemático del robot. Cada una de estas piezas posee un orificio circular y es justo en el centro del orificio del eslabón 4 (k_4 en la Figura 4.34b) donde se ha posicionado a Σ_4. Es preciso indicar que en las dos vistas indicadas en la figura en cuestión, deben coincidir en posición los centros de los orificios de ambas piezas una vez articulado el robot. Una vez definida esta posición, con la ayuda de SolidWorks™ se hicieron las mediciones de las distancias correspondientes, obteniendo $A_4 = 11.07$ cm y $B_4 = 5$ cm.

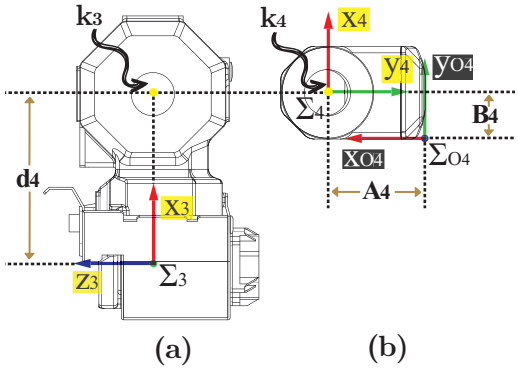


Figura 4.34. Distancias A_4 y B_4.
((a) en *vista derecha* y (b) en *vista izquierda*)

Para poder establecer la distancia entre el origen del modelo y Σ_4 sobre el eje z_{O4} también se ha recurrido a SolidWorks™ para hacer mediciones al respecto que permitan el cálculo de esta magnitud. Es así que, se ha medido la distancia entre Σ_3 y la parte del eslabón 3 (pieza de color rojizo) que fricciona con el eslabón 4, quedando indicada tal distancia como el segmento teñido de rosa en la Figura 4.35, dando como resultado 18.35 cm. Ahora bien, conociendo que el valor de $r_4 = 64$ cm establecido en el esquema cinemático se obtiene $C_4 = 45.65$ cm, estableciéndose Σ_4 dentro de los límites del propio modelo.

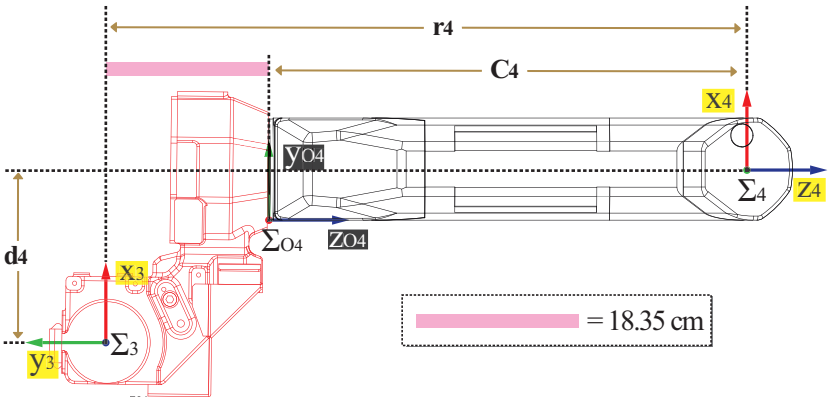


Figura 4.35. Plano que indica la distancia C_4.
(*Vista frontal*)

4.2.6 Eslabón 5

Hemos llegado al punto donde hay que describir el procedimiento seguido para determinar la posición del sistema de coordenadas del modelo del eslabón 5 o nombrado también Σ_5. Este marco se define con respecto al marco auxiliar del modelo (Σ_{O5}) y los esquemas de lo dicho anteriormente se presentan en las Figuras 4.36 y 4.37.

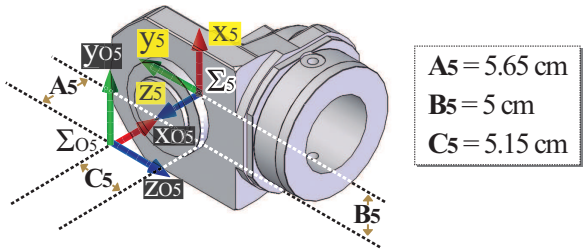


Figura 4.36. Σ_5 con respecto al marco auxiliar del modelo del eslabón 5.
(*Vista isométrica*)

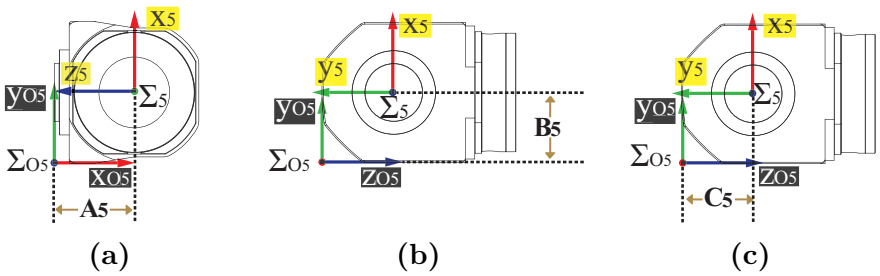


Figura 4.37. Distancias de Σ_5 con respecto al marco auxiliar del modelo.
((a) en *vista derecha*; (b) y (c) en *vista frontal*)

La Figura 4.38 presenta al eslabón 4 con un tinte rojizo para diferenciarlo del eslabón 5, el cual está unido al primero y con una coincidencia en la posición de Σ_4 y Σ_5. Cabe indicar que ambos eslabones contienen un orificio circular en cuyos centros (punto amarillo) se establece su respectivo sistema de coordenadas, como lo expone también dicha figura. De tal modo que una vez definida la posición de Σ_5, ha sido posible la determinación de las distancias de éste con respecto a Σ_{O5}, en función de x_{O5} e y_{O5}, resultando en este plano A_5 = 5.65 cm y B_5 = 5 cm respectivamente, mismas que se determinaron mediante SolidWorks™.

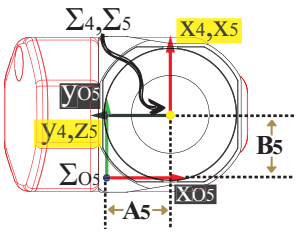


Figura 4.38. Distancias A_5 y B_5.
(*Vista derecha*)

A efecto de calcular C_5, que representa la distancia de Σ_5 con respecto a Σ_{O5} sobre el eje z_{O5} se presenta la Figura 4.39. En tal figura se señala con j_5 la zona circular del eslabón 5 que contacta con la parte señalada con j_4 en el eslabón 4. Dado que Σ_4 previamente ha sido establecido y que conforme al esquema cinemático del robot, Σ_5 debe coincidir con su predecesor, la posición de Σ_5 deseada se ubica en el centro del círculo, apuntado por j_5 e indicado con un punto amarillo, obteniendo así C_5 = 5.15 cm como la distancia deseada en este plano. Finalmente, de acuerdo a lo detallado aquí se tiene que Σ_5 se ubica dentro de los límites del eslabón 5. Ya que se ha articulado correctamente el robot, Σ_4 y Σ_5 deben coincidir en posición.

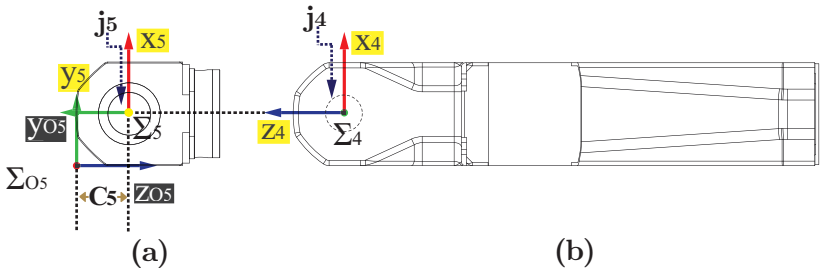


Figura 4.39. Distancia C_5.
((a) en *vista frontal* y (b) en *vista posterior*)

4.2.7 Eslabón 6

El último modelo de pieza que conforma al robot Fanuc es el eslabón 6. La definición de la posición del marco de referencia ligado a tal modelo, nombrado como Σ_6 se ha llevado a cabo de forma similar a como se hizo en los modelos previos. En ese sentido, ha sido conveniente establecer la distancia de Σ_6 con respecto a Σ_{O6}. Se tiene a SolidWorks™ como la herramienta principal para el cálculo de las magnitudes que componen dicha distancia. Estos marcos se exhiben en las Figuras 4.40 y 4.41.

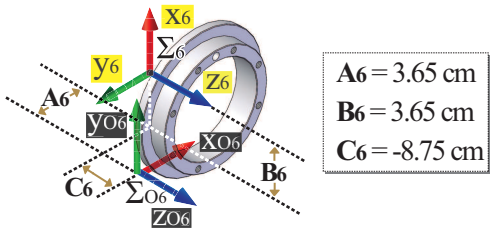


Figura 4.40. Σ_6 con respecto al marco auxiliar del modelo del eslabón 6.
(*Vista isométrica*)

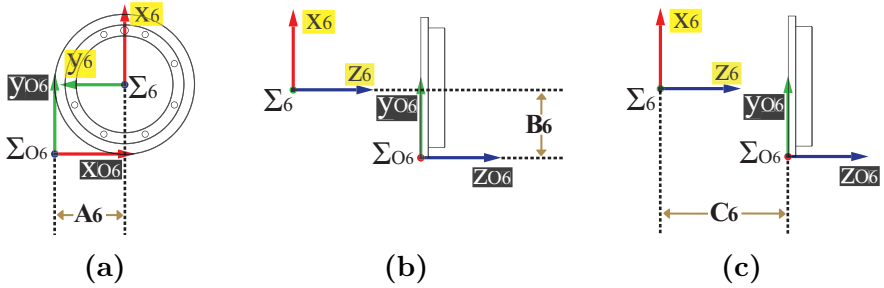


Figura 4.41. Distancias de Σ_6 con respecto al marco auxiliar del modelo.
((a) en *vista derecha*; (b) y (c) en *vista frontal*)

En un plano con la vista lateral derecha de los eslabones 5 y 6, como el representado por la Figura 4.42a ha sido posible determinar A_6 y B_6, que representan las distancias de Σ_6 con respecto a Σ_{O6} sobre los ejes x_{O6} e y_{O6}. En esta figura el eslabón 6 se distingue del eslabón 5 porque éste último ha sido pintado de un tono rojizo. Una peculiaridad de éste es un orificio circular, donde previamente, en cuyo centro se ha colocado Σ_5. En esta misma posición, considerando el esquema cinemático del robot se localiza Σ_6 y dado que el eslabón 6 posee un orificio circular al igual que su predecesor, es justamente en el centro de dicho orificio (punto amarillo) donde se localiza el marco de referencia deseado. Por lo que después de hacerse las mediciones respectivas con SolidWorks™ han quedado $A_6 = 3.65$ cm y $B_6 = 3.65$ cm.

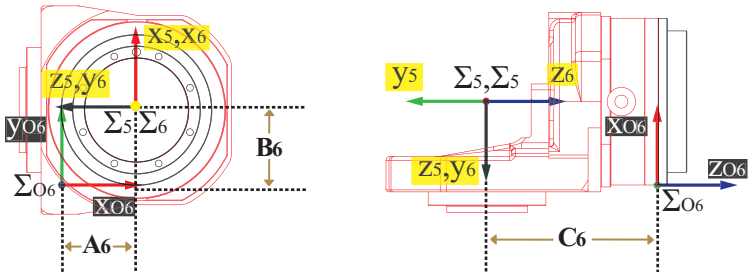


Figura 4.42. Planos que indican las distancias A_6, B_6 y C_6. ((a) en *vista derecha* y (b) en *vista superior*)

Con las dos distancias previamente calculadas, sólo falta el cálculo de la distancia de Σ_6 con respecto a Σ_{O6} en términos de z_{O6}. Para este fin, se ha requerido también SolidWorks™. Con este software, se ha medido la distancia que hay de Σ_5 hasta el límite del eslabón 5 que contacta con el eslabón 6. En la Figura 4.42b tal distancia representa la longitud deseada, que en este caso es $C_6 = -8.75$ cm. El signo negativo de esta distancia indica que hay que recorrerla en sentido contrario al que apunta z_{O6}. Entonces, se hace notar que Σ_6 queda fuera de los límites del eslabón 6.

En esta sección, se han definido la posición de los marcos de referencias de los modelos de las piezas que constituyen al robot Fanuc. No obstante, además de las piezas virtuales señaladas también se requiere el modelo geométrico del órgano terminal del robot, que en este caso es una antorcha de soldadura.

4.2.8 Antorcha de soldadura

Aunque en la Sección 3.2 ya se definió un modelo de antorcha de soldadura, los parámetros utilizados para la ubicación del marco de referencia de ese modelo de antorcha en especial no corresponden a los requeridos para determinar su posición dentro de la cadena cinemática que representa el robot Fanuc. Es justo señalar, que el modelo geométrico de la antorcha definida en la Sección 3.2 es el mismo que el modelo de la antorcha que cumple la función de órgano terminal en el robot virtual, únicamente difiere en la posición de su marco de referencia. Lo anterior se presenta en las Figuras 4.43 y 4.44. Esta antorcha, como más adelante se verá, tomará las poses que se almacenen con la otra antorcha, la cual es aquella que se manipula con la interfaz háptica y usará estas poses como entradas en el cálculo de la cinemática inversa para que, mediante una simulación, el robot ejecute las tareas deseadas.

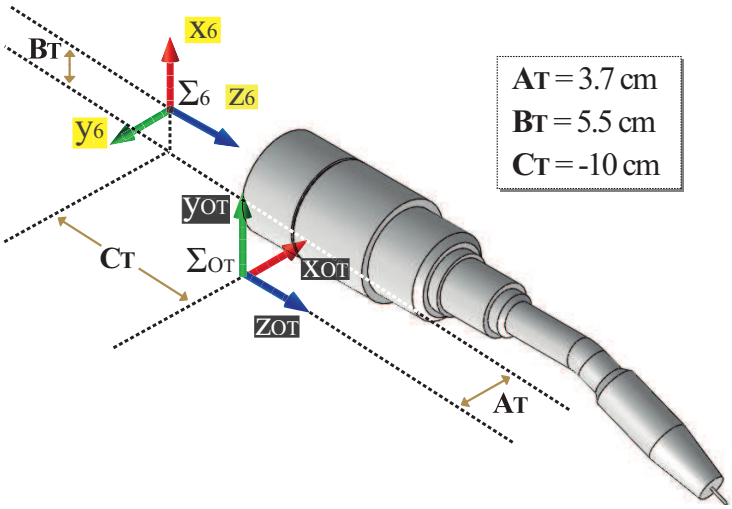


Figura 4.43. Σ_6 con respecto al marco auxiliar del modelo de la antorcha. (*Vista isométrica*)

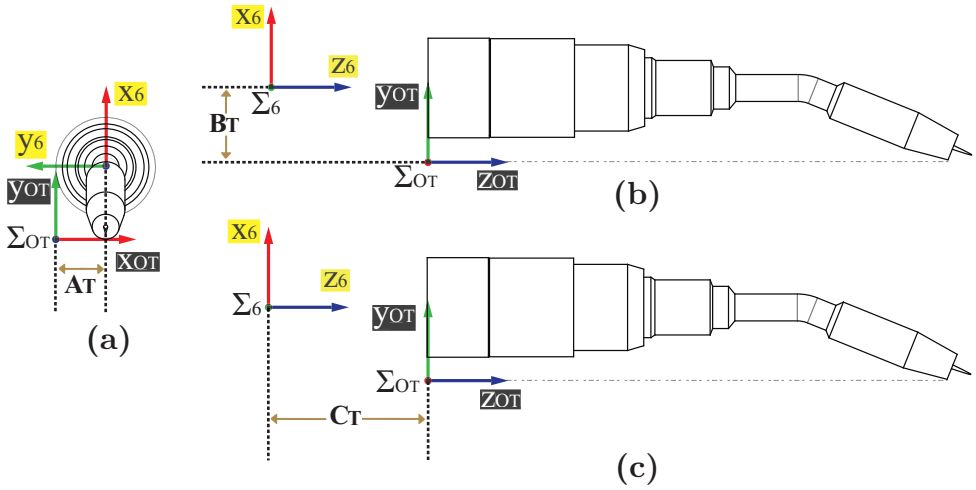


Figura 4.44. Distancias de Σ_6 con respecto al marco auxiliar del modelo. ((a) en *vista derecha*; (b) y (c) en *vista frontal*)

Dado que el esquema cinemático carece de un marco de referencia específico para la antorcha, se ha tomado a Σ_6 para que funja como el marco de la misma. Cabe decir que la base de la antorcha posee una forma circular en el plano de la Figura 4.45b, dentro de la cual, en su centro se ha ubicado Σ_6, indicado con un punto amarillo y con k_T en la Figura 4.45a, de tal forma que al rotar la antorcha lo haga alineada con el centro del eslabón 6, el cual se ha etiquetado con k_6 en la misma figura. Para conocer las medidas deseadas se ha utilizado SolidWorks™, permitiendo obtener $A_T = 3.7$ cm y $B_T = 5.5$ cm como las distancias de Σ_6 con respecto al marco auxiliar del modelo sobre x_{OT} e y_{OT} respectivamente.

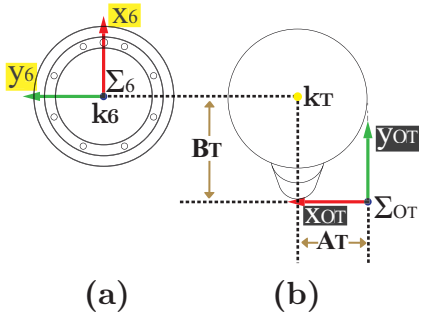


Figura 4.45. Planos que indican las distancias A_T y C_T. ((a) en *vista derecha*) y (b) en *vista izquierda*)

Basándose en la Figura 4.46 se puede hacer el cálculo de la distancia de Σ_6 con respecto a Σ_{OT} en función de z_{OT}. Para fines de distinción visual, el eslabón 6 ha sido pintado de un tono rojizo. Dicho lo anterior, desde una vista superior se observa que la distancia C_T es la suma acumulada de dos segmentos sobre el eje z_{OT}. El primero de ellos corresponde a la dimensión d del eslabón 6 indicada en la Tabla 8. En la figura aludida, esta distancia está representada por un segmento rectangular color naranja. El segundo segmento en cuestión pertenece a la distancia de Σ_5 a la zona de contacto del eslabón 5 con el eslabón 6, esta longitud ha sido indicada con un rectángulo color rosa (C_6). Es entonces que una vez que se recurren a las matemáticas nos queda que $C_T = -10$ cm, por lo que se tiene que ir en sentido contrario al que apunta z_{OT} para recorrer esta distancia. Como se observa, el marco de referencia de la antorcha queda al exterior de la misma.

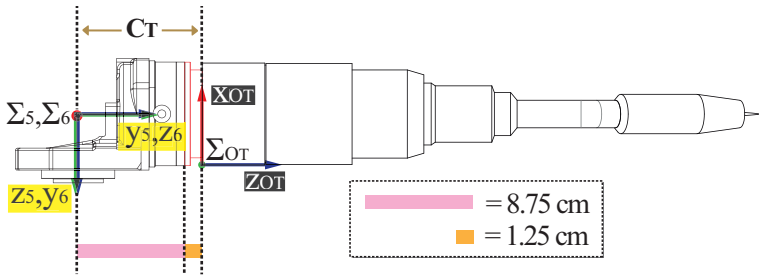


Figura 4.46. Plano que indica la distancia C_T.
(*Vista superior*)

Además de calcular la posición del punto de pivote que la antorcha requiere para desempeñarse adecuadamente dentro de la cadena cinemática relativa al robot Fanuc, todavía falta especificar la pose de la punta de la antorcha con respecto a Σ_6 (Figura 4.47). Esto se hace debido a que al momento de que el robot virtual ejecute la tarea deseada, la punta de la antorcha debe pasar por los puntos nodo generados con la interfaz háptica. De esta forma, en la simulación en el ambiente virtual, el manipulador recorre la trayectoria definida por el usuario y efectúa la tarea de soldadura con la punta de la antorcha.

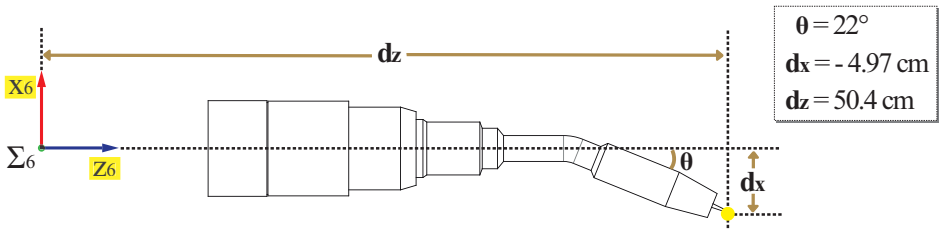


Figura 4.47. Plano con la posición de la punta de la antorcha con respecto a Σ_6. (*Vista frontal*)

La matriz ${}^{6}_{H}T$ contiene los parámetros de la antorcha con respecto a Σ_6 que se requieren para obtener las coordenadas operacionales de la punta de la misma. En la Figura 4.47 se observan tales parámetros; y a partir de ésta, la matriz ${}^{6}_{H}T$ queda conformada de la siguiente forma:

$$ {}^{6}_{H}T = \begin{bmatrix} c\theta & 0 & -s\theta & dx \\ 0 & 1 & 0 & 0 \\ s\theta & 0 & c\theta & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.7} $$

Donde $c\theta$ significa $cos\theta$ y $s\theta$ representa a $sen\theta$.

4.3 Simulación

En atención a lo expuesto en el proceso de PFL, es digno de mención que en la presente sección se va a pormenorizar la simulación del robot dentro de la escena virtual. Dicha etapa se encuentra en el cuarto lugar de tal proceso. A juicio de Neto *et al* [89] , las ventajas que se pueden obtener con la simulación son:

- Evaluación y validación del sistema en respuesta a eventos anormales.
- Detección de colisiones.
- Predicción de desempeño de calidad, productividad y eficiencia.
- Ajustar tiempos de producción.

- Dar respuestas a preguntas sobre la planificación de trayectorias, restricciones del espacio de trabajo y temas de coordinación con otros sistemas.
- Reducir el tiempo de inactividad incurrido por la programación en línea.
- Comprender qué tipo de modificaciones y mejoras deberían hacerse.
- Predecir el desempeño del robot sin quitarlo de la línea de producción durante la fase de programación.
- Probar programas de robot rápidamente, intuitivamente y en un ambiente seguro.
- Evaluación del desempeño: generación de rutas y planificación de trayectorias.

En este trabajo se lleva a cabo una simulación que reproduce un robot ejecutando tareas de soldadura dentro de un ambiente virtual. Ahora bien, para que dicha simulación pueda ejecutarse, en primera instancia es necesario señalar que existen algunas matrices que en conjunto con las matrices elementales han de permitir la reproducción visual de los movimientos del robot virtual.

4.3.1 Matriz de emplazamiento del robot

El emplazamiento del robot Fanuc dentro del ambiente virtual se hará con respecto a las placas a soldar; y la matriz que define lo anterior es ${}^{P}_{0}\boldsymbol{T}$. Específicamente corresponde a la matriz de transformación de la base del robot ($\boldsymbol{\Sigma}_0$) con respecto al marco de las placas ($\boldsymbol{\Sigma}_P$); y dado que el robot virtual se dibuja a partir de la base del robot, la función de esta matriz es facilitar la colocación de $\boldsymbol{\Sigma}_0$ en diversas poses y consecuentemente el robot completo ha de presentar diferentes configuraciones en la escena virtual.

La configuración predeterminada de la matriz ${}^{P}_{0}\boldsymbol{T}$ es la que se muestra en la Ecuación 4.8. Con estos valores, dicha matriz permite una colocación del robot

virtual de tal manera que la base del mismo se encuentra como lo ilustra la Figura 4.48. A esta situación se le denomina emplazamiento en piso.

$$ {}^{P}_{0}\boldsymbol{T} = \begin{bmatrix} 1 & 0 & 0 & xa \\ 0 & 0 & 1 & ya \\ 0 & -1 & 0 & za \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.8} $$

Donde xa, ya y za corresponden a las distancias en cm de Σ_0 con respecto a Σ_P sobre sus ejes $\{x, y, z\}$ respectivamente.

Mediante la modificación de los valores de la matriz de rotación contenida en la matriz ${}^{P}_{0}\boldsymbol{T}$, la cual está indicada con color magenta en la Ecuación 4.8, es posible cambiar la orientación de la base del robot virtual. Un caso específico es en el que la base se encuentra suspendida en el techo, tal y como lo pone en evidencia la Figura 4.49. Por lo que a esta situación se le denomina emplazamiento en techo y la matriz que determina tal orientación es la siguiente:

$$ {}^{P}_{0}\boldsymbol{T} = \begin{bmatrix} 1 & 0 & 0 & xa \\ 0 & 0 & -1 & ya \\ 0 & 1 & 0 & za \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.9} $$

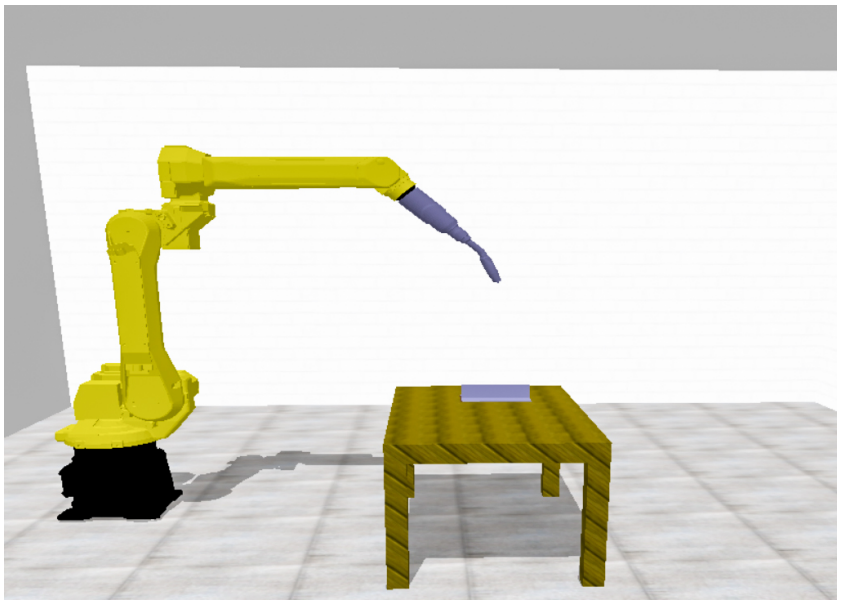


Figura 4.48. Emplazamiento en piso del robot en el ambiente virtual.

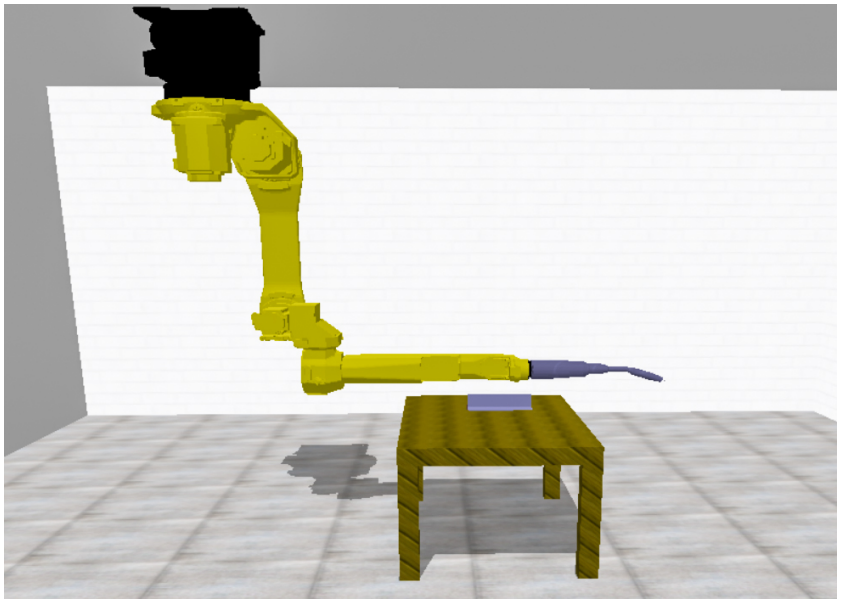


Figura 4.49. Emplazamiento en techo del robot en el ambiente virtual.

La simulación implica un proceso por medio del cual se representa una recreación de los movimientos del robot obtenidos mediante la programación de una secuencia de funciones, cuyos resultados definen las posiciones de los eslabones del mismo con sus respectivos valores en sus articulaciones. En la Figura 4.50 se despliega el proceso empleado en la simulación en nuestro sistema.

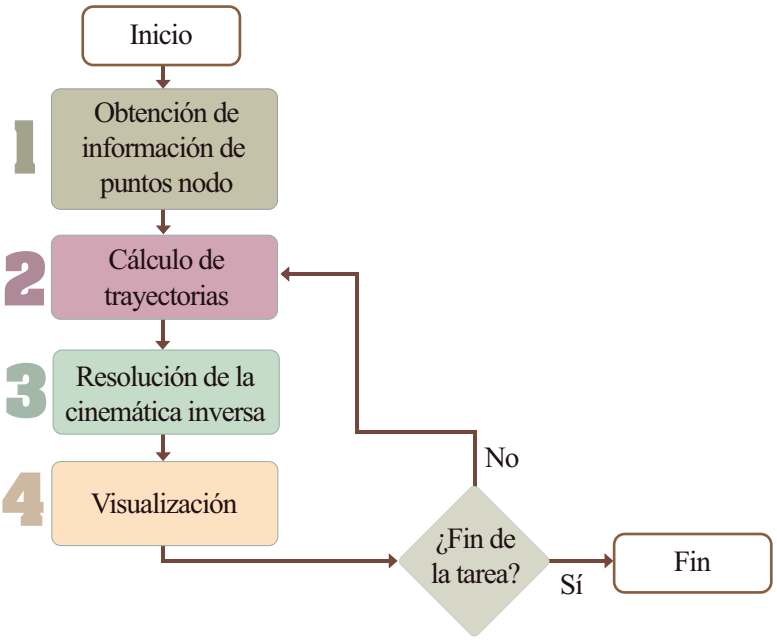


Figura 4.50. Proceso de la simulación.

4.3.2 Obtención de información de puntos nodo

El principal rasgo a considerar en la simulación radica en los recorridos que el robot hace en la escena virtual. Tales recorridos conllevan trayectorias específicas que el robot debe seguir con el fin de efectuar tareas para las que fue programado. Las trayectorias están conformadas por una colección de puntos por los que debe pasar la antorcha de soldadura virtual, de tal modo que exista un trayecto desde un punto origen a un punto final. En este sentido, los puntos que rigen a las trayectorias son los puntos nodo generados por la interfaz háptica y que se han almacenado en un archivo de texto (Sección 3.4.3). Resulta pertinente decir que estos puntos conforman la trayectoria global a seguir por parte de la antorcha de soldadura del robot virtual durante la simulación.

La Tabla 5 describe los parámetros que contiene el citado archivo de texto y corresponden a los n puntos nodo, que a su vez representan las n poses de la antorcha de soldadura capturadas en n instantes. Cada una de estas poses contiene tres valores de posición ($Px, Py\ y\ Pz$) y tres valores más de orientación ($Rx, Ry\ y\ Rz$). Asimismo,

dichas poses dan la pauta a la ruta que tiene que hacer el robot virtual y a su vez guían los movimientos que éste debe efectuar durante las tareas programadas. Evidentemente se requiere leer este archivo y de este modo obtener la información de estas poses, que para diferenciarlas se utiliza un índice, de tal forma que la primer pose se le denomina P_1, la segunda P_2 y así sucesivamente hasta la n pose, misma que se le nombra P_n.

De forma complementaria a las poses de la antorcha extraídas del archivo de texto, se considera a la pose del órgano terminal cuando el robot se encuentra en posición de *home* (Figura 4.51) y se agrega tanto al inicio como al final de la trayectoria global, con las denominaciones P_0 y P_{n+1} respectivamente. La Tabla 10 muestra los valores de los ángulos de cada articulación del robot virtual en posición de *home*. Si estos valores se usan como entrada en el modelo cinemático directo del robot se obtiene la matriz ${}^{0}_{6}T$, misma que se requiere para el cálculo de la pose de la punta de la antorcha con respecto a Σ_G cuando el robot esté en su posición inicial:

$$ {}^{G}_{H}T = {}^{G}_{P}T \ {}^{P}_{0}T \ {}^{0}_{6}T \ {}^{6}_{H}T \tag{4.10} $$

Donde ${}^{G}_{P}T$ es la matriz de las placas con respecto al marco global (Ecuación 3.1); ${}^{P}_{0}T$ es la matriz de emplazamiento (Ecuación 4.8); ${}^{0}_{6}T$ es la matriz del eslabón 6 con respecto al eslabón 0 (Ecuación 4.3) y ${}^{6}_{H}T$ es la matriz de la punta de la antorcha con respecto al eslabón 6 (Ecuación 4.7).

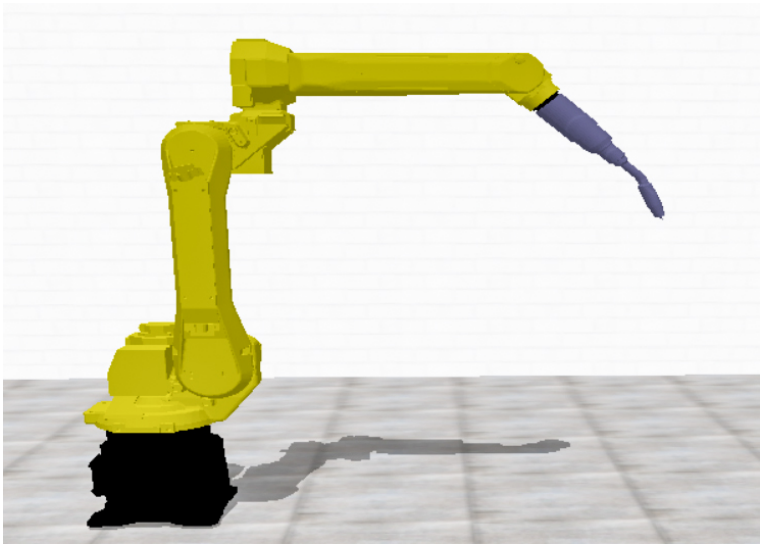


Figura 4.51. Posición de *home* del robot virtual.

Tabla 10. Valores angulares del robot virtual en posición de *home*.

Ángulo	Valor(°)
θ_1	0
θ_2	90
θ_3	0
θ_4	0
θ_5	-45
θ_6	0

4.3.3 Cálculo de trayectorias

Con el cálculo de trayectorias se busca generar una secuencia de valores en el tiempo dentro de las restricciones dadas, que determine la posición y orientación del órgano terminal del robot virtual. Como ya quedó establecido, cada uno de los puntos nodo simboliza una configuración de la antorcha de soldadura con tres para la posición (Px, Py, Pz) y otros tres valores para la orientación (Rx, Ry, Rz), por lo que se tienen seis variables a controlar mientras la antorcha de soldadura cumple el trayecto para el que fue programado. Para llevar a cabo este control se recurre a

funciones de interpolación lineal (Ecuaciones 4.11), tratando que el movimiento de la antorcha debe trasladarse de una configuración origen a una configuración destino de forma gradual. A partir de esta consideración fundamental, es necesario que dicha planificación produzca trayectorias suaves, dicho de otra forma, trayectorias con un alto grado de continuidad [14, 90].

$$f(i_k) = \frac{t}{T} \tag{4.11a}$$

$$f(i_c) = \left(\frac{t}{T}\right) - \left(\frac{1}{2\pi}\right)\left(sen\left(\frac{2\pi t}{T}\right)\right) \tag{4.11b}$$

Donde $f(i_k)$ es la función de interpolación constante y $f(i_c)$ la función de interpolación cicloidal. Además, t es el tiempo transcurrido y T el tiempo de recorrido global.

Dentro de la programación de la simulación, se han definido trayectorias individuales entre los distintos puntos nodo, a las cuales se les ha llamado segmentos de la tarea; de tal forma que la tarea inicia con el segmento $\overline{P_0\ P_1}$, el cual implica un recorrido de la antorcha desde P_0 a P_1 y concluye con el segmento $\overline{P_n\ P_{n+1}}$, que representa el trayecto efectuado por el órgano terminal del robot de P_n a P_{n+1}. A cada uno de estos segmentos se les ha establecido un tiempo de recorrido individual. La sumatoria de estos tiempos individuales nos da como resultado el tiempo de recorrido global T. A partir de esta consideración fundamental, se plantean la Ecuaciones 4.13 y 4.12 para producir trayectorias continuas, entendiendo que éstas se conforman de puntos de paso, mismos que poseen los seis valores que determinan la configuración de la antorcha en un instante.

Dado que los segmentos de la tarea van de un punto nodo a otro, para fines prácticos, denominaremos a j como el índice que controla el número de punto nodo activo y toma valores desde 0 a n, donde n es el número de puntos nodo involucrados en la tarea planificada. En este sentido, la simulación considera sucesivamente el segmento $\overline{P_j\ P_{j+1}}$. Para esto, los valores x_p, y_p y z_p conforman la posición de la punta de la

antorcha durante cada segmento de la tarea y se calculan como sigue:

$$x_p = Px_j + (f(i)\ \Delta_x) \tag{4.12a}$$

$$y_p = Py_j + (f(i)\ \Delta_y) \tag{4.12b}$$

$$z_p = Pz_j + (f(i)\ \Delta_z) \tag{4.12c}$$

Donde Px_j, Py_j y Pz_j corresponden a los términos Px, Py y Pz de P_j respectivamente. Asimismo, $f(i)$ es la función de interpolación lineal; mientras que Δ_x, Δ_y y Δ_z resultan consecuentemente de la operación $P_{j+1} - P_j$ en términos de Px, Py y Pz.

Al mismo tiempo, los valores λ_p, μ_p y ν_p que corresponden a los ángulos de Bryant, constituyen la orientación de la antorcha de soldadura en el recorrido de cada segmento, cuya resolución se logra de la siguiente forma:

$$\lambda_p = Rx_j + (f(i)\ \Delta_\lambda) \tag{4.13a}$$

$$\mu_p = Ry_j + (f(i)\ \Delta_\mu) \tag{4.13b}$$

$$\nu_p = Rz_j + (f(i)\ \Delta_\nu) \tag{4.13c}$$

Donde Rx_j, Ry_j y Rz_j se refieren a los valores Rx, Ry y Rz de P_j respectivamente. Además, $f(i)$ es la función de interpolación lineal; así como Δ_λ, Δ_μ y Δ_ν son el resultado de la operación $P_{j+1} - P_j$ en términos de Rx, Ry y Rz.

4.3.4 Resolución de la cinemática inversa

Una vez establecido lo anterior, es importante decir que aunada a las matrices definidas hasta este momento, se tiene a la matriz de transformación que contiene la información de la pose de la antorcha con respecto a Σ_P. Tal matriz es llamada ${}^{P}_{H}T$. Abundando sobre esta matriz, es preciso señalar que la composición rotacional de la misma es la matriz de rotación de los ángulos de Bryant, alimentándola con los parámetros calculados en las Ecuaciones 4.13, mientras que el vector de posición está

compuesto por los valores de las Ecuaciones 4.12, dando como resultado la siguiente matriz:

$$ {}^{P}_{H}T = \left[\begin{array}{c|c|c|c} c\mu_p c\nu_p & -c\mu_p s\nu_p & s\mu_p & x_p \\ s\lambda_p s\mu_p c\nu_p + c\lambda_p s\nu_p & -s\lambda_p s\mu_p s\nu_p + c\lambda_p c\nu_p & -s\alpha c\mu_p & y_p \\ -c\lambda_p s\mu_p c\nu_p + s\lambda_p s\nu_p & c\lambda_p s\mu_p s\nu_p + s\lambda_p c\nu_p & c\lambda_p c\mu_p & z_p \\ 0 & 0 & 0 & 1 \end{array}\right] \tag{4.14} $$

Donde $c\lambda_p$ significa $cos\lambda_p$; $s\lambda_p$ representa a $sen\lambda_p$ y así sucesivamente.

Para poder resolver el problema de la cinemática inversa del robot virtual mediante el método de Paul, se requiere calcular la matriz *snap*, la cual es la matriz ${}^{0}_{6}T$, misma que se obtiene de la siguiente forma:

$$ {}^{0}_{6}T = {}^{0}_{P}T \ {}^{P}_{H}T \ {}^{H}_{6}T \tag{4.15} $$

Donde ${}^{0}_{P}T$ es la matriz inversa de la matriz de emplazamiento (Ecuación 4.8); ${}^{P}_{H}T$ es la matriz de la antorcha con respecto a Σ_P (Ecuación 4.14) y ${}^{H}_{6}T$ es la matriz inversa de ${}^{6}_{H}T$ (Ecuación 4.7).

Haciendo la asignación de valores de la matriz ${}^{0}_{6}T$ acorde a los términos de la matriz *snap* tenemos:

$$ {}^{0}_{6}T = \begin{bmatrix} s_x & n_x & a_x & p_x \\ s_y & n_y & a_y & p_y \\ s_z & n_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.16} $$

Es con el establecimiento de esta matriz que se está en posibilidades de resolver la cinemática inversa aplicando el modelo explicado en la Sección 4.1.4, donde es de nuestro interés obtener los valores angulares de cada articulación del robot, para esto se echa mano de las Ecuaciones 4.6 para obtener $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$ y θ_6. Estos valores se utilizan como entrada para calcular las matrices de los eslabones con respecto a

Σ_G, considerando las matrices elementales del robot (Ecuaciones 4.1):

$$
\begin{aligned}
{}^{G}_{0}T &= {}^{G}_{P}T \; {}^{P}_{0}T && (4.17a)\\
{}^{G}_{1}T &= {}^{G}_{0}T \; {}^{0}_{1}T && (4.17b)\\
{}^{G}_{2}T &= {}^{G}_{1}T \; {}^{1}_{2}T && (4.17c)\\
{}^{G}_{3}T &= {}^{G}_{2}T \; {}^{2}_{3}T && (4.17d)\\
{}^{G}_{4}T &= {}^{G}_{3}T \; {}^{3}_{4}T && (4.17e)\\
{}^{G}_{5}T &= {}^{G}_{4}T \; {}^{4}_{5}T && (4.17f)\\
{}^{G}_{6}T &= {}^{G}_{5}T \; {}^{5}_{6}T && (4.17g)
\end{aligned}
$$

Donde ${}^{G}_{P}T$ es la matriz de las placas con respecto a Σ_G; ${}^{P}_{0}T$ es la matriz de emplazamiento; ${}^{G}_{i}T$ es la matriz del eslabón i con respecto a Σ_G para $i = 1, 2, 3, 4, 5, 6$, mientras que ${}^{0}_{1}T$, ${}^{1}_{2}T$, ${}^{2}_{3}T$, ${}^{3}_{4}T$, ${}^{4}_{5}T$ y ${}^{5}_{6}T$ son las matrices elementales del robot.

4.3.5 Visualización

Es a partir de las ecuaciones 4.17 que es posible dibujar a las piezas del robot virtual durante la simulación. Para ejecutar su renderizado con OpenGL®, se debió efectuar el proceso de exportación de las piezas al ambiente virtual aplicando la metodología expuesta en la Sección 3.3.2. Al igual que para la visualización de la antorcha virtual maniobrada con la interfaz háptica, se requiere calcular la traslación y rotación de las piezas del robot en la escena en cada iteración del ciclo infinito para la actualización de los gráficos en pantalla. Para conseguir lo anterior, se utilizan dichas ecuaciones, que representan la posición de cada uno de los eslabones con respecto al marco de referencia global.

Dado que el orden de rotaciones sucesivas de las piezas virtuales atiende a la rotación de los ángulos de Bryant, la posición y orientación de las mismas se calculan como en la Sección 3.4.3 con la resolución de la pose de la antorcha con respecto a las placas. Para el caso de la antorcha como órgano terminal del robot virtual, ésta se

dibuja con base en la matriz ${}^{G}_{6}\boldsymbol{T}$.

El tiempo de cada segmento de la trayectoria global se ha establecido en dos segundos. Para que se cumpla la condición de finalizar la tarea, el robot virtual debe llevar a cabo el trayecto comprendido entre $\boldsymbol{P}_0$ hasta $\boldsymbol{P}_n$. En la Figura 4.52 se muestran imágenes de la simulación del sistema.

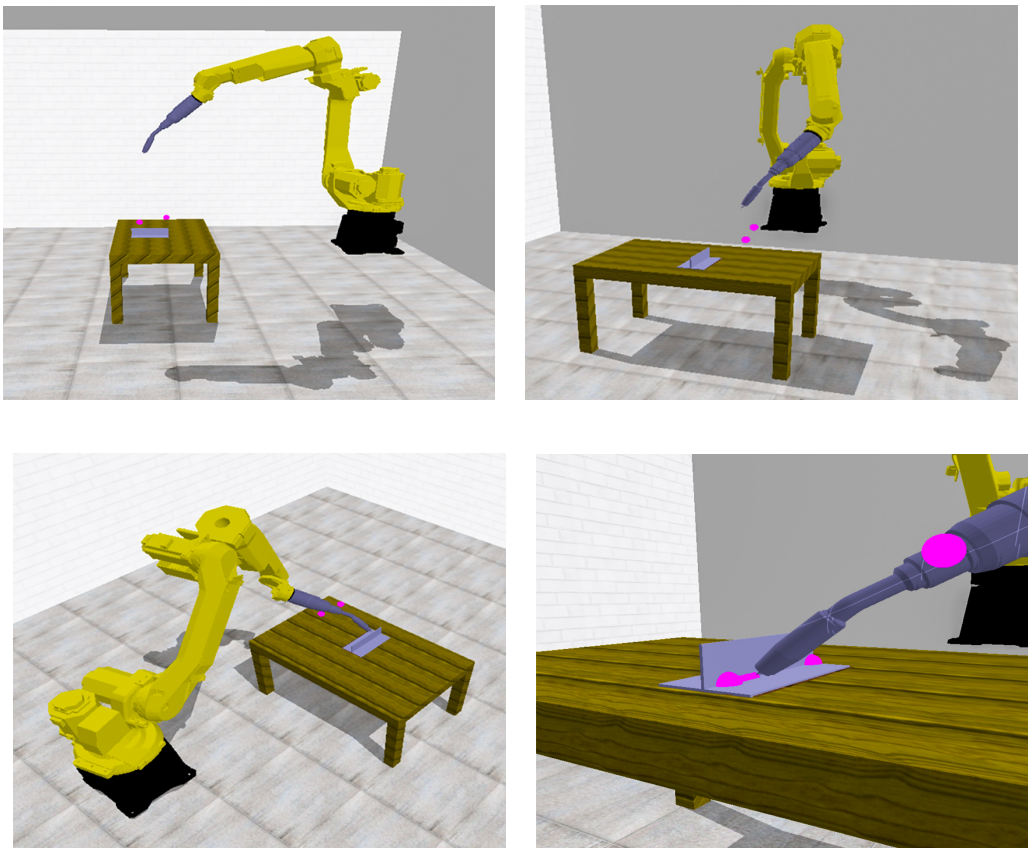


Figura 4.52. Simulación de los movimientos del robot.

Si los resultados de la simulación no son satisfactorios, es necesario regresar a los pasos previos para verificar y modificar la posición ya sea de la mesa de trabajo con respecto al robot o viceversa. Un emplazamiento diferente de los objetos de la escena virtual nos puede dar como resultado una tarea accesible donde anteriormente no lo era.

Capítulo 5

Antorcha de soldadura virtual con comportamiento dinámico

El contenido de este capítulo describe la metodología utilizada para la incorporación del comportamiento dinámico en la antorcha vitual. También se dan detalles sobre la aplicación del modelo resorte-amortiguador en la antorcha para el cálculo de la fuerza a enviar a la interfaz háptica. Por último, se desarrolla un procedimiento experimental para la comparación de la precisión entre los puntos de soldadura colocados con antorchas que poseen diferentes configuraciones.

5.1 Arquitecturas de hardware y software

El hardware utilizado para incorporar un comportamiento dinámico a la antorcha virtual es el mismo que se enlista en el modelado del ambiente virtual (Sección 3.1). Asimismo y de forma análoga a la sección recién mencionada, el modelado geométrico de la antorcha virtual es realizado con SolidWorks™, el cual se convierte a un formato de mallas triangulares a través del software Deep Exploration™. Mientras que el renderizado de la antorcha en el ambiente virtual se consigue utilizando las bibliotecas gráficas de OpenGL®, mismas que serán utilizadas en la plataforma de programación Visual C++. Por otro lado, cuando el usuario maniobre la antorcha virtual dentro del entorno, ésta es controlada por las funciones que proporciona OpenHaptics® y

que brindan la oportunidad de establecer una intercomunicación entre la interfaz háptica y el ambiente virtual.

El software clave del sistema propuesto es AGEIA PhysX©, el cual es un motor de modelado basado en física empleado para el comportamiento dinámico de la antorcha virtual y la detección de colisiones con los objetos del entorno. De tal forma que el kit de desarrollo de software de AGEIA PhysX© gestiona el comportamiento de la AVD (antorcha virtual dinámica) en el ambiente virtual, mientras que el paquete Virtual Hand Toolkit™ (VHT) se encarga del comportamiento de la AVC (antorcha virtual cinemática). Los programas descritos previamente son esquematizados en la Figura 5.1.

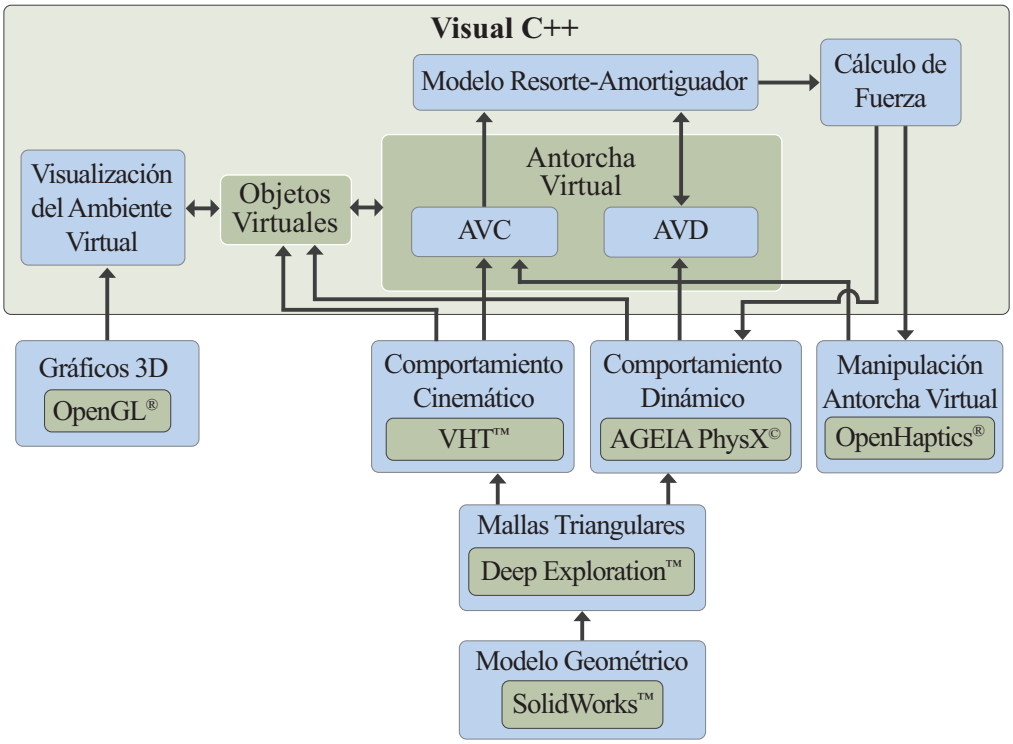


Figura 5.1. Arquitectura de software del ambiente virtual con comportamiento dinámico.

5.2 Modelado de la antorcha de soldadura virtual

Al igual que los objetos del ambiente virtual y las piezas del robot virtual se requiere que el modelo de la antorcha posea las características de una malla triangular para que tal modelo sea exportado al ambiente virtual. El proceso para lograr dicha exportación se explica a detalle en la Sección 3.3.2. En la Figura 5.2 se indican las etapas por las que se transita para incluir el modelo de la antorcha en el ambiente virtual. En primera instancia se observa el modelo de antorcha tridimensional realizado en SolidWorks™ (Figura 5.2a), el cual corresponde al mismo modelo de antorcha detallado en la Sección 3.2. Posteriormente, la antorcha es convertida en un formato geométrico llamado STL (Figura 5.2b). Después se tiene una versión del modelo basado en una malla triangular (Figura 5.2c). Finalmente, se dibuja la antorcha en el ambiente virtual a partir de la información de la malla triangular (Figura 5.2d).

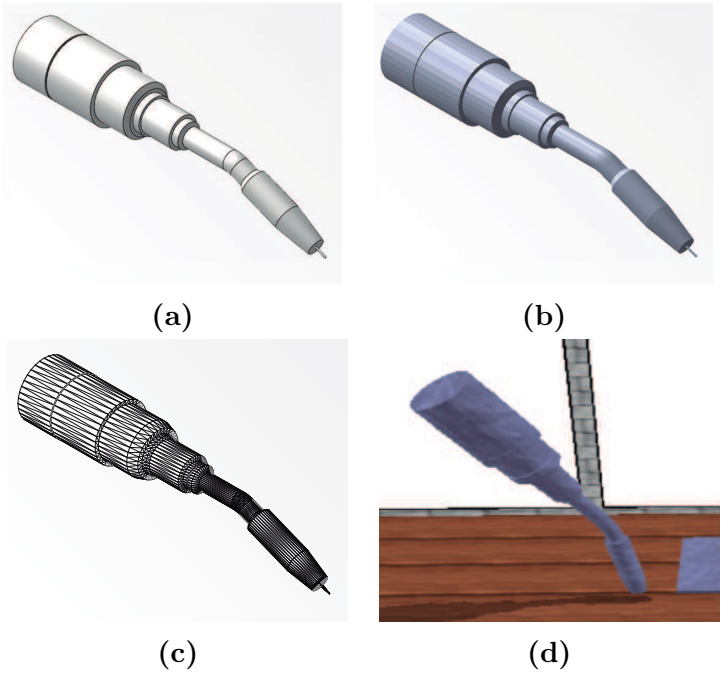


Figura 5.2. Exportación de la antorcha al ambiente virtual.

5.3 Aplicación del modelo resorte-amortiguador en la manipulación de la antorcha virtual

Recapitulando, para llevar a cabo la planificación de las trayectorias de soldadura, el usuario debe manipular la antorcha virtual con la interfaz háptica de acuerdo a lo explicado en la Sección 3.4. Se ha sugerido que el usuario tenga que operar la interfaz moviendo su órgano terminal con su mano para ubicar la antorcha de soldadura en el ambiente virtual. El movimiento de esta herramienta virtual se logra en tiempo real durante la manipulación de la interfaz háptica. De esta forma, las maniobras de la antorcha en el ambiente virtual hacen que el usuario se sienta como en un entorno de programación en línea. Sin embargo, el enfoque de manipulación de la antorcha virtual detallado en dicha sección no garantiza un comportamiento realista de la antorcha durante su operación; esto es debido a que no existe ninguna restricción que impida traspasar los demás objetos presentes en la escena. Ante esta condición, si el usuario eventualmente aplica suficiente fuerza en la interfaz háptica, la antorcha puede penetrar a los objetos virtuales y permitir en algún momento la colocación de puntos nodo "dentro" de las placas a soldar cuando se desee definir el cordón de soldadura.

A continuación, se presenta el modelo resorte-amortiguador soportado por un motor de modelado basado en física, de esta forma se aborda el problema de la interpenetración visual durante la manipulación de la antorcha virtual y se realiza la representación de la fuerza calculada en el marco de la misma, la cual será transmitida a la interfaz háptica.

5.3.1 Integración del motor de modelado basado en física

En los ambientes virtuales, el comportamiento dinámico se consigue mediante la integración de un motor basado en física, en este caso AGEIA PhysX©, el cual es una plataforma para la simulación física en tiempo real, publicado y soportado por

NVIDIA® Corp. Este motor proporciona una biblioteca integrada para la simulación basada en leyes de la física y soporta la simulación de cuerpos rígidos y objetos blandos con un gran rendimiento [91]. Además, con AGEIA PhysX© se simplifica la construcción de los modelos de colisión y la simulación de las interacciones físicas de los objetos durante la planificación de trayectorias de las tareas de soldadura en el entorno virtual. Asimismo, el sistema de PFL se ve favorecido gracias a los enfoques relevantes de detección de colisiones, la biblioteca de computación numérica y los esquemas de simulación dinámica para reducir significativamente la complejidad y el esfuerzo en el desarrollo de software [92].

Este motor de modelado basado en física emplea una escena para los gráficos denominada *NxScene* para gestionar los objetos en un entorno virtual. Puede haber varias instancias de escena a la vez, pero la simulación se realiza por escena. Por otro lado, un objeto en una escena se llama actor, que representa una entidad que interactúa con otros en el entorno virtual. Cada objeto es un *NxActor* y se compone de una unión de formas y un modelo para las colisiones. Una forma es llamada *NxShape* y puede ser una de las primitivas predefinidas de AGEIA PhysX© tal como cápsula, plano, esfera, forma convexa o malla triangular. La detección de colisiones se consigue con una estructura de datos llamada *PMap*, mediante una clase denominada *NxPMap* [93].

El enfoque formulado en esta tesis sugiere que la antorcha virtual sea representada en la escena virtual mediante dos modelos tridimensionales: una antorcha virtual cinemática (AVC) y una antorcha virtual dinámica (AVD), donde ambos modelos han de interconectarse mediante sistemas resorte-amortiguador (SRA). Este enfoque está basado en el que fue aplicado por Borst e Indugula en un sistema para mover una mano virtual en un ambiente virtual [46]. Con la AVD se pretende que el proceso de soldadura se asemeje lo más posible a la realidad, ya que por más fuerza que se aplique al dispositivo háptico, dará la sensación que las placas de soldadura virtuales no se penetran durante la programación de tareas de soldadura.

Asimismo, se plantea que durante la manipulación de la interfaz háptica, la AVC y la

AVD posean la misma pose, buscando que exista un acoplamiento virtual. Con el fin de lograr dicho acoplamiento se utiliza el modelo resorte-amortiguador, mismo que también permite proporcionar al usuario la sensación de contacto cuando la antorcha colisione con otros objetos en la escena, ya que permite calcular la fuerza que se envía a la interfaz háptica[36].

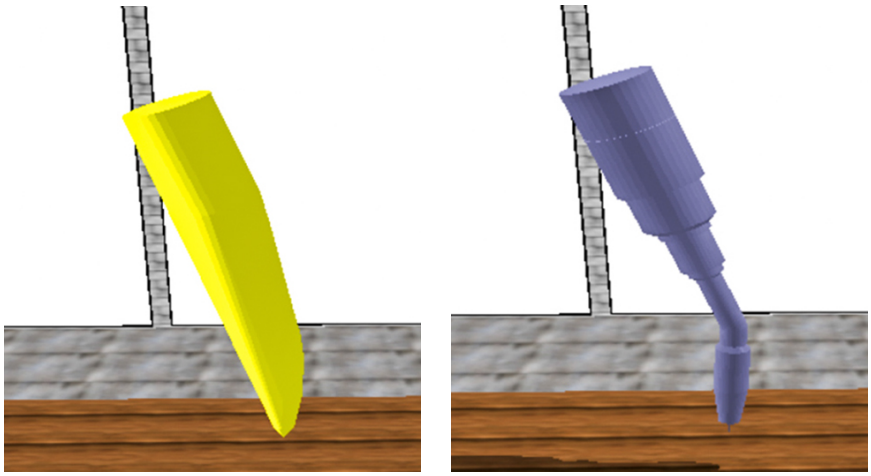
Para poder incorporar el comportamiento dinámico referido a la antorcha virtual primero hay que considerar que la AVC es controlada por las funciones que proporciona OpenHaptics® dentro de Visual C++ y que brindan la oportunidad de establecer una intercomunicación entre la interfaz háptica y el ambiente virtual. La Ecuación 3.8 se utiliza para calcular la pose de la AVC dentro de la escena virtual al momento que el usuario mueve la interfaz háptica, facilitando en gran medida la programación del sistema, mientras que las funciones de VHT son usadas para renderizar la AVC como una forma convexa. En el proceso de manipulación de la antorcha en el ambiente virtual, la AVC está "fuera de pantalla", por lo que se le considera como la antorcha no-visible (Figura 5.3a).

Una función de OpenHaptics® declara un ciclo infinito que es usado para refrescar los gráficos en la pantalla, dentro del cual la escena es redibujada (Figura 3.21). Asimismo, en cada iteración del referido ciclo, la pose de la AVC es calculada basada en la matriz ${}^{P}_{A}\boldsymbol{T}$, de tal forma que los movimientos que el usuario hace con la interfaz háptica son reflejados directamente en la AVC.

Cuando el usuario mueve la AVC, la AVD tiende a seguirla imitando el comportamiento de una antorcha real durante la planificación de trayectorias de soldadura. El paquete AGEIA PhysX© suministra la detección de colisiones y gracias a esto se evita que la AVD penetre en otros objetos en el ambiente virtual. Durante este proceso, el usuario solamente ve la AVD, la cual es también llamada antorcha visible y es una forma no-convexa (Figura 5.3b).

Durante la manipulación de la antorcha virtual, la escena está conformada solamente por formas no-convexas de AGEIA PhysX© renderizadas con las funciones de

OpenGL®. Por su parte, todos los objetos están renderizados "fuera de pantalla" como formas convexas de VHT. OpenGL® utiliza el archivo de la malla triangular para dibujar, en la escena virtual, las formas geométricas convexas y no-convexas. Debido al modelado físico de la AVD realizado en AGEIA PhysX©, el sistema puede detectar cualquier colisión entre formas no-convexas mediante algoritmos propios, de esta forma se evita el traspaso de la AVD y se provoca en el usuario una sensación de más realismo cuando éste interactúa con la interfaz háptica.



(a) AVC **(b)** AVD

Figura 5.3. Antorchas renderizadas

Se ha planteado que la AVD y la AVC se encuentren unidas mediante SRA, donde un SRA está formado por un resorte-amortiguador y una masa de referencia en la AVD con su referencia correspondiente en la AVC. El resorte-amortiguador puede ser un resorte-amortiguador lineal (RAL) o un resorte-amortiguador torsional (RAT). Dado lo anterior, en [94], se propusieron tres antorchas con diferentes configuraciones de SRA, donde obtuvo mejor desempeño la antorcha que utiliza cuatro SRA lineales (SRAL): uno en su punta ($\boldsymbol{M}_1$), otro está ubicado cerca de su base ($\boldsymbol{M}_2$) y los otros dos se colocaron a sus lados ($\boldsymbol{M}_3$ y $\boldsymbol{M}_4$). En la Figura 5.4 se ilustra esta configuración y aunque en ésta no hay SRA torsionales (SRAT), las fuerzas combinadas de los cuatro RAL permiten el movimiento de rotación de la AVD.

Se utilizó una estrategia en la que, una vez que se calcula la fuerza de contacto, el

componente resultante se aplicará a la masa de referencia de la AVD correspondiente. De esta manera, la AVD es "empujada" hacia la AVC, lo que hace que ambas tengan la misma postura eventualmente [36]. En la antorcha con la configuración de le Figura 5.4, las masas M_1, M_2, M_3 y M_4 se adhieren a la AVD como masas de referencia. Particularmente, en esta configuración, las masas no están físicamente unidas a la AVD, pero mantienen una posición fija con con respecto al marco de la antorcha durante su movimiento.

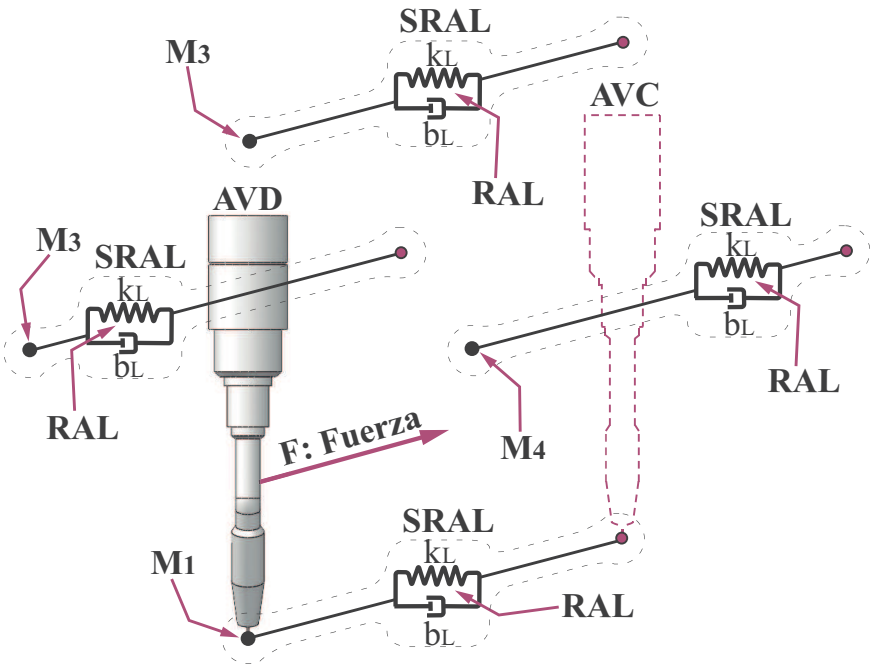


Figura 5.4. Configuración de SRA.

5.3.2 Acoplamiento virtual

Con el fin de obtener un comportamiento realista durante el contacto entre la antorcha virtual y los otros objetos en el entorno, el modelo resorte-amortiguador se ha adoptado basándose en el uso de un acoplamiento virtual entre la AVC y la AVD. Esta técnica fue propuesta originalmente por Colgate et al. [95]. Este enfoque propone interconectar la AVC y la AVD con SRA. Además, el cálculo de la fuerza de contacto se realiza utilizando la diferencia de posiciones de ambas antorchas (Figura 5.5), como en el modelo de cálculo desarrollado por Borst e Indugula [80].

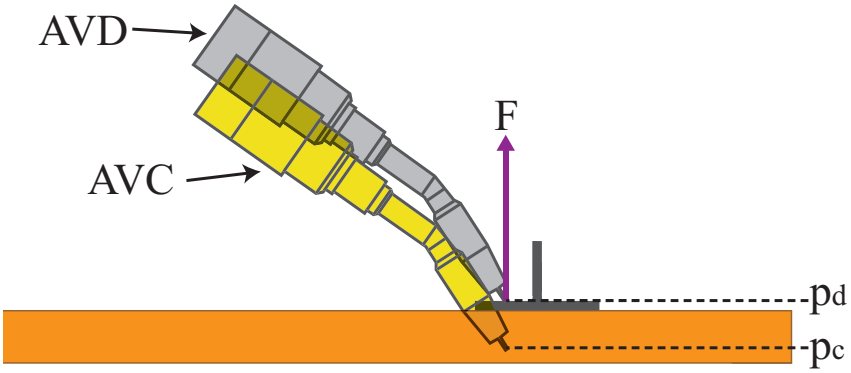
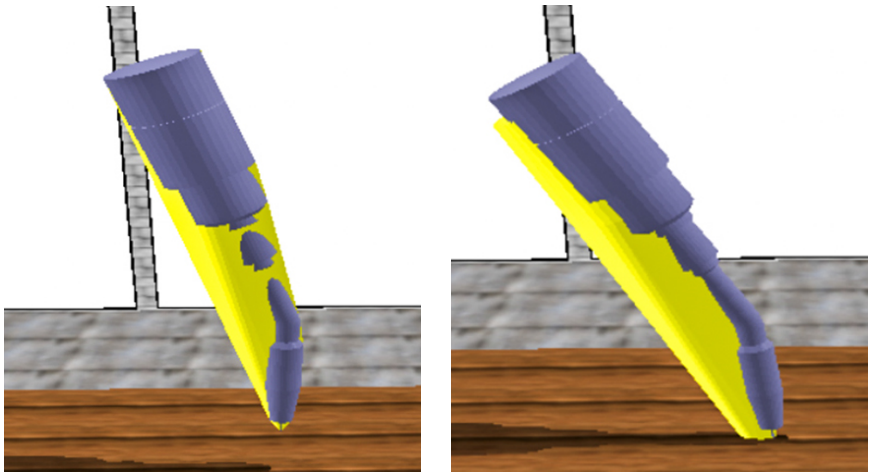


Figura 5.5. Diferencia de posiciones de las antorchas sobre el eje y.

Antes de que tenga lugar la planificación de las trayectorias de soldadura, tanto la AVC como la AVD tienen la misma pose. Lo mismo ocurre cuando la AVC no presenta desplazamientos, es decir, cuando la interfaz háptica no presenta movimiento alguno por parte del usuario. En esta situación, ambas antorchas parecen ser solo un modelo. Ahora bien, al accionar la interfaz háptica, la AVC se mueve y a su vez la AVD reproduce la misma pose de la AVC sin interpenetar visualmente a los otros objetos en la escena.

En el caso de la AVC, su representación como una forma convexa tiene como objetivo hacer que se diferencie visualmente de la AVD. El programa tiene una función para ver simultáneamente ambas antorchas. Es así que, tanto la AVC como la AVD pueden ser identificados por el usuario cuando tienen la misma pose (Figura 5.6a). Mientras no haya contacto de la AVD con algún objeto, la superposición de ambas antorchas permanece. Sin embargo, cuando se produce una colisión, la AVC penetra en el objeto y la AVD se mantiene "pegada" a éste (Figura 5.6b).



(a) Superposición de la AVC y la AVD

(b) Colisión de la AVD con la mesa.

Figura 5.6. Acoplamiento virtual

5.3.3 Resorte-amortiguador lineal

Cada resorte-amortiguador lineal produce una fuerza de translación denominada componente lineal F.

$$F = k_L(p_c - p_d) - b_L(v_d - v_c) \tag{5.1}$$

Donde:

F	es la fuerza aplicada a la AVD con el fin de seguir a la AVC
k_L	es la constante de resorte lineal. ejecutando un movimiento lineal.
b_L	es la constante de amortiguador lineal.
p_d, p_c	son las posiciones de la masa de referencia en la AVD y su correspondiente referencia en la AVD respectivamente (con respecto a Σ_G).
v_d, v_c	son las velocidades de la masa de referencia en la AVD y su correspondiente referencia en la AVD respectivamente.

También:

$$v_d = \frac{p_d^{\,t} - p_d^{\,t-1}}{\Delta t} \quad \text{y} \quad v_c = \frac{p_c^{\,t} - p_c^{\,t-1}}{\Delta t} \tag{5.2}$$

Donde:

$p_d^{\,t}, p_c^{\,t}$	son las posiciones actuales de la masa de referencia en la AVD y su correspondiente referencia en la AVD respectivamente (con respecto a Σ_G).
$p_d^{\,t-1}, p_c^{\,t-1}$	son las posiciones previas de masa de referencia en la AVD y su correspondiente referencia en la AVD respectivamente (con respecto a Σ_G).
Δt	es el tiempo transcurrido entre la posición en t y $(t-1)$.

5.3.4 Cálculo de la fuerza de contacto

Para calcular la fuerza que se va a transmitir al usuario a través de la interfaz háptica, durante la manipulación de la antorcha virtual, se requiere la diferencia de posiciones de la punta de la AVC y la punta de la AVD. Lo anterior se expresa:

$$F_H = \alpha(p_c - p_d) \tag{5.3}$$

Donde:

α	es el grado de la fuerza, $1 \leq \alpha \leq 3$.
p_c, p_d	son las posiciones de la AVC y la AVD respectivamente.

El valor de F_H es un vector de coordenadas Cartesianas, mismo que representa la dimensión de la fuerza que el usuario siente en su mano vía la interfaz háptica. El valor de la fuerza enviado a ésta durante el contacto de la AVD y cualquier objeto en el ambiente es representada en el sistema por un vector saliendo de la punta de

la antorcha (Figura 5.7). Si la AVD no tiene contacto con los otros objetos, la AVD y la AVC están totalmente superpuestas. En este caso no hay fuerza de contacto:

$$(p_k - p_d) = 0 \therefore F_H = 0 \tag{5.4}$$

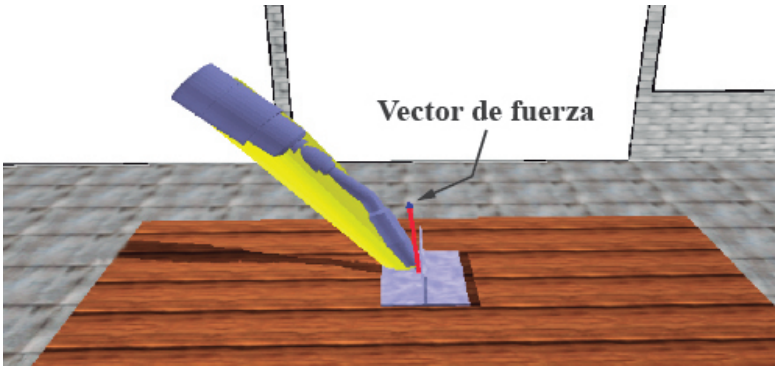


Figura 5.7. Fuerza indicada durante la manipulación de la antorcha.

5.4 Desarrollo experimental de acoplamiento virtual

El acoplamiento virtual tiene como finalidad que durante la manipulación de la antorcha virtual mediante la interfaz háptica exista una distancia mínima entre la AVC y la AVD, así como una diferencia menor entre los valores angulares de las mismas. Para encontrar el mejor acoplamiento posible, se consideraron tres parámetros en la configuración de SRA:

- Constantes de resorte-amortiguador.
- Posiciones de las masas.
- Valores de las masas.

Además, se debe observar un movimiento estable y uniforme de la AVD y la sensación de fuerza debe ser suave. El estudio experimental tiene como objetivo evaluar los

valores de posición y orientación de las antorchas (AVD y AVC) en el acoplamiento virtual dentro de la escena virtual.

5.4.1 Caso de estudio

El acoplamiento virtual entre la AVC y la AVD con tres configuraciones de SRA se verificó y validó en dos tipos de experimentos [94]: el primero de ellos, X_1, consistió en ejecutar una trayectoria automática con la antorcha virtual y en el segundo, X_2, el usuario realizó una trayectoria con la antorcha virtual utilizando la interfaz háptica. En ambos experimentos, el comportamiento esperado fue que la AVD siguiera a la AVC lo más cerca posible. Cabe señalar que durante este caso de estudio, Σ_G se localizó en el centro de la mesa.

En el experimento X_1, se programó una trayectoria lineal automática donde la AVC presentaba una velocidad constante. Durante el curso del recorrido, las posiciones de la punta y el centro de masa de ambas antorchas, así como sus valores angulares, se registraron en cada instante de tiempo (con respecto a Σ_G). Esta trayectoria comienza en el punto P_1 y termina en el punto P_2. Las coordenadas Cartesianas de la AVC con respecto a Σ_G en el punto P_1 y en el punto P_2 son Px_1 = -15 cm, Py_1 = 10 cm, Pz_1 = -15 cm y Px_2 = 25 cm, Py_2 = 10 cm, Pz_2 = -15 cm respectivamente. Del mismo modo, los ángulos de Bryant de la AVC en P_1 y P_2 con respecto a Σ_G son $\lambda_1 = 120°$, $\mu_1 = 20°$, ν_1 = -50° y $\lambda_2 = 30°$, $\mu_2 = 70°$, $\nu_2 = 140°$ respectivamente. Una interpolación lineal de estos valores se realizó en un intervalo de 2 segundos.

El experimento X_2 consistió en mover la AVC desde P_1 a P_2, pasando por P_K (Figura 5.8). Los valores de posición de la punta y el centro de masa de la AVC y la AVD, así como los valores de orientación de ambas antorchas se registraron durante el trayecto en cada instante de tiempo (con respecto a Σ_G). El proceso de interpolación se realizó en un lapso de 2 segundos. Diez sujetos, todos ellos estudiantes de licenciatura sin experiencia previa con sistemas de realidad virtual participaron en este experimento, todos ellos varones, diestros y de edades comprendidas entre los

18 y los 23 años. El mismo grupo de sujetos participó en las tres condiciones experimentales de la antorcha para llevar a cabo la manipulación de la antorcha a través de la interfaz háptica (Figura 5.9).

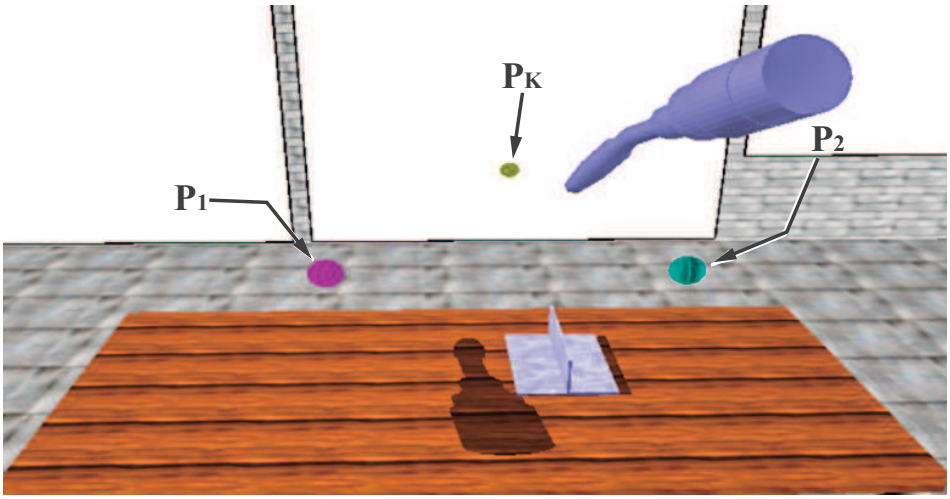


Figura 5.8. Escena durante experimentos de acoplamiento virtual.

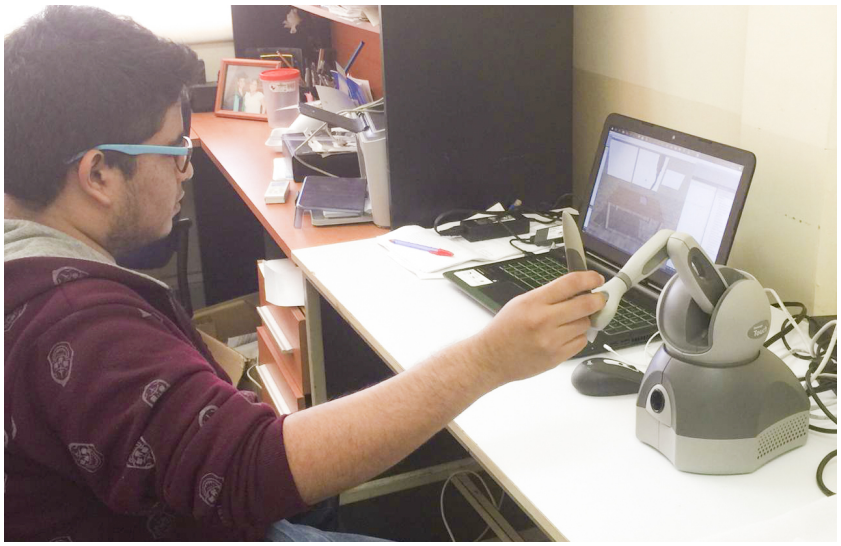


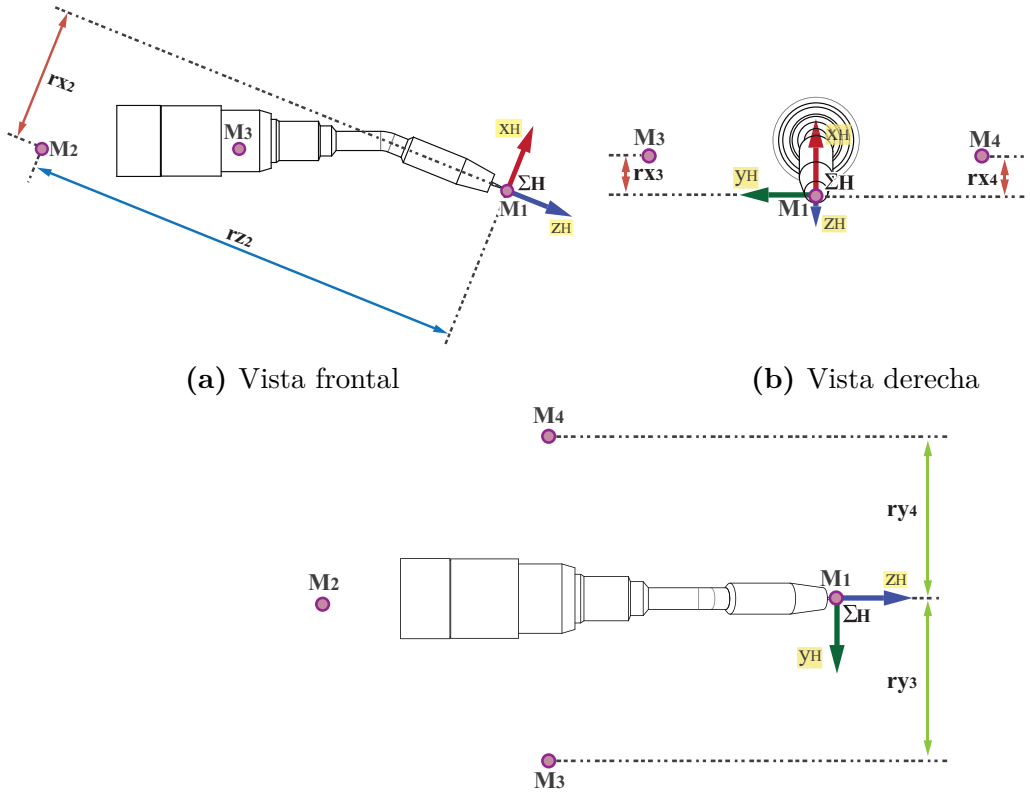
Figura 5.9. Sujeto durante un experimento de acoplamiento virtual.

5.4.2 Resultados

De manera general, de las tres configuraciones de SRA presentadas en [94] la configuración de SRA que presentó mejor acoplamiento virtual fue la que utilizó

cuatro SRAL (Figura 5.4), donde los valores de las constantes de cada RAL (k_L, b_L) se obtuvieron al realizar pruebas de manipulación de la antorcha virtual, en las cuales se fueron ajustando con el objetivo de lograr el mejor acoplamiento virtual posible.

Como se emplearon cuatro SRAL se obtuvieron cuatro conjuntos de constantes lineales en ese sentido (Tabla 11). Por otro lado, salvo la posición de M_1, las demás posiciones de las masas unidas a la antorcha en dicha configuración de SRA también se modificaron eventualmente, donde se probaron experimentalmente diferentes valores hasta obtener un acoplamiento virtual consistente en las pruebas realizadas. La distribución de masas obtenida se muestra en la Figura 5.10 y sus posiciones se enumeran en la Tabla 12.



(a) Vista frontal **(b)** Vista derecha

(c) Vista superior

Figura 5.10. Masas en la AVD.

Otra variable considerada fue el valor de cada una de las masas unidas a la AVD para

conseguir un acoplamiento virtual estable durante las pruebas de manipulación. De acuerdo a estas pruebas, en la Tabla 13 se muestran los valores obtenidos al respecto. Por su parte, para el cálculo de la fuerza a enviar a la interfaz háptica, el coeficiente α se determinó empíricamente realizando pruebas de manipulación ajustando el valor de α hasta que se obtuvo una sensación de fuerza constante, resultando $\alpha = 0.75$.

Tabla 11. Constantes de resorte-amortiguador.
$Unidades \rightarrow k_L : kg/seg^2 \quad b_L : kg/seg$

Masa	k_L	b_L
M_1	263,863	72,769
M_2	51,423	14,059
M_3	473	128
M_4	473	128

Tabla 12. Posiciones de las masas en la AVD con respecto a Σ_H (en cm).

Masa	rx_i	ry_i	rz_i
M_1	0.0	0.0	0.0
M_2	-13.87	0.0	-47.35
M_3	-6.39	39.19	-28.90
M_4	-6.39	-39.19	-28.90

Tabla 13. Valores de las masas en la AVD (en kg).

Masa	Valor
M_1	5,059.31
M_2	1,338.95
M_3	9.14
M_4	9.14

Para evaluar el acoplamiento virtual de las tres configuraciones de SRA se consideraron en primer término, parámetros en función de la posición de la AVC y la AVD tales como:

- Máxima diferencia en las distancias de las puntas (MDDP)
- Máxima diferencia en las distancias de los centros de masa (MDDCM)
- Promedio de la diferencia en las distancias de las puntas (PDDP)
- Promedio de la diferencia en las distancias de los centros de masa (PDDCM)

También se tomaron en cuenta parámetros en función de la orientación de ambas antorchas:

- Máxima diferencia angular (MDA)
- Diferencia angular promedio (DAP)

Los datos recopilados en el experimento X_1, así como los promedios de los datos registrados en las diez pruebas para el experimento X_2 son presentados en las Tablas 14 y 15.

Tabla 14. Diferencias de distancias entre la AVC y la AVD (en *cm.*)

Experimento	MDDP	MDDCM	PDDP	PDDCM
X_1	1.09	3.16	0.18	0.48
X_2	0.45	1.20	0.14	0.34

Tabla 15. Diferencias angulares entre la AVC y la AVD (en *grados*).

Experimento	MDA	DAP
X_1	3.53	0.71
X_2	2.01	0.50

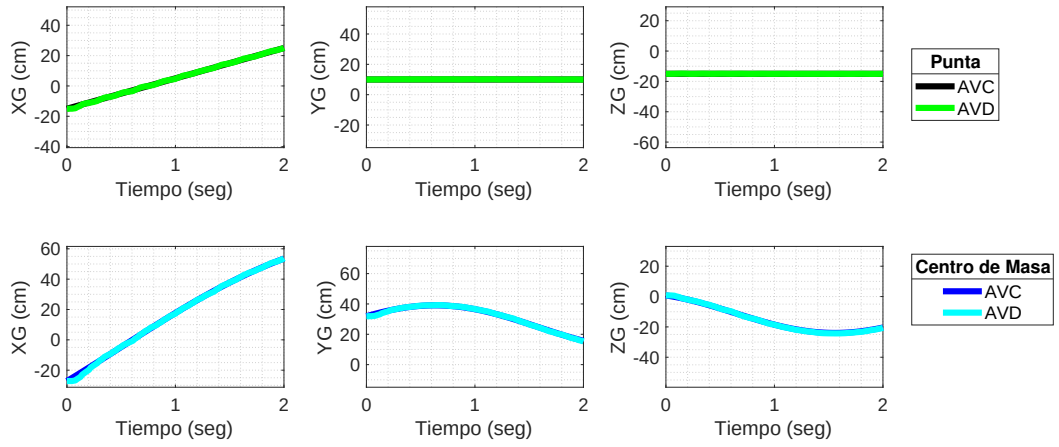
También, en la Tabla 16 se enumeran los datos obtenidos en el experimento X_2, donde se ha calculado un coeficiente de ponderación global (CPG) mediante la suma de todos los parámetros pertenecientes a cada sujeto. El valor mínimo del PCG se muestra en negrita y su valor máximo de ponderación se ha subrayado.

Tabla 16. Datos obtenidos en experimento X_2.

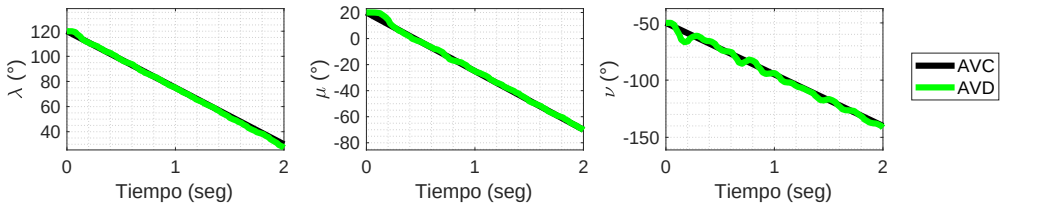
	Sujeto									
	1	2	3	4	5	6	7	8	9	10
MDDP (*cm*)	0.39	0.51	0.54	0.58	0.40	0.51	0.49	0.39	0.41	0.26
MDDCM (*cm*)	1.18	1.03	1.06	1.37	0.94	2.63	1.32	0.94	0.80	0.66
PDDP (*cm*)	0.13	0.15	0.14	0.16	0.14	0.12	0.15	0.17	0.13	0.08
PDDCM (*cm*)	0.39	0.33	0.34	0.41	0.32	0.37	0.38	0.35	0.27	0.24
MDA (*grados*)	4.29	2.04	1.35	1.62	1.12	3.47	1.97	2.09	1.00	1.12
DAP (*grados*)	0.87	0.51	0.51	0.50	0.36	0.63	0.49	0.51	0.32	0.32
CPG	7.25	4.57	3.94	4.64	3.28	7.73	4.80	4.45	2.93	**2.68**

Para ilustrar el acoplamiento virtual que manifiesta la configuración de SRA propuesta durante los experimentos, se generaron gráficos en ese sentido. De tal forma que en la Figura 5.11a se muestran las posiciones de las puntas de la AVC y la AVD durante la trayectoria automática que se programó para el experimento X_1. Del mismo experimento se obtuvieron las posiciones de los centros de masa de ambas antorchas, mismas que son presentadas en la citada figura. Por otro lado, en la Figura 5.11b se exponen gráficamente los ángulos de Bryant de la AVC y la AVD registrados mientras se ejecutaba el experimento X_1.

Debido a que el experimento X_2 dio como resultado diez conjuntos de datos, de forma ilustrativa únicamente se presentan las gráficas de los mejores y peores resultados. En este sentido y tomando en cuenta el valor de CPG de la Tabla 16, el peor resultado de acoplamiento virtual en el experimento lo presentó el sujeto 6. La información gráfica tanto de posición como de orientación al respecto se muestra en la Figura 5.12. Finalmente, quien manifestó el mejor acoplamiento virtual durante la prueba experimental fue el sujeto 10. En la Figura 5.13 se ilustran dichos resultados.

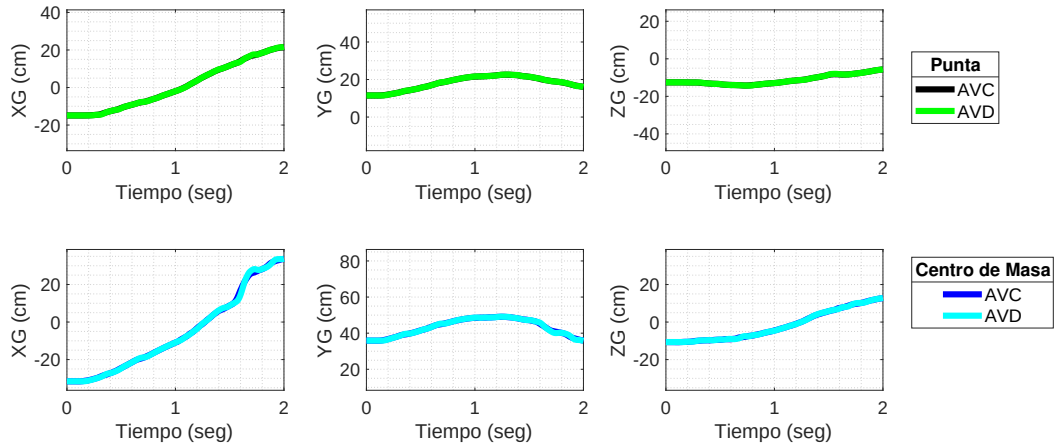


(a) Datos de posición

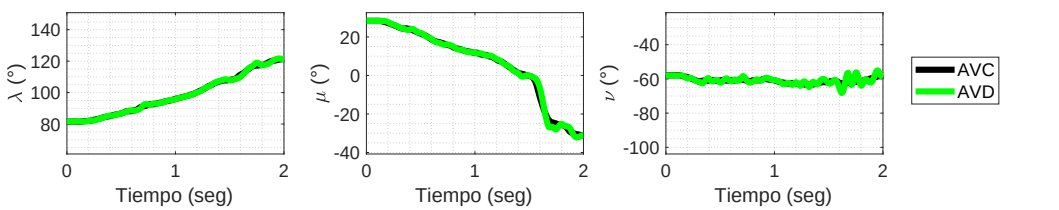


(b) Datos de orientación

Figura 5.11. Resultados en experimento X_1.

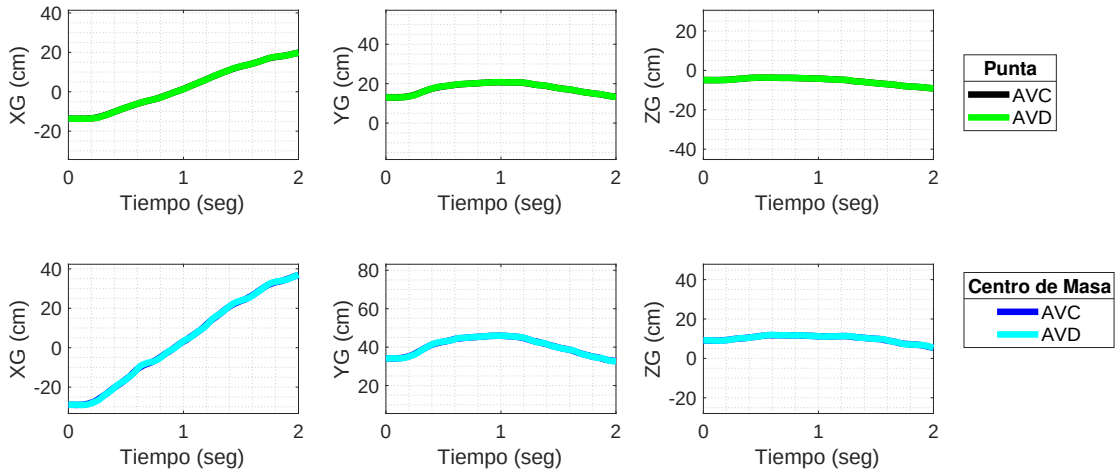


(a) Datos de posición

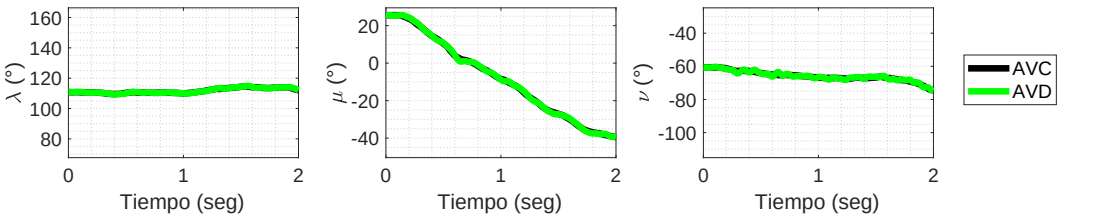


(b) Datos de orientación

Figura 5.12. Peor resultado en experimento X_2 (Sujeto 6).



(a) Datos de posición



(b) Datos de orientación

Figura 5.13. Mejor resultado en experimento X_2 (Sujeto 10).

5.5 Desarrollo experimental de precisión

En la planificación de trayectorias en el sistema para PFL, la precisión con la que se definen los puntos nodo dentro del ambiente virtual es esencial, puesto que mientras más precisos sean los valores posicionales y angulares de la antorcha de soldadura registrados en esta fase, menos compleja se torna la programación del robot de soldadura. Lo anterior se debe a que si los datos generados por el usuario con la interfaz háptica poseen un alto nivel de precisión, el trabajo en las fases subsecuentes del proceso de la PFL es menor para que la ejecución de la tarea en el robot real sea la deseada.

La precisión referida implica que cuando el usuario desee registrar los puntos de soldadura sobre las placas en T, éstos sean establecidos sobre las placas en T y no

dentro de ellas ni flotando. La fase experimental abordada en esta sección pretende analizar la precisión en ese sentido, para lo cual fueron tomadas en cuenta dos variables que afectaron a la manipulación de la antorcha virtual:

- Retorno de fuerzas.
- Comportamiento dinámico.

También, se llevó a cabo una evaluación subjetiva para evaluar la experiencia del usuario al probar con diferentes condiciones en la maniobra de la antorcha virtual.

5.5.1 Caso de estudio

La precisión de los puntos de soldadura registrados en el ambiente virtual mediante la interfaz háptica se revisaron y estimaron en cuatro experimentos con condiciones distintas cada uno de ellos. La Tabla 17 describe tales condiciones.

Tabla 17. Condiciones en experimentos de precisión.

Condición	Características
X	Sin retorno de fuerzas y sin comportamiento dinámico
F	Con retorno de fuerzas y sin comportamiento dinámico
C	Sin retorno de fuerzas y con comportamiento dinámico
$F + C$	Con retorno de fuerzas y con comportamiento dinámico

En la condición X no existe retorno de fuerzas en la interfaz háptica, esto es, que si la antorcha virtual colisiona con algún objeto en el ambiente virtual, no se transmite ninguna fuerza a dicho dispositivo. Asimismo, los objetos virtuales del sistema no presentan comportamiento dinámico, por lo que no existe ninguna restricción que impida el traslape de dichos objetos.

Cuando la antorcha virtual impacta contra algún objeto de la escena y el usuario siente tal choque en la interfaz háptica, pero si dicho usuario aplica una fuerza

considerable en ésta, la antorcha puede traspasar la mesa o las placas virtuales, se tienen las características de la condición $\boldsymbol{F}$. En caso contrario tenemos a la condición $\boldsymbol{C}$, el cual presenta como primer rasgo que el usuario no siente ningún tipo de fuerza en la interfaz háptica en el momento de colisión de la antorcha con otros objetos virtuales. Sin embargo, la antorcha no penetra ni las placas ni la mesa en el ambiente virtual en ningún momento.

Para finalizar esta serie de descripciones, se tiene a la condición $\boldsymbol{F + C}$. Aquí, el sistema detecta cualquier choque de la antorcha virtual. Luego, el usuario percibe el impacto en su mano vía la interfaz háptica. Aunado a esto, el modelado dinámico de los objetos virtuales ofrece un comportamiento más realista al impedir que los mismos se traspasen entre sí.

En los casos de las condiciones que comprenden el retorno de fuerzas ($\boldsymbol{F}$ y $\boldsymbol{F + C}$), el cálculo de la fuerza de contacto se efectúa de acuerdo al procedimiento detallado en la Sección 3.4.2 donde se utiliza la Ley de Hooke para tal fin. Ahora bien, cuando la condición presenta el comportamiento dinámico ($\boldsymbol{C}$ y $\boldsymbol{F + C}$), la fuerza de contacto se calcula basándose en la metodología expuesta en la Sección 5.3.4, en la cual se recurre al modelo resorte-amortiguador para efectuar dicho cálculo.

Sin embargo, pese a que en el procedimiento experimental de la Sección 5.4.2 se logró una configuración de SRA para una manipulación estable de la antorcha virtual, los parámetros obtenidos en tal experimento no funcionan adecuadamente cuando se requiere hacer movimientos más finos en distancias pequeñas. En ese sentido se hicieron algunos ajustes en las constantes de resorte-amortiguador y en los valores de las masas adheridos a la AVD para conseguir movimientos consistentes de la antorcha virtual al manipularse con la interfaz háptica durante los experimentos de precisión. En las Tablas 18 y 19 se presentan estos datos. Por su parte, las posiciones de las masas en la AVD que se requirieron en este estudio experimental fueron las de la Tabla 12.

Tabla 18. Constantes de resorte-amortiguador.
Unidades $\rightarrow k_L : kg/seg^2 \quad b_L : kg/seg$

Masa	k_L	b_L
M_1	99,963	41,769
M_2	40,823	20,259
M_3	629	194
M_4	629	194

Tabla 19. Valores de las masas en la AVD (en kg).

Masa	Valor
M_1	5,080.31
M_2	2,938.95
M_3	9.14
M_4	9.14

El procedimiento experimental planteado consideró las condiciones de la Tabla 17 para determinar cuál de ellas ofrece una mayor precisión en la colocación de puntos de soldadura virtuales mediante la interfaz háptica. Para la evaluación de estas condiciones se dispusieron tres puntos referenciales sobre las placas de soldadura en el ambiente virtual. El primer punto, P_1, se dibujó en la placa horizontal; el segundo de ellos, P_2, se situó en la placa vertical y el tercero, P_3, se ubicó en la intersección de ambas placas (Figura 5.14). Las coordenadas Cartesianas de P_1, P_2 y P_3 con respecto al marco de las placas (Σ_P) son $Px_1 = 6$ cm, $Py_1 = 2$ cm, $Pz_1 = 0$ cm; $Px_2 = 6$ cm, $Py_2 = 0$ cm, $Pz_2 = 2$ cm y $Px_3 = 6$ cm, $Py_3 = 0$ cm, $Pz_3 = 0$ cm respectivamente.

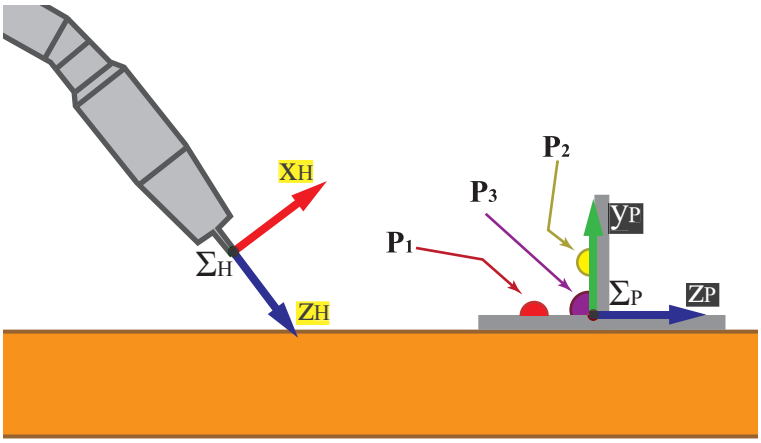


Figura 5.14. Puntos de referencia en experimento de precisión.

Los sujetos participantes en el experimento debían manipular la antorcha virtual accionando la interfaz háptica; y recurriendo tanto a su perspectiva como a su convencimiento, posicionar la punta de la antorcha primeramente en P_1 y registrar la posición de Σ_H. Después se hizo lo mismo considerando a P_2, para finalmente hacer lo propio con P_3. Las posiciones en los tres casos se registraron con respecto a Σ_P. Para esta etapa experimental, se recurrió a trece sujetos varones, estudiantes de posgrado sin experiencia previa con sistemas de realidad virtual, diestros y con edades entre los 23 y 30 años. Todos ellos participaron en las cuatro condiciones experimentales que controlaban el comportamiento de la antorcha virtual durante la colocación de puntos de soldadura.

Figura 5.15. Sujeto durante el experimento de precisión.

5.5.2 Resultados

Los resultados aquí presentados están basados en primer término en la posición de la punta de la antorcha al registrar la punta de la antorcha en los tres puntos detallados en la sección anterior. También, son consideradas cada una de las cuatro condiciones que controlan la manipulación de la antorcha durante la experimentación. Dando un total de doce gráficas con dicha información.

Para fines prácticos, se añade también una tabla que resume la precisión lograda sobre cada punto de referencia en el experimento, es decir, el contenido de cada tabla incluye los porcentajes de puntos de soldadura que los participantes colocaron:

- En la placa. En los casos de P_1 y P_2 se refieren a los puntos de soldadura que fueron registrados en la placa horizontal y placa vertical respectivamente. En cambio, en P_3, los que se registraron en la junta de ambas placas. Para determinar si un punto de soldadura se registró en la placa, se consideró una tolerancia de ± 0.05 cm con respecto a la posición de los puntos de referencia del experimento.

- Flotantes. Poseen esta categoría los puntos cuya posición en y_P sea mayor a 0.05 cm cuando se trate de aquéllos colocados en la placa horizontal (P_1). Para el caso de los registrados en la placa vertical (P_2), se consideran los puntos con valores menor a -0.05 cm en su posición en z_P. Si las posiciones de los puntos en y_P son mayores que 0.05 cm o en z_P son menores que -0.05 cm cuando se contemplan los puntos de la junta de las placas (P_3), también adquieren esta categoría. Lo anterior considerando las coordenadas Cartesianas de P_1, P_2 y P_3 con respecto al marco Σ_P y la tolerancia fijada para determinar los puntos en las placas. Se les denomina flotantes porque fueron colocados sin tocar las placas y se encuentran "flotando" en el aire.

- Adentro. Son la parte complementaria de las dos anteriores y conforman la parte contraria de los puntos flotantes en cuanto a las condiciones a cumplir. Estos puntos reciben tal calificativo porque fueron dispuestos placas adentro.

5.5.2.1 En la placa horizontal

En este caso únicamente se consideró el eje z del marco Σ_P para evaluar las posiciones de los puntos de soldadura. Dado que $Pz_1 = 0$ cm y tomando en cuenta la tolerancia establecida, los puntos registrados que posean una posición en z dentro del rango de ± 0.05 cm se consideraron colocados en la placa horizontal. De la Figura 5.16 a la 5.19 se visualizan los puntos de soldadura durante el experimento de colocación del punto P_1.

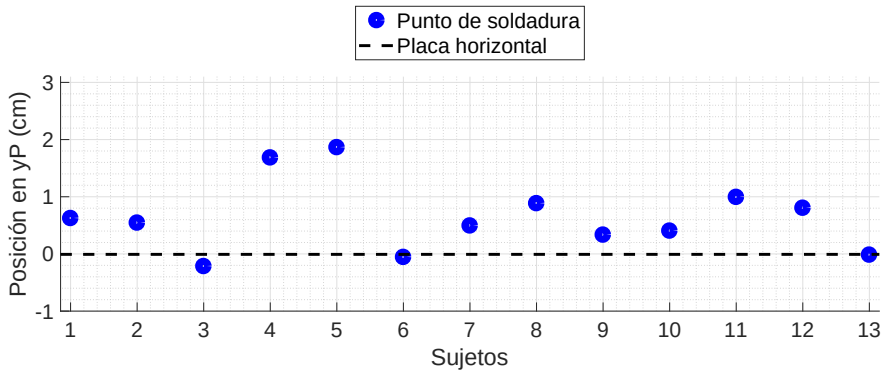


Figura 5.16. Puntos de soldadura en la placa horizontal con condición X.

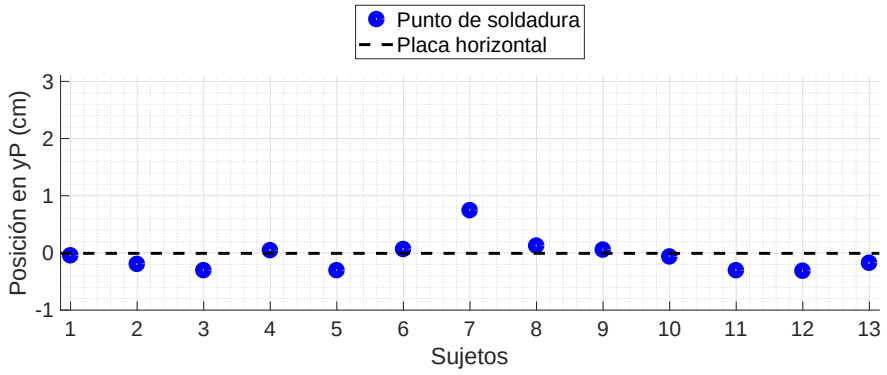


Figura 5.17. Puntos de soldadura en la placa horizontal con condición F.

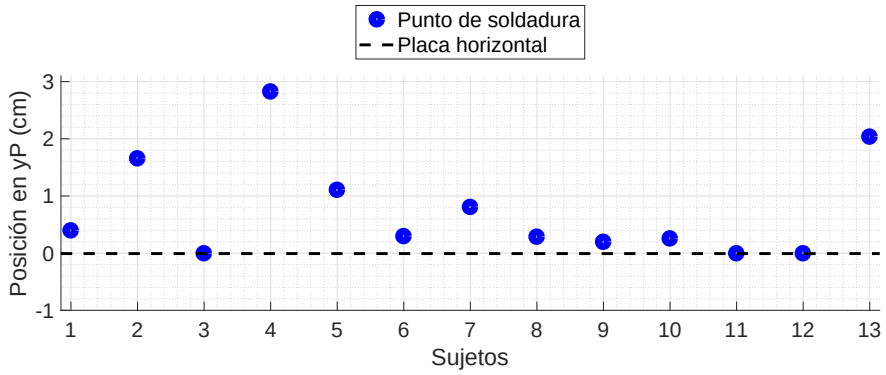


Figura 5.18. Puntos de soldadura en la placa horizontal con condición C.

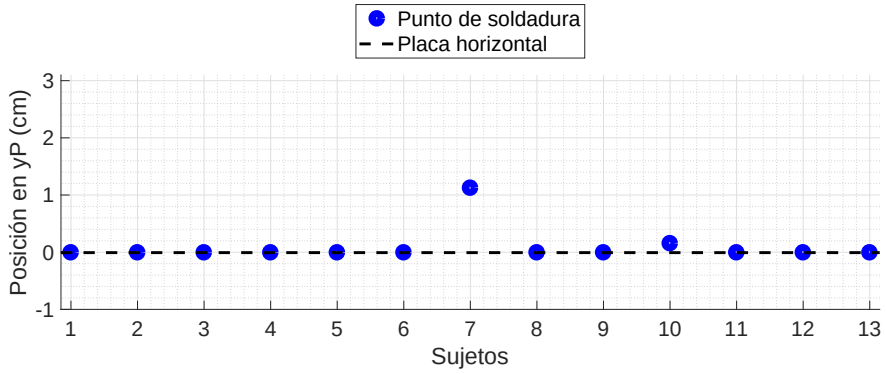


Figura 5.19. Puntos de soldadura en la placa horizontal con condición $F + C$.

Con fines comparativos se introdujo en la Tabla 20 los datos sobre la posición de los puntos registrados en las cuatro condiciones del experimento en turno.

Tabla 20. Concentrado de puntos de soldadura en la placa horizontal.

Condición	En la placa	Flotantes	Adentro
X	15.38%	76.92%	7.69%
F	15.38%	38.46%	46.15%
C	23.08%	76.92%	0.00%
$F + C$	84.62%	15.38%	0.00%

5.5.2.2 En la placa vertical

El criterio para definir los puntos de soldadura que se colocaron en la placa vertical es similar al definido para la placa horizontal, sólo que esta vez fue tomado en cuenta el eje y del marco Σ_P para tal valoración. Puesto que $Py_2 = 0$ cm, aquellos puntos cuya posición en y que oscilaron entre ± 0.05 cm se evaluaron como registrados en la placa vertical. Al respecto, de la Figura 5.20 a la 5.23 indican gráficamente tales puntos.

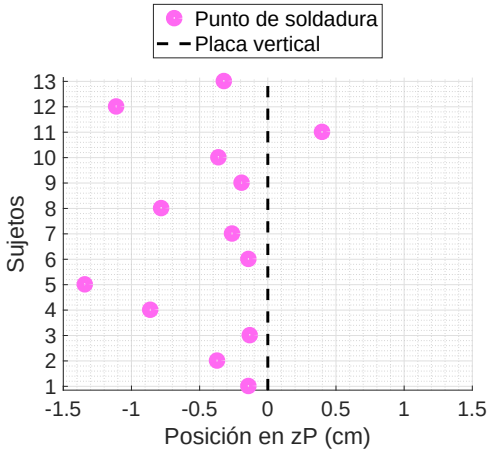


Figura 5.20. Puntos de soldadura en la placa vertical con condición X.

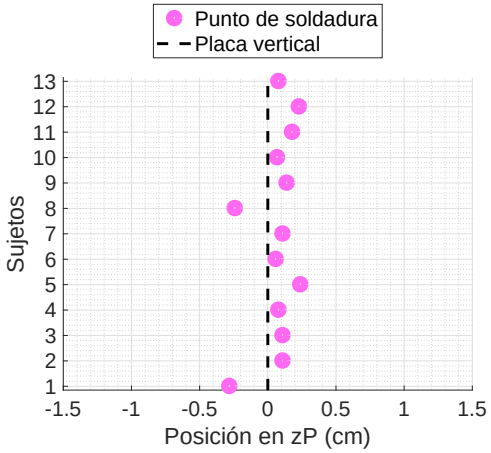


Figura 5.21. Puntos de soldadura en la placa vertical con condición F.

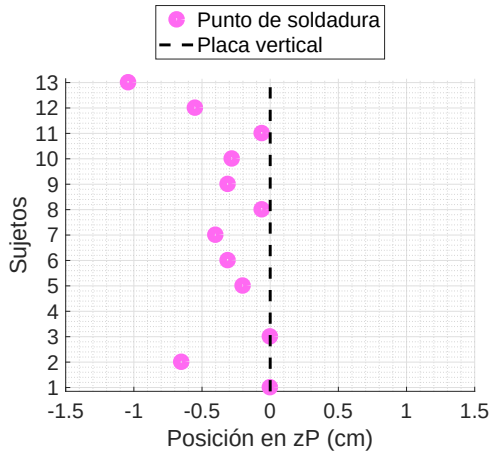


Figura 5.22. Puntos de soldadura en la placa vertical con condición C.

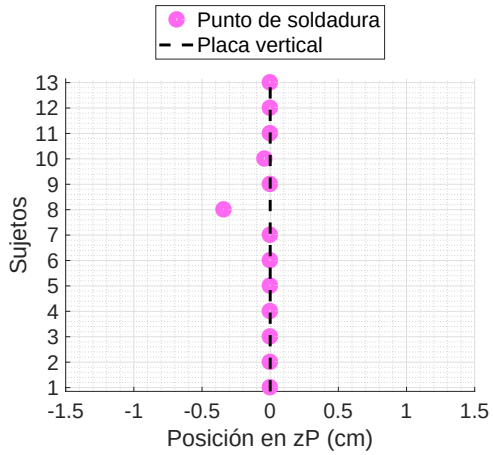


Figura 5.23. Puntos de soldadura en la placa vertical con condición $F + C$.

La Tabla 21 presenta información sobre la posición de los puntos de soldadura en este experimento que dan la pauta para un posterior análisis.

Tabla 21. Concentrado de puntos de soldadura en la placa vertical.

Condición	En la placa	Flotantes	Adentro
X	0.00%	92.31%	7.69%
F	0.00%	15.38%	84.62%
C	15.38%	64.62%	0.00%
$F + C$	92.31%	7.69%	0.00%

5.5.2.3 En la junta de las placas

Si se parte de que $Py_3 = 0$ cm y $Pz_3 = 0$ los puntos de soldadura cuyas posiciones en y y en z estén entre ± 0.05 se tomaron en cuenta como colocados en la junta de las placas horizontal y vertical. Para este caso se usaron los ejes y y z del marco Σ_P. Se ilustra para una mejor comprensión la posición de dichos puntos con respecto a ambas placas, de la Figura 5.24 a la 5.27.

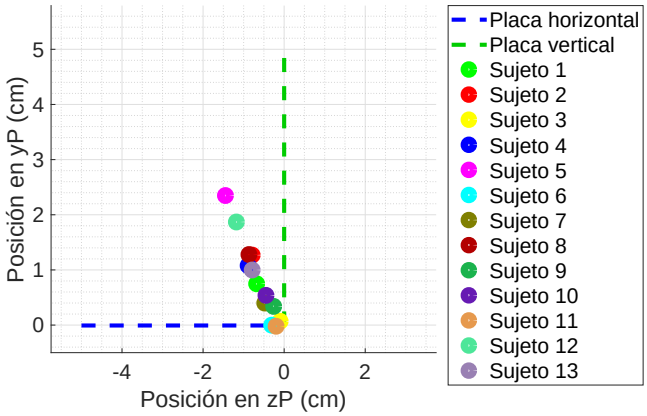


Figura 5.24. Puntos de soldadura en la junta de las placas con condición X.

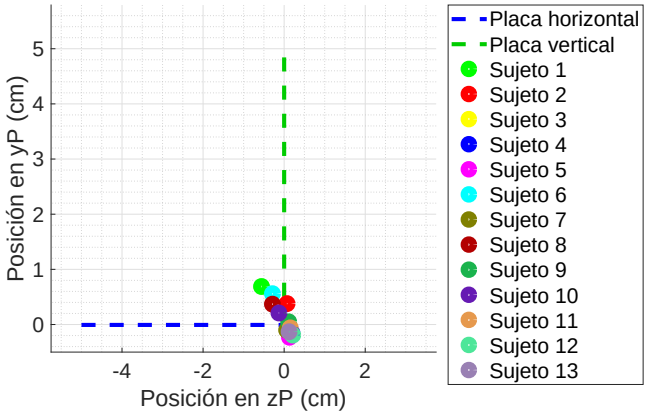


Figura 5.25. Puntos de soldadura en la junta de las placas con condición F.

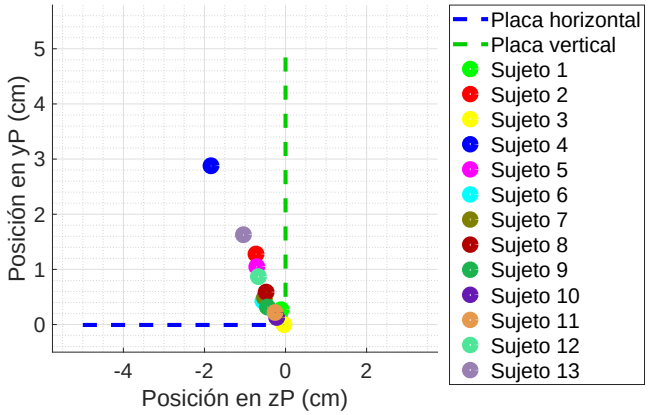


Figura 5.26. Puntos de soldadura en la junta de las placas con condición C.

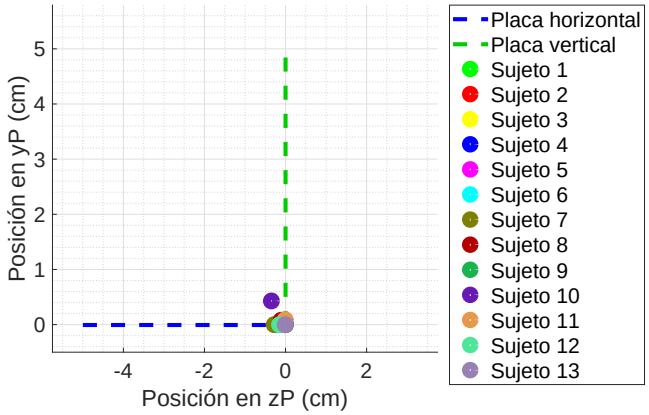


Figura 5.27. Puntos de soldadura en la junta de las placas con condición $F + C$.

El resumen con los datos sobre la posición de los puntos de soldadura con respecto a la junta de las placas se presenta en la Tabla 22.

Tabla 22. Concentrado de puntos de soldadura en la junta de las placas.

Condición	En la junta	Flotantes	Adentro
X	0.00%	84.62%	15.38%
F	0.00%	30.77%	69.23%
C	7.69%	92.31%	0.00%
$F + C$	46.15%	53.85%	0.00%

5.5.3 Evaluación subjetiva

Con el fin de obtener una retroalimentación de los usuarios participantes en los experimentos de precisión, se realizó un procedimiento de evaluación subjetiva basado en la información recopilada de los sujetos después de su participación en los experimentos con las condiciones X, F, C y $F + C$. Cada sujeto respondió a una serie de preguntas relacionadas con su apreciación del realismo de la interacción, la precisión de la ejecución de la tarea y la facilidad de su ejecución durante la manipulación de la antorcha virtual en la colocación de puntos de soldadura en dichos experimentos (Anexo A).

Dado que las variables involucradas en tales experimentos son por un lado el retorno de fuerzas en la interfaz háptica y por el otro el comportamiento dinámico en la escena virtual, se evaluaron por separado las contribuciones de ambas a la percepción del usuario. Las dos primeras preguntas del cuestionario están avocadas a recabar información al respecto. Para responder a estas preguntas, los sujetos expresaron su apreciación con una puntuación de 0 a 5.

Los aspectos a estimar en las dos primeras preguntas son la contribución de dichas variables a la percepción de realismo durante la interacción; la precisión de la ejecución de la tarea y la facilidad con que ésta se ejecutó. Los resultados se muestran en las Figuras 5.28 y 5.29. Asimismo, los promedios de estimaciones de las apreciaciones con ambas variables se indican en la Tabla 23.

También este cuestionario incluye una pregunta en la cual el sujeto tuvo que clasificar las condiciones que controlaron a la antorcha virtual durante los experimentos. Aquí, se tuvo que ordenar de la mejor a la peor en cuanto a la percepción propia y experiencia de cada usuario durante estos experimentos, asignándoles los números del 1 al 4 para tal fin (Figura 5.30).

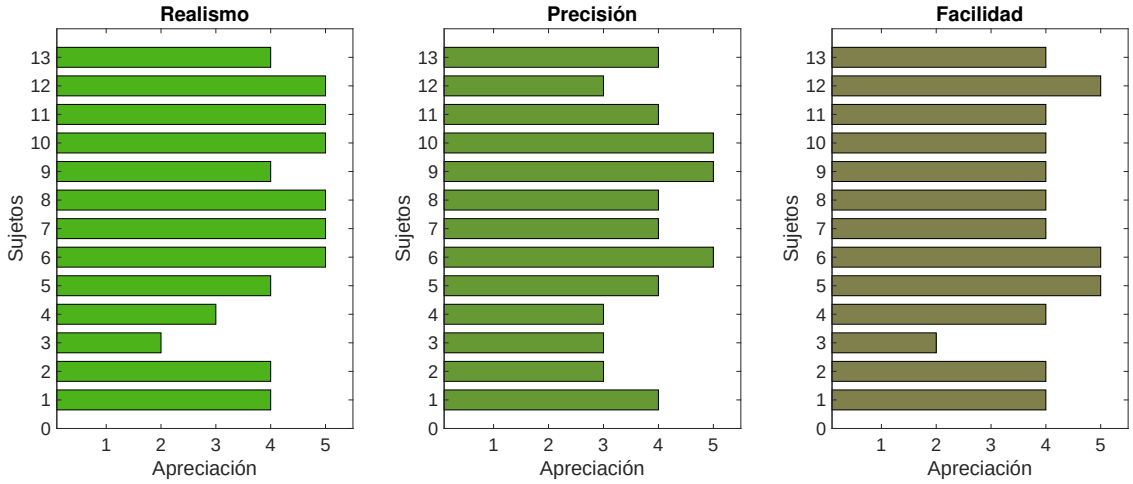


Figura 5.28. Evaluación subjetiva con retorno de fuerzas.

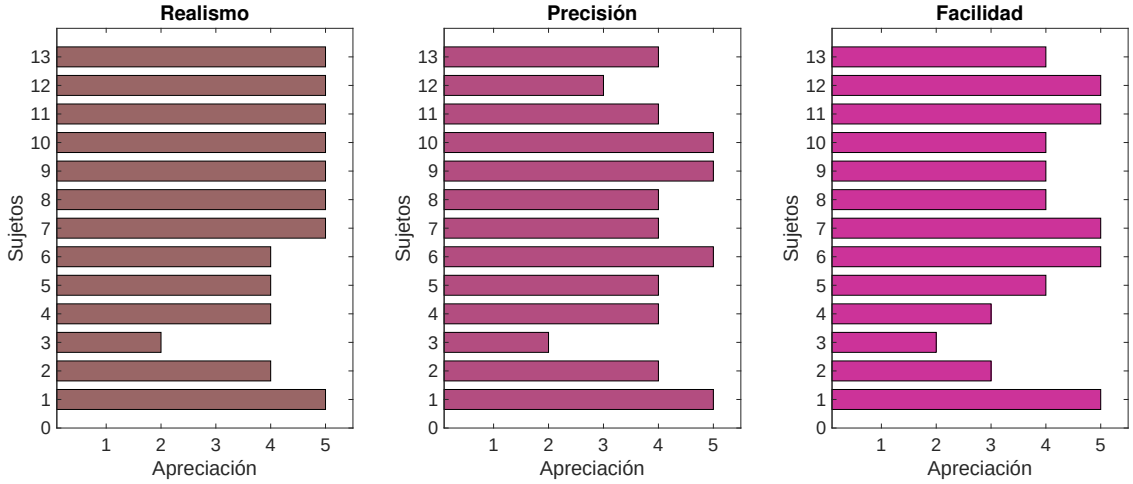


Figura 5.29. Evaluación subjetiva con comportamiento dinámico.

Tabla 23. Promedios de estimaciones en la evaluación subjetiva.

Variable	Realismo	Precisión	Facilidad
Retorno de fuerzas	4.23	3.92	4.08
Comportamiento dinámico	4.46	4.08	4.08

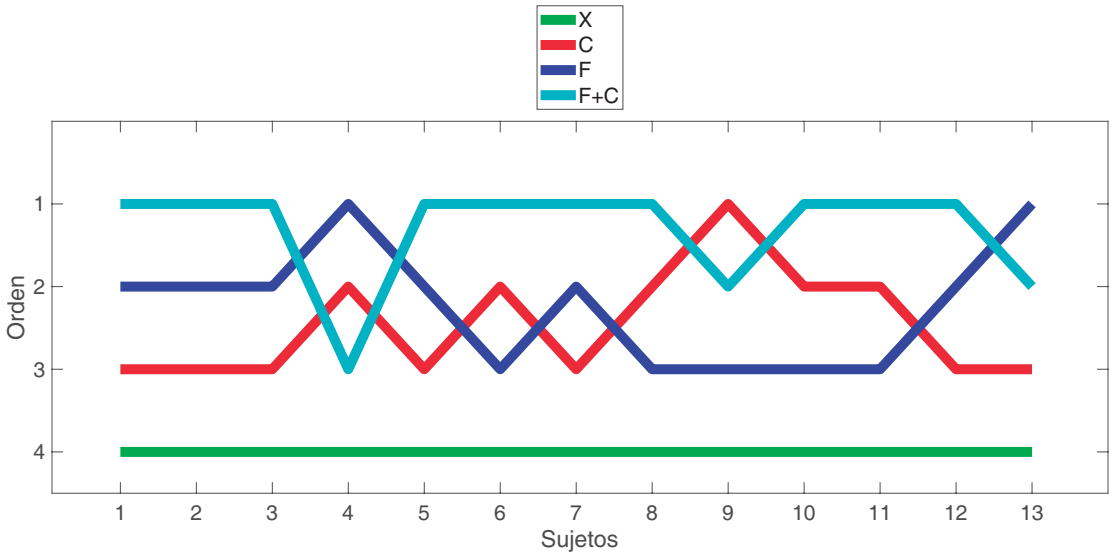


Figura 5.30. Evaluación subjetiva con comportamiento dinámico.

Capítulo 6

Programación del robot de soldadura

Este capítulo hace una exposición de la metodología utilizada para la programación del robot de soldadura a partir del sistema de PFL. Se describe el hardware y el software del robot en cuestión. También se define un procedimiento para traducir las consignas articulares generadas en la simulación de los movimientos del robot virtual en comandos que sean entendidos por el robot real, y una vez efectuado un procedimiento de calibración éste ejecute las tareas programadas.

6.1 Controlador del robot

Considerando la metodología de PFL que respalda el desarrollo de este documento, una vez que se ha efectuado con la interfaz háptica la planificación de los movimientos del robot y la viabilidad de éstos ha sido verificada durante la simulación, se está en posibilidad de validarlos en el robot Fanuc ARC Mate 100iC. El modelo R30i B es el controlador que soporta a este robot (Figura 6.1).

El controlador contiene a la computadora que opera el robot. Alberga también el software de aplicación, la fuente de poder, los circuitos de control, la memoria que dirige la operación del robot y el *teach-pendant*, el cual es conectado al controlador externamente. El controlador también es responsable de la comunicación con

dispositivos externos. El usuario controla el robot usando el *teach-pendant* como el mostrado en la Figura 6.2.

Figura 6.1. Controlador modelo R30i B.
(Cortesía Fanuc Robotics).

El *teach-pendant* es un dispositivo de interacción para el operador que despliega los menús del software del controlador. Está conectado al controlador a través de un cable que se conecta a la placa del procesador principal (CPU) dentro del controlador.

La pantalla del *teach-pendant* es táctil y provee una interfaz gráfica a color mediante la cual el usuario puede ingresar a los diversos menús contextuales, gestionar múltiples ventanas y acceder a las opciones de configuración del software del controlador. Asimismo, sus botones se han diseñado para que el software del controlador sea fácil de usar (Figura 6.2a). En el Anexo B se muestra el conjunto de botones que presenta el modelo de *teach-pendant* del robot Fanuc, mediante los cuales el usuario interactúa tanto con el propio manipulador como con el software del controlador y que se localizan en la parte inferior del dispositivo.

Los principales usos del *teach-pendant* son:

- Mover el robot.
- Crear, editar y probar programas.

- Gestionar el proceso de producción.

- Verificar estatus del robot durante la ejecución de programas.

- Ejecutar funciones manuales.

- Configurar parámetros de las aplicaciones del controlador.

Los interruptores *deadman*, que se encuentran en la parte de atrás del *teach-pendant* (Figura 6.2b), se utilizan uno a la vez para garantizar la seguridad personal cuando éste se encuentra encendido, al interrumpir el movimiento del robot en situaciones de emergencia. El operador debe agarrar y mantener cualquiera de los dos interruptores *deadman* presionado durante la activación del movimiento del robot y durante la ejecución del programa. Si el operador liberara su agarre del interruptor mientras se realiza este proceso, se retira la servoalimentación y se aplican inmediatamente los frenos del robot.

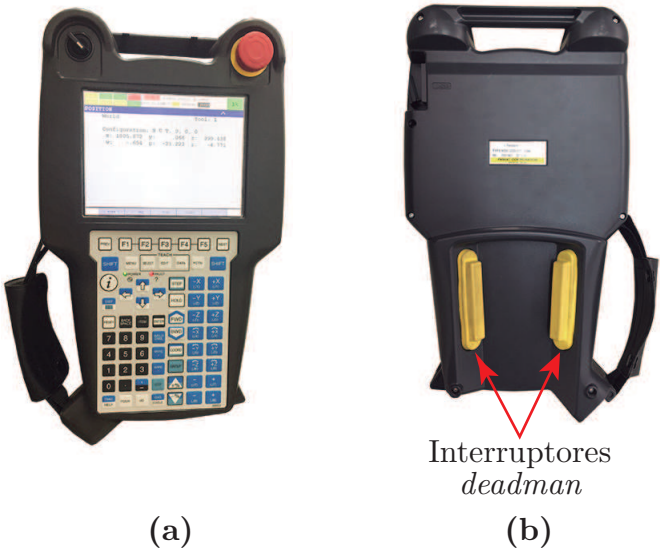


Figura 6.2. *Teach-pendant* del robot Fanuc.
((a) en *vista frontal* y (b) en *vista trasera*)

6.1.1 Software del controlador

El software del controlador consta de dos paquetes: el software de control del robot, presente en todo tipo de robot Fanuc, y el software de aplicación ArcTool™, instalado únicamente en robots de soldadura. El ArcTool™ manipula el equipo de soldar: antorcha, enrollador y máquina de soldar. Ambos están integrados en forma trasparente para definir la información requerida para la programación de aplicaciones; probar programas; ejecutar la producción y monitorear información de los procesos.

Este software de aplicación está diseñado para simplificar y estandarizar la configuración y el funcionamiento de las aplicaciones de soldadura por arco del robot Fanuc. Con ArcTool™, el operador tiene control total sobre el proceso de soldadura. Además, las instrucciones de soldadura de arco son fáciles de entender y la pantalla del *teach-pendant* muestra los menús donde éstas se agrupan. Todas las funciones del robot pueden ser realizadas usando estos menús [96].

Por otro lado, el paquete en cuestión soporta la soldadura para muchos tipos de equipos de soldadura (Lincoln Electric Power Wave i400 en nuestro caso). Las configuraciones de fábrica de equipos de soldadura específicos pueden ser seleccionadas, y hay soldaduras de propósito general que pueden ser aplicados a la mayoría de equipos de soldadura. ArcTool™ es capaz de monitorear varias señales de entrada del equipo de soldadura; si alguna de estas señales indican un problema, la soldadura y la ejecución del programa se detienen y un mensaje de error es desplegado en la pantalla del *teach-pendant*.

6.1.2 Programación con el *teach-pendant*

Un programa es una serie de comandos de robot que le dice a éste cómo moverse y qué hacer para ejecutar una tarea. Mientras los programas son creados, son almacenados automáticamente en la memoria del controlador. El hecho de utilizar el *teach-pendant*

para programar el robot Fanuc significa que se está realizando la programación en línea. Esta sección se justifica debido a que para poder llevar a cabo la PFL con el enfoque propuesto, es necesario que previamente se creen algunos programas en línea para su posterior uso.

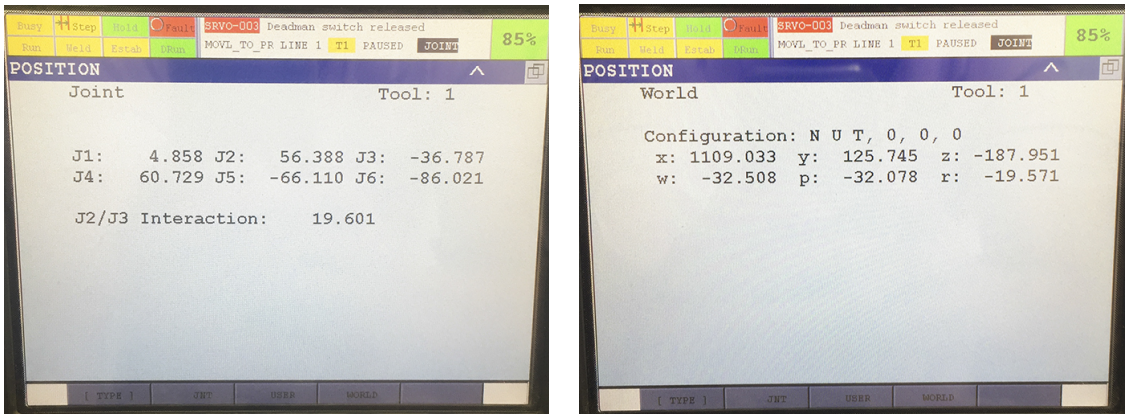
Para generar programas mediante este enfoque, es necesario en primer término que el usuario pueda manipular el robot con los botones de dicho dispositivo (Anexo B), para lo cual debe sujetar y permanecer oprimiendo un interruptor del *deadman* para mover el robot. El método más eficiente para lograrlo es el modo JOINT, debido a que el controlador no tiene que calcular una trayectoria lineal o mantener la punta de la antorcha de soldadura en línea con un plano. Este modo permite los movimientos independientes de cada articulación del robot, uno a la vez, ya sea en dirección positiva o negativa. Los valores angulares de las articulaciones en este modo son expresados con $J1$, $J2$, $J3$, $J4$, $J5$ y $J6$. En cualquier instante durante el movimiento del robot, el usuario puede consultar en la pantalla del *teach-pendant* estos valores. En la Figura 6.3a se presenta la pantalla del *teach-pendant* desplegando esta información en grados.

También existe el método de mover el robot con respecto al mundo (modo WORLD). Este método permite mover la punta de la antorcha de soldadura con respecto al sistema de coordenadas del mundo, cuyo origen coincide con $\mathbf{\Sigma}_0$ del robot virtual (Figura 4.19). Este marco es el asignado por default y la orientación de la antorcha de soldadura puede ser:

- Ya**W** - (w) : Rotación en x.
- **P**itch - (p) : Rotación en y.
- **R**oll - (r) : Rotación en z.

Los movimientos mientras se usa el marco del mundo están basados en el sistema de coordenadas Cartesiano (x-y-z). El robot moverá varias articulaciones simultáneamente para mantener un movimiento lineal y así alcanzar un punto en el espacio. La posición de la antorcha de soldadura es expresada en tres dimensiones,

las cuales son medidas en milímetros a partir del origen del marco del mundo en las direcciones x, y, z. Por su parte, la orientación de la antorcha de soldadura en grados se expresa en w, p y r. Esta información se despliega en todo momento en la pantalla del *teach-pendant* y en la Figura 6.3b es mostrada una imagen al respecto.



(a) **(b)**

Figura 6.3. Pantalla del *teach-pendant* con información del robot.
((a) modo JOINT; (b) modo WORLD)

6.1.2.1 Instrucciones de movimiento

Un programa puede contener una o más instrucciones de movimiento, que a su vez, cada una de éstas está constituida por cuatro variables. En este contexto, una instrucción de movimiento dirige al robot a moverse de una forma específica a una posición deseada en la estación de trabajo usando una velocidad determinada.

Una vez que el usuario consigue mediante el *teach-pendant* la posición deseada del robot, la cual implica valores puntuales en $\{J1, J2, J3, J4, J5, J6\}$ y/o en $\{x, y, z, w, p, r\}$, puede efectuar el registro de una instrucción de movimiento. Mientras se estén registrando instrucciones de movimiento en el programa, automáticamente se están almacenando en un archivo en la memoria del controlador. A este tipo de archivos se le denomina programa TP, dado que ésta es su extensión. Es un archivo binario que sólo se puede ver y editar mediante el *teach-pendant*.

Una instrucción de movimiento incluye las siguientes variables:

1. **Tipo de movimiento**.

   Establece cómo se mueve el robot a una posición destino. El programador puede elegir entre tres tipos:

   - Joint (J): Ocasiona que el robot mueva todas las articulaciones requeridas para alcanzar la posición de destino simultáneamente. El movimiento de cada articulación comienza y termina al mismo tiempo. La punta de la antorcha de soldadura traza un arco durante la trayectoria.
   - Lineal (L): Provoca que el robot mueva la punta de la antorcha de soldadura en línea recta desde la posición inicial a la final.
   - Circular (C): Origina que la punta de la antorcha se mueva en un arco desde la posición inicial a través de un punto intermedio hasta la posición final.

2. **Información posicional**.

   Describe la posición, orientación y configuración de la antorcha de soldadura cuando una instrucción de movimiento se añade al programa. Existen dos maneras de almacenar esta información:

   - Posición (P): Esta incluye el número de posición n, que indica su ubicación dentro del programa. Esta localidad sólo está disponible en el programa que existe y donde fue guardado.
   - Registros de posición (PR): Pueden ser usados para almacenar posiciones globales con un índice m, tal como una posición de *home* definida por el usuario. También permiten que las posiciones estén predefinidas para el uso compartido de muchos programas.

3. **Velocidad**.

   Define qué tan rápido se mueve el robot a la posición. Se puede configurar dependiendo del tipo de movimiento seleccionado, la velocidad puede ser especificada en porcentaje (lo cual depende de si el interruptor de llave está en T1, T2 o en automático), mm/seg, cm/min, pulgadas/min, grados/seg o tiempo para ejecutar un movimiento.

4. **Tipo de terminación**.

   Dispone cómo termina el robot el movimiento a la posición. Hay dos tipos de terminación:

   - FINE: Causa que el robot se detenga en la posición destino antes de moverse a la próxima posición. La punta de la herramienta del robot acelera a la velocidad definida y entonces desacelera mientras se aproxima a la posición guardada y llega a una detención por completo, antes de continuar a la siguiente posición guardada.
   - CONTINUOUS (CNT): Permite al robot desacelerar mientras se aproxima a la posición final pero no se detiene en la posición guardada, antes acelera hacia la siguiente posición. El valor de 0 a 100 define qué tanto el robot se acerca a la posición destino. En 0 el robot está lo más cerca con una desaceleración máxima. En 100 el robot está lo más alejado con una mínima desaceleración; es la más lejana y más fiel a la velocidad del programa.

Tomando en cuenta los detalles de las variables recién dados, en la Figura 6.4 se indica una instrucción de movimiento con un tipo de movimiento lineal, almacenada en la posición 1 del programa, a una velocidad del 50% de la capacidad del robot, con una terminación FINE.

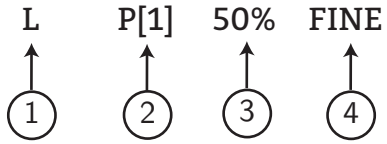


Figura 6.4. Instrucción de movimiento.

6.1.2.2 Instrucciones de soldadura

Si lo que se desea es que además de moverse el robot, éste ejecute alguna tarea específica durante su movimiento, se debe agregar una variable al final de una instrucción de movimiento para hacer lo propio. Para fines de este trabajo, las

tareas que se pueden especificar en este caso son dos, la primera define el inicio del cordón de soldadura y la segunda el final del mismo, por lo que la quinta variable convierte una instrucción de movimiento en una instrucción de soldadura.

Ahora bien, la calidad de la unión de soldadura depende de ciertos parámetros, de los cuales, unos son determinados por la pose de la antorcha de soldadura con respecto al punto de trabajo (ángulo de ataque o de arrastre), mientras que otros tienen que ser definidos en el controlador del robot, tales como la corriente de soldadura por arco, el voltaje del arco, la velocidad de alimentación del microalambre, la velocidad del robot, la longitud del arco y por último el caudal del gas [97]. En ese sentido, la configuración de la soldadura se lleva a cabo en ArcTool™ mediante la creación de procedimientos de soldadura.

Un procedimiento de soldadura es un archivo que define cómo una soldadura va a ser ejecutada. Contiene horarios de soldadura, entradas y salidas de la soldadura, ampliación de la información y otros elementos de configuración que habilitan características y ajustes de tiempo. Los procedimientos de soldadura pueden ser modificados, copiados, almacenados, cargados o borrados como archivos [98]. Se pueden definir y usar múltiples procedimientos de soldadura, cada uno tendrá su propio conjunto de horarios de soldadura.

La soldadura de arco usa horarios de soldadura (*schedules*) para controlar las condiciones de soldadura. Éstos definen la información que determina cómo será realizada la soldadura. La relación entre el horario de soldadura y el proceso de soldadura es más clara en un procedimiento de soldadura. La tabla del horario de soldadura (Tabla 24) y la definición del proceso de soldadura en sí forman parte de una entidad, el procedimiento de soldadura.

Los procedimientos de soldadura utilizan las instrucciones del *teach-pendant* llamadas *Weld Start* y *Weld End* para indicar el inicio del cordón de soldadura y el fin del mismo respectivamente. Ambas instrucciones requieren dos parámetros. El primer parámetro es el número de procedimiento de soldadura. Este número

es verificado cuando se introduce y confirma que el procedimiento está presente en el controlador. El editor no permitirá introducir un número inválido. El segundo parámetro es el número de horario de soldadura en el procedimiento de soldadura. Este número es verificado que exista dentro del procedimiento de soldadura especificado.

Tabla 24. Elementos de un horario de soldadura.

Parámetro	Descripción
Weld schedule [*num*][*com*]	Muestra el número del horario para el cual la información actualmente está siendo desplegada y el comentario sobre dicho horario.
Voltage (*Volts*)	Cantidad de voltaje.
Current (*Amps*)	Amperaje.
Wire Feed (*IPM*)	Velocidad de alimentación del microalambre.
Trim (*Volts*)	Este elemento es un comando de referencia para la longitud del arco eléctrico.
Travel Speed (*IPM*)	Velocidad a la cual el robot se moverá durante la soldadura, en unidades definidas en la pantalla SETUP Weld System.
Feedback Voltage	Indica el voltaje de realimentación de la última soldadura.
Feedback Current	Indica la corriente de realimentación de la última soldadura.

En [99] se hizo un estudio cuidadoso de las numerosas variables de soldadura robotizada de comienzo y final del cordón, en el proceso se seleccionaron las posibles variables activas durante la soldadura robotizada del cordón. Para esto, se definió un procedimiento de soldadura donde se configuró el tipo de soldadura de acuerdo a la información del fabricante del equipo de soldadura (Lincoln Electric), como el material y diámetro del electrodo consumible, el tipo de gas de protección, el control del arco y su rango máximo, que sirvieron de referencia para los experimentos.

En el procedimiento de soldadura en cuestión se configuraron los horarios de soldadura. Aquí, se definieron los parámetros de velocidad de viaje/soldadura,

voltaje, corriente y velocidad de alimentación del microalambre con los que se consiguieron cordones de soldadura con una buena terminación superficial. Los valores de estos parámetros se presentan en la Tabla 25. Las velocidades por defecto del sistema son IPM (pulgadas por minuto) y *Trim* es una instrucción ajustable que permite al operador configurarlo como un valor de voltaje que ajusta la longitud del arco eléctrico.

Tabla 25. Valores del horario de soldadura.

Parámetro	Valor
Wire Feed (*IPM*)	450
Trim (*Volts*)	0.9
Travel Speed (*IPM*)	47

Con el procedimiento de soldadura y el horario de soldadura ya definidos y almacenados en el controlador con sus respectivos índices, pueden ser invocados en una instrucción de soldadura. En la Figura 6.5 se presenta una instrucción de este tipo, en la cual se indican los cuatro parámetros de la instrucción de movimiento de la Figura 6.4, más el parámetro #5 que sugiere la ejecución de una tarea específica durante el movimiento del robot. En este caso, se indica que el robot comience a soldar invocando el procedimiento de soldadura #2 con su horario de soldadura #1.

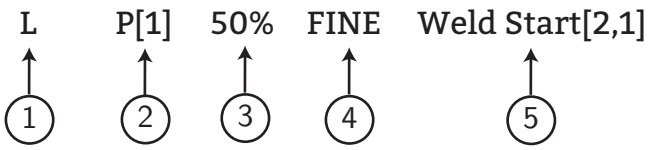


Figura 6.5. Instrucción de soldadura.

6.2 Lenguaje de programación del robot

KAREL es el lenguaje del robot para la arquitectura del sistema de los manipuladores Fanuc. El sistema KAREL de Fanuc consta de un robot, un controlador y un software

de sistema. Realiza tareas industriales utilizando programas escritos en el lenguaje de programación KAREL, el cual puede manipular datos, controlar y comunicarse con equipos relacionados e interactuar con un operador [100].

El lenguaje de programación KAREL es una combinación práctica de las características lógicas de lenguajes de alto nivel en inglés, como Pascal y PL/1, con la eficacia comprobada de fábrica de los lenguajes de control de máquinas. KAREL incorpora estructuras y convenciones comunes a los lenguajes de alto nivel, así como características desarrolladas especialmente para aplicaciones de robótica. A continuación se da una lista de características del lenguaje KAREL.

- Creación de varias rutinas que permiten utilizar muchas funciones del controlador.
- Manipulación de eventos de entrada/salida y variables independientes de la secuencia de programas TP.
- Entradas/salidas mediante operación con archivos o teclas.
- Intercambio de datos desde el puerto serial o Ethernet.

Además, KAREL tiene funciones de manipulación de eventos de vectores, datos de posicion y entrada/salida independiente del controlador, así como funciones estándar de creación que controlan al robot. Los usuarios pueden crear sus propios sistemas utilizando estas funciones.

El programa creado con KAREL, es un archivo de texto ASCII y es llamado programa KL debido a su extensión. Este programa es lo mismo que un programa creado con el *teach-pendant* (programa TP) en el sentido que ambos pueden ser ejecutados desde el controlador. Sin embargo, el propósito de su uso tiene puntos diferentes. Por un lado, el programa TP es para ejecutar movimientos del robot e instrucciones de aplicación, mientras que los programas KL proporcionan una forma de crear y manipular datos de posición, pero no admite instrucciones de movimiento. Los programas TP pueden ser creados, editados, y ejecutados en el *teach-pendant*, pero los programas KL no

pueden ser creados ni editados en el controlador del robot. La creación del programa KL en una PC y su conversión (llamada compilación aquí y más tarde) y la ejecución del programa es ejecutada tras cargarlo en el controlador. El programa TP puede ser cambiado según necesidad de la operación habitual, pero el programa KL nunca puede ser cambiado durante la operación habitual.

Las características de los programas TP es que controlan la secuencia de los movimientos del robot. Por otro lado, los programas KL controlan otras funciones excepto el control de movimiento. KAREL permite realizar al usuario o al sistema original funciones que no necesiten cambiar el software del robot, es decir, con KAREL el usuario puede acceder a programas, configuraciones y aplicaciones en el controlador pero sin poder modificarlos.

Partiendo de que en el ambiente virtual se genera un programa KL, en la Figura 6.6 se plasma el ciclo de desarrollo de este tipo de programas, el cual inicia con la creación del mismo en una PC; después debe pasar por un proceso de compilación/traducción en lenguaje robot; posteriormente el programa compilado se carga en el controlador del robot para al final ejecutar el programa con el *teach-pendant*.

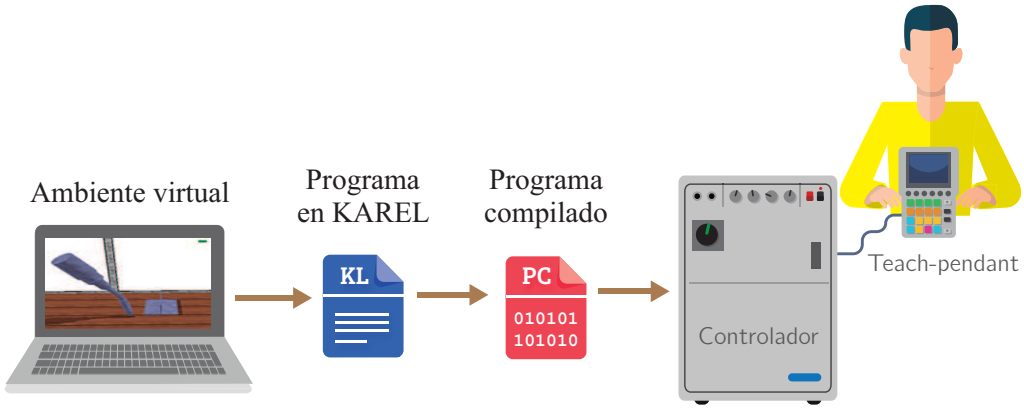


Figura 6.6. Ciclo de desarrollo de un programa KL.

6.3 Calibración

Si los resultados de la simulación son satisfactorios, el programa ya puede cargarse en el controlador del robot. Sin embargo, antes de la prueba y la ejecución, se debe aplicar un paso importante, que se llama calibración. Un programa fuera de línea no se puede ejecutar correctamente en un robot real sin calibración. Este proceso permite sincronizar las poses de la antorcha de soldadura y de la pieza de trabajo entre el robot virtual y el robot real.

La calibración aumenta la precisión de un robot. Es un proceso en el que las mediciones tomadas en el mundo real se comparan con mediciones conocidas en el ambiente virtual, para determinar la precisión de las mediciones hechas y aumentar esta precisión al compensar estos errores. La calibración puede reducir el error en el movimiento de un robot sin modificarlo físicamente.

El proceso de calibración en este sistema se efectuó experimentalmente en las placas del mundo real, considerando el siguiente procedimiento:

1. Con respecto a Σ_0 se localizó con la punta de la antorcha de soldadura el origen de Σ_{PR} (marco de las placas en el mundo real) y también se localizó un punto sobre x_{PR} y otro sobre y_{PR} (ambos de Σ_{PR}). Los tres vectores localizados son ${}^0P_{o_{PR}}$, ${}^0P_{x_{PR}}$ y ${}^0P_{y_{PR}}$ respectivamente. Las coordenadas Cartesianas (x, y, z) de estos puntos se leyeron de la pantalla del *teach-pendant* (Figura 6.3b). En la Figura 6.7 se observa a la punta de la antorcha colocada en el origen (o_{PR}) de unas placas en T en el mundo real. Asimismo, se indican los puntos sobre los ejes x_{PR} e y_{PR}.

2. Se calcularon los vectores unitarios ${}^0\hat{x}_{PR}$ y ${}^0\hat{y}_{PR}$ mediante las Ecuaciones 6.1 y 6.2.

$$ {}^0\hat{x}_{PR} = \frac{{}^0P_{x_{PR}} - {}^0P_{0_{PR}}}{\left\| {}^0P_{x_{PR}} - {}^0P_{0_{PR}} \right\|} \tag{6.1} $$

$$ {}^{0}\hat{y}_{PR} = \frac{{}^{0}P_{y_{PR}} - {}^{0}P_{0_{PR}}}{\left\| {}^{0}P_{y_{PR}} - {}^{0}P_{0_{PR}} \right\|} \tag{6.2} $$

Además:

$$ {}^{0}\hat{z}_{PR} = {}^{0}\hat{x}_{PR} \times {}^{0}\hat{y}_{PR} \tag{6.3} $$

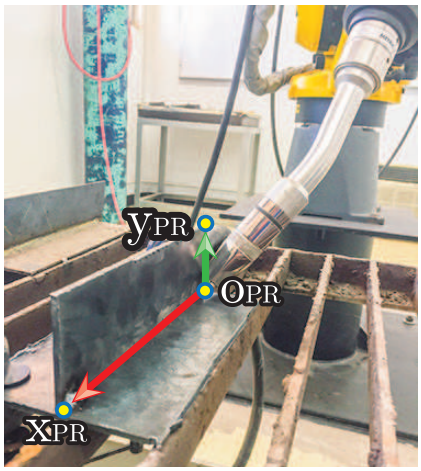


Figura 6.7. Localización de puntos para la calibración.

3. Por lo tanto, con ${}^{0}\hat{x}_{PR}$, ${}^{0}\hat{y}_{PR}$, ${}^{0}\hat{z}_{PR}$ y ${}^{0}P_{0_{PR}}$ se definió la matriz homogénea (Ecuación 6.4) que determina la pose del marco Σ_{PR} con respecto al marco Σ_0.

$$ {}^{0}_{PR}T = \begin{bmatrix} {}^{0}\hat{x}_{PR} & {}^{0}\hat{y}_{PR} & {}^{0}\hat{z}_{PR} & {}^{0}P_{0_{PR}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.4} $$

4. Por otra parte se conoce ${}^{0}_{P}T$ que especifica la pose del marco de las placas (Σ_P) con respecto a la base del robot en el mundo virtual. Esa pose la define el usuario arbitrariamente. También se conocen las matrices ${}^{P}_{H_i}T$ que especifican las poses del marco de la antorcha correspondientes a los puntos p_i de la ruta deseada ($i = 1, ..., np$), con respecto al marco Σ_P en el mundo virtual.

5. Además, se sabe que

$$ {}^{0}_{P}T = {}^{0}_{PR}T \; {}^{PR}_{P}T \tag{6.5} $$

Como ${}^{0}_{P}T$ y ${}^{0}_{PR}T$ son conocidas, entonces la matriz de calibración ${}^{PR}_{P}T$ se calcula con este producto.

$$ {}^{PR}_{P}T = {}^{PR}_{0}T \ {}^{0}_{P}T \tag{6.6} $$

6. Considerando que ${}^{P}_{H_i}T$ son las matrices que se especifican de las poses deseadas de la antorcha con respecto a Σ_P en el mundo virtual. Es decir, son las que se obtienen con la interfaz háptica. Se calcula ${}^{0}_{6}T$ mediante:

$$ {}^{0}_{6_i}T = {}^{0}_{PR}T \ {}^{PR}_{P}T \ {}^{P}_{H_i}T \ {}^{H}_{6}T \tag{6.7} $$

La matriz ${}^{0}_{6_i}T$ es la que define la pose deseada del marco de la antorcha en el punto p_i con respecto al marco Σ_0 del mundo real. Esta matriz permite resolver el modelo inverso de posición en el mundo real ya que funge como la matriz *snap* (Ecuación 4.5). Por lo tanto este modelo da los ángulos θ_i requeridos por el robot Fanuc para ejecutar la tarea en el mundo real.

En el ambiente virtual se ha habilitado una opción para visualizar simultáneamente las placas tanto en su pose original como en su pose en el mundo real. En la Figura 6.8 se observan estos dos tipos de placas, están aquéllas cuya posición y orientación son las que tenían al momento de realizarse la definición de puntos nodo con la interfaz háptica, mientras que las otras son las placas con una pose ya calibrada.

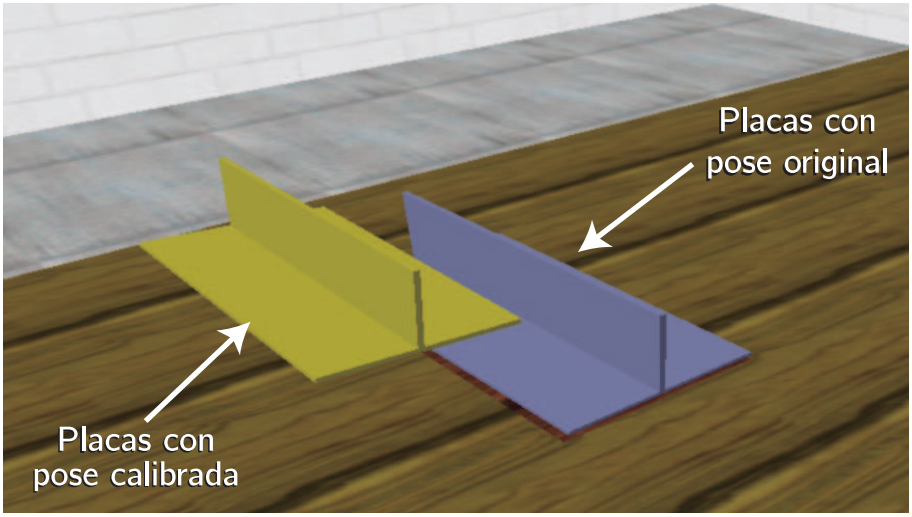


Figura 6.8. Placas a soldar calibradas.

6.4 Generación de programas en el ambiente virtual

Una de las funciones del sistema de PFL presentado en este trabajo consiste en la creación de programas a partir de los movimientos del robot representados en la simulación, considerando previamente la calibración del sistema para corregir las diferencias entre el mundo virtual y el mundo real. Se propone que una vez que finalice la simulación, se habrá generado un archivo de texto con una estructura en KAREL (programa KL).

Las consignas que eventualmente se le darán al robot Fanuc para que éste lleve a cabo las tareas programadas, están basadas en los valores articulares del mismo; esto es, los valores de $J1$, $J2$, $J3$, $J4$, $J5$ y $J6$ son utilizados como parámetros para definir los movimientos del robot en el programa KL que se ha de generar en esta sección.

Es importante señalar que los ángulos de las articulaciones del robot en el mundo virtual y en el mundo real presentan algunas diferencias entre ellos que se deben considerar antes de crear el programa KL. Considerando a θ_1, θ_2, θ_3, θ_4, θ_5 y θ_6 como los valores articulares del robot virtual, su correspondencia con los valores

articulares del robot Fanuc se define en la Tabla 26. Estos ajustes se deben hacer para compensar la disparidad de los modelos cinemáticos de ambos robots.

Tabla 26. Valores articulares del robot Fanuc y del robot virtual.

Robot Fanuc	Robot virtual
$J1$	θ_1
$J2$	$\theta_2 - 90°$
$J3$	$J2 + \theta_3$
$J4$	$-\theta_4$
$J5$	θ_5
$J6$	$-\theta_6$

Un caso especial se presenta con $J3$, dado que su valor angular es relativo a Σ_0. Esto es debido al diseño de algunos robots Fanuc, donde $J3$ está enlazado con $J2$. Si únicamente cambia el valor de $J2$, entonces $J3$ permanece con el mismo ángulo con respecto a la base del robot. Si se quiere obtener el valor de $J3$ se debe agregar $J2$ al valor actual de θ_3.

Para que el programa KL pueda hacer que el robot Fanuc mueva sus articulaciones, debe haber una forma de especificar la información posicional del robot a través del programa. Dado que un programa KL no brinda funciones para el control del movimiento del robot Fanuc, fue preciso utilizar los registros de posición para conseguir tal fin.

En el entendido que los registros de posición almacenan información posicional global en el controlador y que cualquier programa puede acceder a ellos para dirigir el movimiento del robot, se reservaron algunos espacios en estos registros para el buen funcionamiento de los programas KL. En total, fueron tres registros de posición dedicados a la ejecución del programa generado fuera de línea por el sistema. El primero de ellos, cuyo índice es el #10, se ocupa de asignar la información posicional requerida para ejecutar las instrucciones de movimiento. Otro registro de posición

requerido es el que posee el índice #11, el cual se emplea para definir los movimientos del robot durante el inicio del cordón de soldadura, mientras el del índice #12 está destinado al fin del cordón.

6.4.1 Programas auxiliares

En el controlador se crearon tres programas en línea que echaron mano de los tres registros de posición y que posteriormente serían invocados dentro del programa KL para la consecución de los movimientos del robot Fanuc.

Primeramente se creó el programa TP llamado *movl_to_pr*, el cual sólo incluye una línea:

L PR[10] 100mm/sec FINE

Esta instrucción de movimiento indica que el robot moverá la antorcha de soldadura con un movimiento lineal de tal forma que la información posicional destino sea la que se estableció en el registro de posición #10 a una velocidad de 100 mm/seg y con una terminación FINE.

De manera similar al programa anterior, fue creado el programa TP nombrado *m_weldstart*, cuyo contenido es también una sola línea:

L PR[11] 25mm/sec CNT100 Weld Start[2,1]

La instrucción anterior es utilizada cuando se desea iniciar el cordón de soldadura, por ende es una instrucción de soldadura. Los parámetros de la misma muestran que el robot desplazará la antorcha de soldadura linealmente de acuerdo a lo establecido en el registro de posición #11, cuya información posicional indicará el inicio del cordón de soldadura a una velocidad de 25 mm/seg, con una terminación CONTINUOUS (con 100 de desaceleración) y utilizando el procedimiento de soldadura #2 con su horario de soldadura #1.

Un programa TP llamado *m_weldend* también fue generado con el *teach-pendant* para su uso posterior. Al igual que los anteriores este programa consta de únicamente una línea:

L PR[12] WELD_SPEED CNT100 Weld End[2,1]

Esta instrucción también es una instrucción de soldadura, misma que se ha definido para finalizar el cordón, para lo cual el robot moverá la antorcha de soldadura de forma lineal considerando la información posicional del registro de posición #12, donde se llevará acabo el fin del cordón de soldadura, con una terminación CONTINUOUS (con 100 de desaceleración) y utilizando el procedimiento de soldadura #2 con su horario de soldadura #1. WELD_SPEED confiere la velocidad definida en el horario de soldadura.

6.4.2 Programas en KAREL

Al finalizar la simulación de los movimientos del robot ejecutando la tarea de soldadura programada, el sistema genera un archivo de texto, el cual contiene un programa en el lenguaje KAREL. El enfoque con el que se generó este archivo se basa en registrar los valores articulares del robot Fanuc ($J1$, $J2$, $J3$, $J4$, $J5$ y $J6$) en cada punto nodo de la trayectoria programada.

En este escenario, los valores articulares del robot virtual (θ_1, θ_2, θ_3, θ_4, θ_5 y θ_6) en cada uno de los puntos nodo pueden ser obtenidos por el sistema. Así, cuando en la simulación el robot alcanza el primer punto nodo, los valores articulares respectivos del robot virtual son registrados por el sistema, posteriormente esta información debe ser procesada para calcular los valores articulares correspondientes para el robot Fanuc basándose en la información de la Tabla 26. El mismo procedimiento se efectúa con los demás puntos nodo. De tal forma que cada punto nodo dio origen a un bloque de instrucciones en KAREL, donde se involucran los registros de posición reservados. En la Figura 6.9 se presenta la asignación de los seis valores articulares del robot Fanuc a un arreglo unidimensional de números de punto flotante llamado *p_joints_a*, con la cual da inicio dicho bloque de instrucciones. Donde $J1i$, $J2i$, $J3i$, $J4i$, $J5i$ y $J6i$ son los valores articulares en grados en el punto nodo i.

```
p_joints_a[1] = J1i
p_joints_a[2] = J2i
p_joints_a[3] = J3i
p_joints_a[4] = J4i
p_joints_a[5] = J5i
p_joints_a[6] = J6i
```

Figura 6.9. Asignación de valores articulares en el programa KL.

En el cuerpo del programa también es requerida una instrucción que convierta el arreglo *p_joints_a* de cada punto nodo en un tipo de datos que represente la información posicional de cada articulación del robot Fanuc expresada en grados. Para esto, la función *CNV_REL_JPOS* hace la conversión del arreglo correspondiente a una variable de tipo *JOINTPOS6* llamada *p_joints*, la cual agrupa todos los valores articulares en cuestión. En la Figura 6.10 se presenta la sintaxis de esta función, donde *status* indica el estado de la operación intentada, si no es igual a 0, se produjo un error.

```
CNV_REL_JPOS(p_joints_a, p_joints, status)
```

Figura 6.10. Conversión de arreglo a *JOINTPOS6* en el programa KL.

Posteriormente los valores asignados a *p_joints* tienen que ser almacenados en el respectivo registro de posición del controlador para una posterior invocación del mismo y que las articulaciones del robot adquieran los ángulos que le son transmitidos mediante el programa en KAREL. Para hacer lo propio, se usó la función *SET_JPOS_REG*, la cual requiere que la información posicional que se le pasa como parámetro sea de tipo *JOINTPOS6*, donde es preciso llevar a cabo un proceso de diferenciación de los puntos nodo al momento de utilizar dicha función. En ese sentido, si en el punto nodo se requiere ejecutar una instrucción de movimiento, se debe utilizar «10» como primer parámetro en la función *SET_JPOS_REG*, dado que éste es el índice del registro de posición reservado para lo propio. Por otro lado, si se desea invocar instrucciones de soldadura en algún punto nodo, los parámetros a usar son «11» y «12» para el inicio del cordón de soldadura y el fin del mismo

respectivamente, puesto que son los índices de los registros de posición destinados para esos fines. En la Figura 6.11 se establece la sintaxis de la función en cuestión, donde ***RP*** es el índice del registro de posición donde se desea almacenar la información posicional de *p_joints* y *status* señala el estado de la operación intentada, si no es igual a 0, se produjo un error.

```
SET_JPOS_REG(RP, p_joints, status)
```

Figura 6.11. Asignación al registro de posición ***RP*** desde el programa KL.

Por otro lado, antes del cuerpo del programa KL se declararon rutinas por medio de las cuales se pueden invocar programas externos, en nuestro caso programas TP. Estas declaraciones incluyen las instrucciones *ROUTINE* y *FROM*, donde la primera de ellas (*ROUTINE*) define el nombre que identifica la rutina dentro del programa, mientras la cláusula *FROM* especifica el nombre del programa externo (Figura 6.12).

El mismo proceso utilizado para decidir los parámetros a incluir en la función *SET_JPOS_REG* es el utilizado para establecer el programa que se debe invocar con la rutina correspondiente. Como quedó explicado en la Sección 6.4.1, si se desea asignar una instrucción de movimiento el programa es *movl_to_pr*. Si se quiere aplicar el inicio del cordón de soldadura el programa es *m_weldstart* y para el fin del cordón es *m_weldend*. Para fines prácticos, cada rutina se nombró de igual forma que su respectivo programa a invocar.

```
ROUTINE movl_to_pr FROM movl_to_pr
ROUTINE m_weldstart FROM m_weldstart
ROUTINE m_weldend FROM m_weldend
```

Figura 6.12. Declaración de rutinas en el programa KL.

El bloque de instrucciones de cada punto nodo termina con la llamada a la rutina propiamente vinculada con el respectivo programa TP. Para esto, sólo se escribe el nombre de la rutina para invocar el programa almacenado en el controlador. Lo anterior implica que con *movl_to_pr* el programa KL invoca al programa TP del

mismo nombre y enviará las consignas al robot a moverse de acuerdo a los valores angulares que previamente se le pasaron al registro de posición #10. Análogamente, las llamadas a *m_weldstart* y *m_weldend* trazarán una trayectoria hacia el inicio del cordón de soldadura y el fin del mismo pero con la información posicional de los registros de posición #11 y #12 respectivamente.

La generación de los programas KL se lleva a cabo a partir de la suposición de que antes de ejecutarlos en el robot Fanuc, éste debe estar en la posición de *home* equivalente a la del robot virtual (Tabla 10). Asimismo, un bloque de instrucciones se agrega en cada uno de los programa KL, el cual da las consignas para llevar al robot Fanuc a la posición de *home* señalada después de que éste pasó por el último punto nodo de la trayectoria planificada.

En el Anexo C se encuentran dos programas KL generados por el sistema durante los experimentos efectuados en el caso de estudio presentado al final de este capítulo.

6.5 Traducción de programas a lenguaje del robot

Fanuc proporciona una herramienta interactiva para la creación y simulación de un ambiente virtual de una estación de trabajo. Esta herramienta se llama ROBOGUIDE y es el software para la PFL de los robots de esta compañía que el fabricante ofrece en el mercado.

Teniendo en cuenta la naturaleza competitiva de la industria de la tecnología robótica, Fanuc hace un gran esfuerzo para garantizar que sólo los clientes con licencia tengan acceso a los recursos de soporte de Fanuc y a sus kits de desarrollo de software. De ahí que ROBOGUIDE incluya el único método para traducir programas en KAREL a archivos compilados en código máquina denominado *p-code* (programas PC) que son ejecutables en el controlador. En este caso, el compilador *ktrans* es el que se encarga de la conversión de los programas KL al lenguaje del robot y sólo está disponible con la instalación de ROBOGUIDE.

Antes de traducir los programas en KAREL al código interno del robot, *ktrans* ejecuta un análisis que se encarga de la comprobación de la corrección de los programas mediante la búsqueda de errores en el código fuente. El proceso de compilación comienza en la primera línea del programa y continúa hasta que encuentra un error o traduce el programa con éxito. Si se encuentra un error, el compilador intenta continuar comprobando el programa, pero no se generará ningún archivo *p-code*. Después de una traducción exitosa, *ktrans* muestra un mensaje al respecto y crea un archivo *p-code*.

Como se observa en la Figura 6.13, *ktrans* utiliza la ventana de comandos de Windows® para cumplir con su cometido. A nivel de archivos, *ktrans* es un ejecutable (EXE) que requiere que los programas KL que se desean compilar se encuentren en su mismo directorio. Asimismo, el archivo compilado que se genera con cada programa KL utilizando este procedimiento, se guardará en dicho directorio y utilizará el nombre del archivo del programa KL pero con la extensión PC. Este archivo contiene una representación interna del programa en KAREL y la información que el sistema necesita para vincular el programa con datos y variables de las rutinas. El compilador *ktrans* es llamado a través de una línea de comandos donde sólo se necesita acompañarse del nombre del programa KL y éste será compilado, enviando los mensajes correspondientes en caso de errores o bien, de una traducción exitosa.

Los programas *p-code* son archivos binarios cuyo contenido no puede ser ni visualizado ni editado en una computadora personal. Lo mismo ocurre en el controlador del robot Fanuc cuando dichos programas se cargan en él. Sin embargo, la compatibilidad de estos archivos con el controlador está garantizada, dando la facultad al programador de ejecutar estos programas mediante el *teach-pendant*.

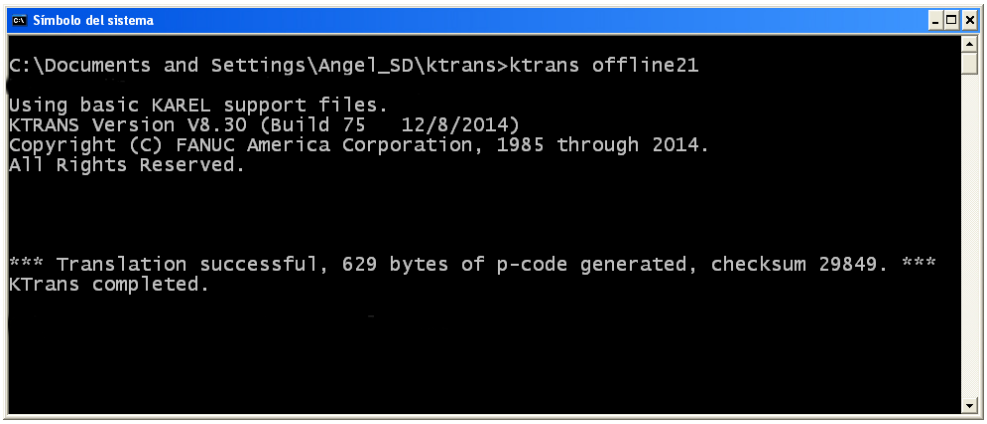


Figura 6.13. Línea de comando con ejecución de *ktrans*.

6.6 Transmisión de programas al robot

Los programas *p-code* deben cargarse en el controlador del robot Fanuc para que se puedan ejecutar. El método elegido para transmitir estos programas fue mediante la utilización de dispositivos de memoria flash USB. Entonces, los archivos PC se guardaron en una memoria de este tipo y se cargaron en el controlador a través de cualquiera de sus dos puertos USB, uno localizado en el propio controlador (Figura 6.14a) y otro en la parte lateral del *teach-pendant* (Figura 6.14b).

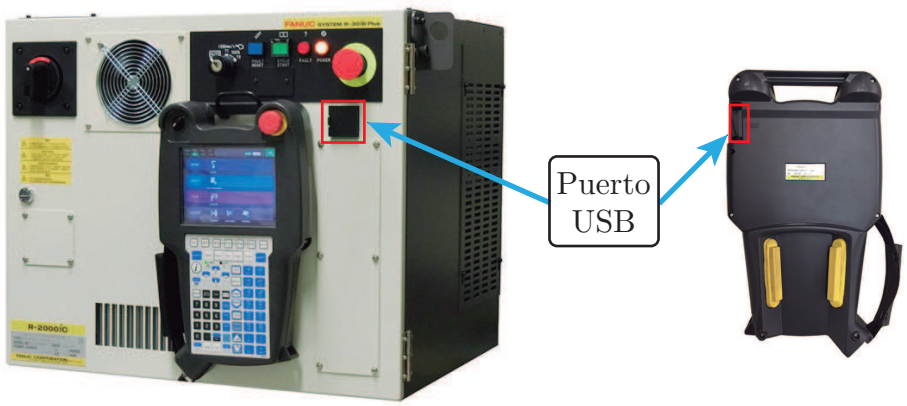


Figura 6.14. Puertos USB en el sistema del robot Fanuc.

El controlador admite unidades de memoria flash USB de hasta 2 GB de tamaño en

formato FAT. Las memorias USB con cualquier función de seguridad o cifrado no son compatibles. Las tarjetas de memoria de más de 2 GB no pueden formatearse FAT, deben formatearse FAT32.

Independientemente del puerto USB seleccionado para transferir los archivos PC, el proceso de cargar programas *p-code* desde una memoria flash USB se efectúa mediante los botones del *teach-pendant* (Anexo B) de la siguiente manera:

1. Almacenar los programas PC en la raíz del directorio de la memoria flash USB.

2. Introducir esta memoria en alguno de los puertos USB.

3. Pulsar el botón MENU y selecciónar «7 FILE» presionando el botón ENTER.

4. Si la memoria USB no está seleccionada como dispositivo de archivos, pulsar el botón F5 «UTIL» y seleccionar «SET DEVICE» con el botón ENTER.

5. Seleccionar de la lista visualizada «UD1:» en el caso de utilizar el puerto del controlador o «UT1:» cuando se use el puerto ubicado en el *teach-pendant*.

6. Para facilitar la localización de los programas *p-code* en la pantalla, se aplica un filtro para mostrar únicamente los archivos del tipo PC pulsando el botón F2 «DIR» y con el botón ENTER elegir «*.PC».

7. Situar el cursor de la pantalla en el programa deseado y pulsar el botón F3 «LOAD».

8. Confirmar dicha selección pulsando el botón F4 «YES».

Los programas PC cargados son visualizados en la pantalla del *teach-pendant* al presionar el botón SELECT, dispuestos a ser ejecutados.

6.7 Ejecución de programas en el robot

Después de que se haya seleccionado un programa de una lista visualizada en la pantalla del *teach-pendant*, la ejecución de dicho programa comienza en la primera línea ejecutable. El controlador del robot posee un compilador que busca errores al momento de ejecutar los programas *p-code*. En este sentido, si en el programa se intenta invocar a una rutina inexistente en el propio controlador, surge un error de ejecución y el programa no correrá. Asimismo, el controlador posee mecanismos de seguridad que detectan problemas como singularidades y puntos inalcanzables para la antorcha de soldadura, en estos casos se detiene el movimiento del robot durante la ejecución de programas.

Ahora bien, una práctica en aras de la seguridad en la estación de trabajo consiste en ejecutar los programas en primera instancia, sin habilitar la soldadura, es decir, antes de ejecutar los programas para que efectúen las tareas de soldadura, se deben validar los movimientos del robot sin activar el equipo de soldadura. La validación referida se lleva a cabo dado que puede haber diferencias considerables entre los movimientos del robot en la simulación en el ambiente virtual y los que el robot Fanuc efectúa al ejecutar los programas *p-code*.

La validación de movimientos del robot implica en primer término la corroboración de trayectorias libres de colisiones. Enseguida, se debe comprobar que las trayectorias del robot Fanuc coincidan con las que se observaron en la simulación en el ambiente virtual. Para estos dos casos de validación, una alternativa es la ejecución de los programas por pasos (STEP).

La ejecución de programas por pasos ejecuta las instrucciones de movimiento individuales del programa, una por una. Utiliza los botones del *teach-pendant* para dar un paso al programa activo que se indica en su pantalla (Anexo B). Resulta conveniente probar el programa paso a paso antes de que éste se ejecute en producción. Por lo que, primero se ejecuta el programa usando una velocidad baja para los movimientos del robot con la soldadura deshabilitada. Posteriormente,

los movimientos del robot se prueban con la velocidad programada también con la soldadura deshabilitada para verificar las posiciones del robot y los tiempos de las tareas.

El procedimiento para la ejecución de los programas por pasos con la ayuda del *teach-pendant* es el siguiente:

1. Presionar el botón SELECT.
2. Seleccionar el programa que se desea ejecutar y presionar el botón ENTER.
3. Presionar el botón STEP. En la pantalla del *teach-pendant* el indicador STEP se prenderá.
4. Mover el cursor a la primera línea del programa que se desea ejecutar. El programa empezará en la posición del cursor actual.
5. Presionar continuamente el interruptor *deadman*.
6. Probar una instrucción de movimiento del programa.
    - Para ejecutar una instrucción en la dirección hacia adelante, presione y mantenga presionado el botón SHIFT y presionar y soltar el botón FWD. Se debe mantener presionado el botón SHIFT continuamente hasta que la instrucción haya terminado de ejecutarse.
    - Para ejecutar una instrucción en dirección hacia atrás, presione y mantenga presionado el botón SHIFT y presione y suelte el botón BWD. Se debe mantener presionado el botón SHIFT continuamente hasta que la instrucción haya completado la ejecución.
7. Repetir el paso 6 para tantas instrucciones se deseen ejecutar.
8. Soltar el interruptor *deadman*.

Una vez que se corroboró que las trayectorias que efectuó el robot Fanuc fueron libres de colisiones, así como que éstas coinciden con las que el robot virtual realizó

en la simulación, existe otro método de ejecución de programas que permite realizar los movimientos totales del robot y observar las trayectorias completas del robot Fanuc sin que éste se detenga. Este método se denomina ejecución continua y consiste en correr un programa desde el principio hasta el fin sin que se suspendan los movimientos del robot. El procedimiento para la ejecución continua utilizando los botones del *teach-pendant* es el siguiente:

1. Presionar el botón SELECT.
2. Seleccionar el programa que se desea ejecutar y presionar el botón ENTER.
3. Deshabilitar la ejecución por pasos. Si el indicador STEP está encendido, presionar el botón STEP para deshabilitarlo.
4. Mover el cursor a la línea #1 del programa, éste empezará ejecutarse en la posición actual del cursor.
5. Presionar continuamente el interruptor *deadman*.
6. Establecer la velocidad deseada (Del 5% al 100% de la velocidad programada).
7. Verificar el estatus del programa en la parte de arriba de la pantalla del *teach-pendant*. Si aparece «PAUSED» entonces presionar el botón FCTN y seleccionar «ABORT (ALL)».
8. Mantener presionado el botón SHIFT y presionar y soltar el botón FWD. El botón SHIFT no se debe soltar hasta que el programa ha finalizado su ejecución.
9. Soltar el interruptor *deadman*.

Este procedimiento fue el utilizado en el caso de estudio que enseguida se presenta. Sólo que para efectuar los cordones de soldadura deseados, fue necesario encender la máquina de soldadura robotizada Lincoln Electric PowerWave i400 (Figura 6.15). Además se debió habilitar la soldadura con el botón WELD ENBL. Los botones

WIRE se utilizaron para desplazar el microalambre, WIRE + para fuera y WIRE - para adentro de la boquilla de la antorcha de soldadura.

Figura 6.15. Máquina de soldadura Electric PowerWave i400.

La Power Wave i400 es una tecnología de alto rendimiento y procesos de soldadura avanzados todo incluido en una fuente de poder invertida, eficiente, diseñada para operaciones de soldadura robotizada. Cuenta con armazón que permite el acceso para el mantenimiento de la parte encargada de la energía. Está diseñada para soportar el controlador Fanuc R30i y es utilizado principalmente para la manufacturación robotizada. El único interruptor que se encuentra al frente de la máquina es el interruptor de encendido y apagado; debe estar encendido el equipo para habilitar la soldadura en los programas a ejecutar.

6.7.1 Caso de estudio

Para probar la eficiencia del sistema de PFL descrito en este documento, se programó una tarea de soldadura a partir de la planificación de trayectorias con la interfaz háptica; donde la pose de las placas a soldar (Σ_P) con respecto a Σ_G durante este proceso quedó establecida en un principio en la Ecuación 3.1. En el caso de la mesa, la Ecuación 6.8 señala la matriz de transformación que define la pose de ésta (Σ_M)

con respecto a las placas dentro del ambiente virtual.

$$ {}^{P}_{M}T = \begin{bmatrix} 0 & 1 & 0 & rx_M \\ 0 & 0 & 1 & ry_M \\ 1 & 0 & 0 & rz_M \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.8} $$

Donde $rx_M = -19\ cm$, $ry_M = 0.32\ cm$ y $rz_M = -33\ cm$ considerando una posición aproximada de la mesa con respecto a las placas en el mundo real.

Mediante la manipulación con la interfaz háptica de la antorcha virtual con comportamiento dinámico, se generaron cuatro puntos nodo: n_1, n_2, n_3 y n_4. Donde n_1 es un punto flotante que permitirá al robot acercarse a la tarea de soldadura; n_2 representa el punto de inicio del cordón de soldadura; n_3 funge como el punto de fin del cordón y n_4 es un punto flotante destinado a que el robot se aleje del cordón para volver a la posición de *home*.

La pose de la antorcha virtual al momento de generar n_1 es ilustrada en la Figura 6.16a. En la Figura 6.16b se capturó la imagen de la antorcha virtual generando n_2, mientras que en la Figura 6.16c se hace un acercamiento para observar la creación de n_3. Finalmente, la punta de la antorcha virtual es visualizada en la Figura 6.16d durante la generación de n_4.

Cabe recordar que cada uno de estos puntos nodo poseen las coordenadas operacionales que representan la posición de la punta de la antorcha con su respectiva orientación con respecto al marco de las placas (Σ_P). Los valores de estas coordenadas fueron registrados en un archivo de texto para su posterior uso y son presentados en la Tabla 27.

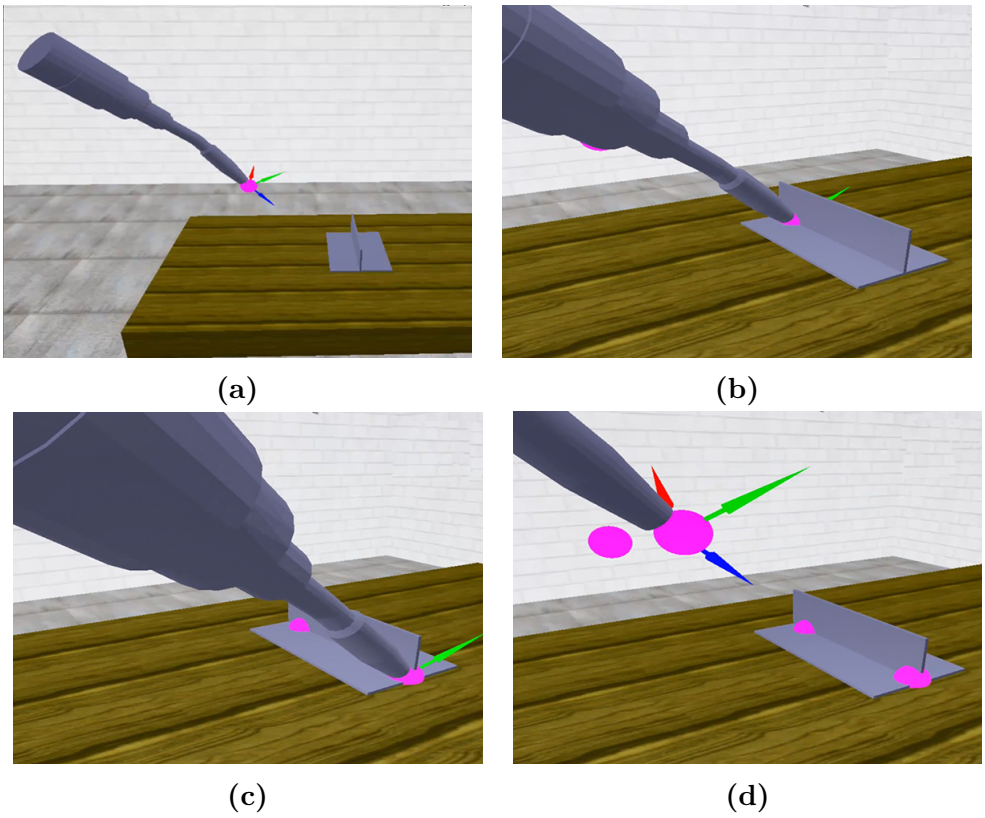


Figura 6.16. Antorcha virtual generando puntos nodo.

Con la lectura del archivo de texto se obtuvieron las coordenadas operacionales de cada uno de los puntos nodo para poder realizar una interpolación lineal entre ellos y resolver la cinemática inversa que define los movimientos que efectuará el robot al seguir la trayectoria calculada para ejecutar la tarea programada. Con la simulación se verificó que los movimientos del robot virtual consiguieron que la antorcha de soldadura alcanzara todos los puntos nodo sin problema alguno.

Tabla 27. Coordenadas operacionales de los puntos nodo.

Punto	Px (cm)	Py (cm)	Pz (cm)	Rx (°)	Ry (°)	Rz (°)
n_1	14.43	14.42	-18.35	55.40	15.45	18.39
n_2	16.13	-0.01	0.02	45.43	11.83	20.13
n_3	0.85	-0.01	0.03	47.14	8.58	22.09
n_4	1.64	17.11	-17.45	44.39	16.78	29.07

En el afán de hacer efectiva la tarea de soldadura programada en el ambiente virtual se dispuso de la estación de trabajo de soldadura robotizada del Laboratorio de Mecatrónica y Control del Instituto Tecnológico de la Laguna, cuyos componentes de interés se señalan en la Figura 6.17.

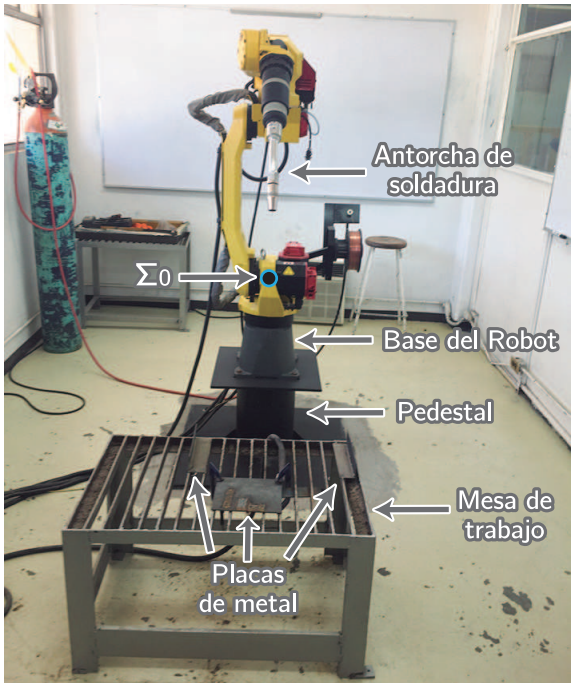


Figura 6.17. Estación de trabajo de soldadura robotizada del ITLag.

Hay algunos puntos a destacar de esta estación de trabajo. Uno de ellos lo representa el pedestal sobre el que está emplazado el robot Fanuc y cuya altura es 39 cm. Esta dimensión se consideró en el emplazamiento del robot virtual durante las simulaciones efectuadas en este caso de estudio. Otra peculiaridad radica en las placas de metal que se desean soldar. Éstas debieron pasar por un proceso de presoldado donde se les aplicó un punto de soldadura de forma manual en cada extremo de la junta de ambas placas, lo cual facilitó el proceso de experimentación. Además, la pose de la mesa de trabajo no se modificó durante los experimentos, permaneciendo la posición de ésta con respecto a Σ_0 en rx_{MR} = 81 cm, ry_{MR} = 24 cm y rz_{MR} = -37 cm.

Sin embargo, considerando las diferencias entre el mundo real con el ambiente virtual, se llevó a cabo el proceso de calibración para poder ejecutar la tarea de

soldadura programada en dos placas en T distintas, en consecuencia, la tarea inicial se transformó en dos tareas calibradas en el mundo real. Las placas en el mundo real se presentan en la Figura 6.18, las cuales fueron colocadas de forma arbitraria sobre la mesa de la estación de trabajo.

Figura 6.18. Placas de soldadura reales en estación de trabajo.
(*Vista superior*).

Aplicando la metodología de calibración descrita en la Sección 6.3 se obtuvieron en primera instancia los vectores unitarios ${}^0\hat{x}_{PR}$, ${}^0\hat{y}_{PR}$ y ${}^0\hat{z}_{PR}$ para las dos placas experimentales. En las Ecuaciones 6.9 y 6.10 se exponen los vectores unitarios resultantes que corresponden a las placas #1 y #2 respectivamente.

$$ {}^0\hat{x}_{PR1} = \begin{bmatrix} 0.9998 \\ -0.0101 \\ 0.0157 \end{bmatrix} \quad {}^0\hat{y}_{PR1} = \begin{bmatrix} -0.0068 \\ -0.0806 \\ 0.9967 \end{bmatrix} \quad {}^0\hat{z}_{PR1} = \begin{bmatrix} -0.0088 \\ -0.9967 \\ 0.0806 \end{bmatrix} \tag{6.9} $$

$$ {}^0\hat{x}_{PR2} = \begin{bmatrix} 0.7703 \\ -0.6377 \\ 0.0082 \end{bmatrix} \quad {}^0\hat{y}_{PR2} = \begin{bmatrix} -0.0361 \\ -0.0119 \\ 0.9993 \end{bmatrix} \quad {}^0\hat{z}_{PR2} = \begin{bmatrix} -0.6371 \\ -0.7700 \\ -0.0322 \end{bmatrix} \tag{6.10} $$

Posteriormente se conformaron las matrices de las dos placas experimentales con

respecto a Σ_0. De tal forma que para las placas #1, la matriz correspondiente está dada por la Ecuación 6.11, mientras que para las placas #2 por la Ecuación 6.12. Utilizando estas matrices se dibujaron estas dos placas en el ambiente virtual, las cuales se pueden ver por separado en las Figuras 6.19 y 6.20, donde también se indican las placas en las que se realizó la definición de los puntos nodo con la interfaz háptica, denominadas placas originales, para contrastar los resultados del proceso de calibración propuesto.

$$ {}^{0}_{PR1}T = \begin{bmatrix} 0.9998 & -0.0068 & -0.0088 & rx_{PR1} \\ -0.0101 & -0.0806 & -0.9967 & ry_{PR1} \\ 0.0157 & 0.9967 & -0.0806 & rz_{PR1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.11} $$

$$ {}^{0}_{PR2}T = \begin{bmatrix} 0.7703 & -0.0361 & -0.6371 & rx_{PR2} \\ -0.6377 & -0.0119 & -0.7700 & ry_{PR2} \\ 0.0082 & 0.9993 & -0.0322 & rz_{PR2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.12} $$

Donde x_{PR1} = 109.14 cm, y_{PR1} = -3.33 cm, z_{PR1} = -37.24 cm, x_{PR2} = 108.67 cm, y_{PR2} = -31.83 cm y z_{PR2} = -37.87 cm.

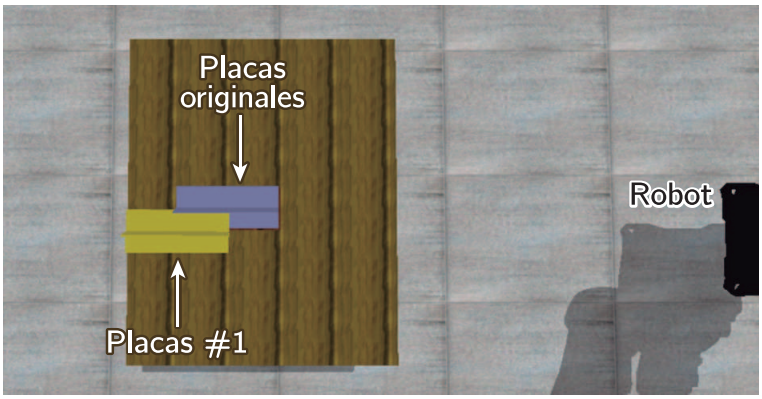


Figura 6.19. Placas #1 en el ambiente virtual.
(*Vista superior*).

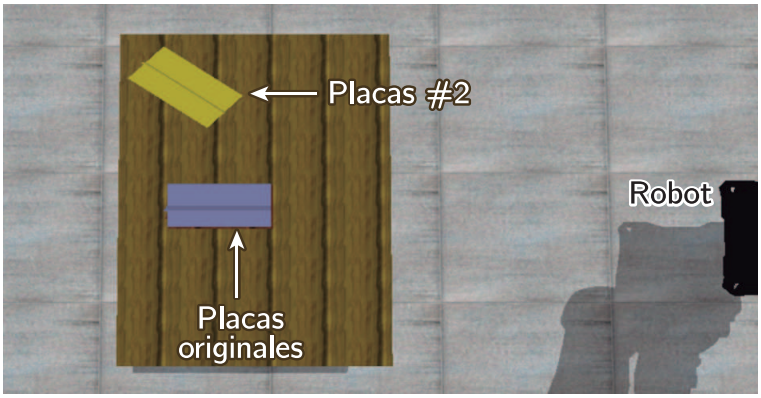


Figura 6.20. Placas #2 en el ambiente virtual.
(*Vista superior*).

A partir de las Ecuaciones 6.11 y 6.12 se obtuvieron las matrices de calibración correspondientes a las placas #1 y a las placas #2, quedando establecidas en las Ecuaciones 6.13 y 6.14 respectivamente.

$$ {}^{PR}_{P1}T = \begin{bmatrix} 1 & 0.0060 & 0.0093 & rx_{P1} \\ -0.0165 & 0.9967 & 0.0805 & ry_{P1} \\ -0.0088 & -0.0807 & 0.9968 & rz_{P1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.13} $$

$$ {}^{PR}_{P2}T = \begin{bmatrix} 0.7699 & 0.0202 & 0.6379 & rx_{P2} \\ -0.0269 & 0.9995 & 0.0196 & ry_{P2} \\ -0.6372 & -0.0322 & 0.7701 & rz_{P2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.14} $$

Con las matrices que resultaron del proceso de calibración se resolvió el modelo inverso de posición con la Ecuación 6.7, de este modo el robot virtual efectuó en las dos placas experimentales la tarea de soldadura que se programó en las placas originales en un intervalo de 5 segundos cada una. Consecuentemente, el sistema generó dos programas en KAREL, uno que contiene los valores articulares del robot Fanuc para ejecutar la tarea de soldadura en las placas #1 y el otro para hacer lo propio en las placas #2 (Anexo C). Ambos programas se compilaron con *ktrans* para posteriormente guardarlos en una memoria USB y cargarlos individualmente

en el controlador. Posteriormente, se verificaron los movimientos del robot Fanuc paso por paso para cada programa. Asimismo, se ejecutó cada programa de forma continua con el equipo de soldadura deshabilitado. Finalmente, se ejecutaron las tareas de soldadura para las dos placas experimentales con los resultados presentados a continuación.

6.7.2 Resultados

En primer término se han de mostrar los resultados en la simulación. Para esto, se hicieron capturas de pantalla del robot virtual al momento de que su antorcha pasa por los cuatro puntos nodo definidos con la interfaz háptica.

El sistema utiliza los valores calculados con la cinemática inversa θ_1, θ_2, θ_3, θ_4, θ_5 y θ_6 como entrada para calcular los valores angulares correspondientes del robot Fanuc ($J1$, $J2$, $J3$, $J4$, $J5$ y $J6$) de acuerdo a la Tabla 26 y son éstos los desplegados en la pantalla pese a que aparezcan las leyendas $Theta[1]$, $Theta[2]$, $Theta[3]$, $Theta[4]$, $Theta[5]$ y $Theta[6]$. Estos valores están en grados y son los que eventualmente serán incluidos en los programas en KAREL.

También se despliega la posición de la punta de la antorcha de soldadura en centímetros con respecto a Σ_0 representada por x_0, y_0 y z_0. Toda esta información es mostrada en las capturas de pantalla. Son un total de cuatro capturas por cada tarea en su respectiva placa experimental. De la Figura 6.21 a la Figura 6.24 corresponden a la tarea #1 y de la Figura 6.25 a la Figura 6.28 pertenecen a la tarea #2.

- **Simulación en la tarea #1.**

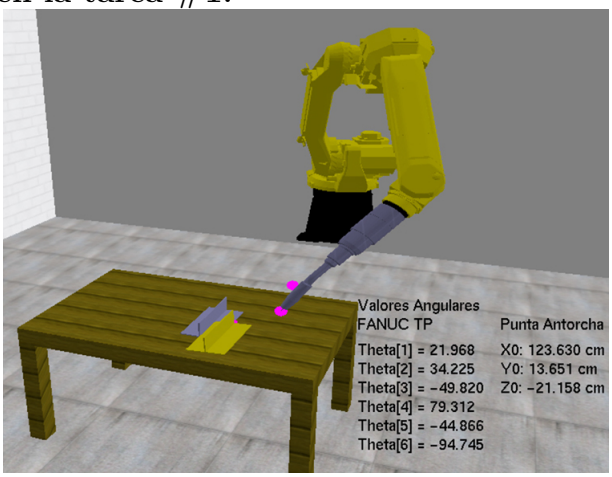


Figura 6.21. Robot virtual en n_1 de la tarea #1.

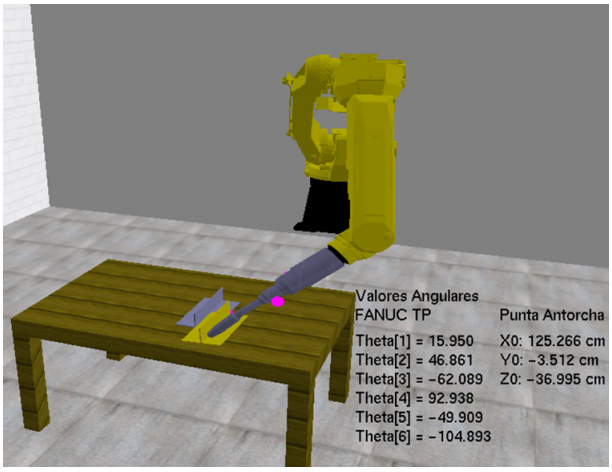


Figura 6.22. Robot virtual en n_2 de la tarea #1.

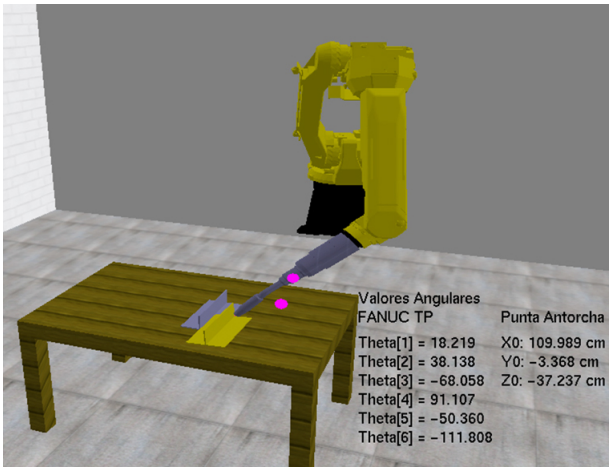


Figura 6.23. Robot virtual en n_3 de la tarea #1.

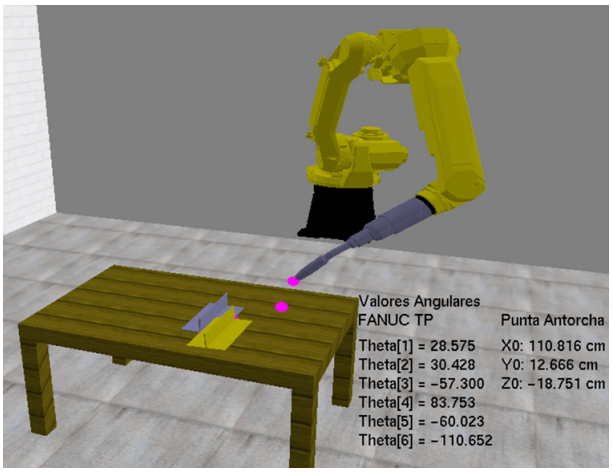


Figura 6.24. Robot virtual en n_4 de la tarea #1.

- **Simulación en la tarea #2.**

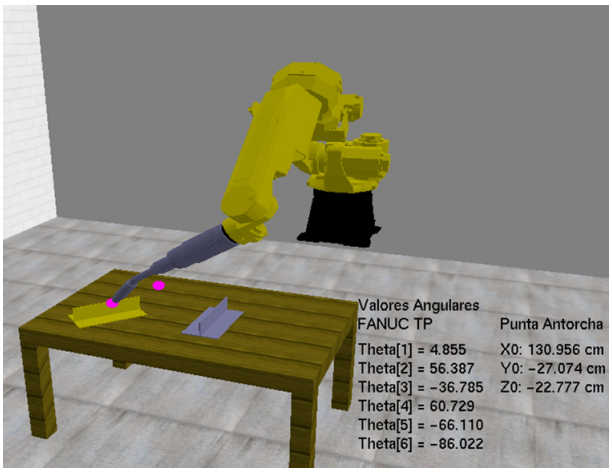


Figura 6.25. Robot virtual en n_1 de la tarea #2.

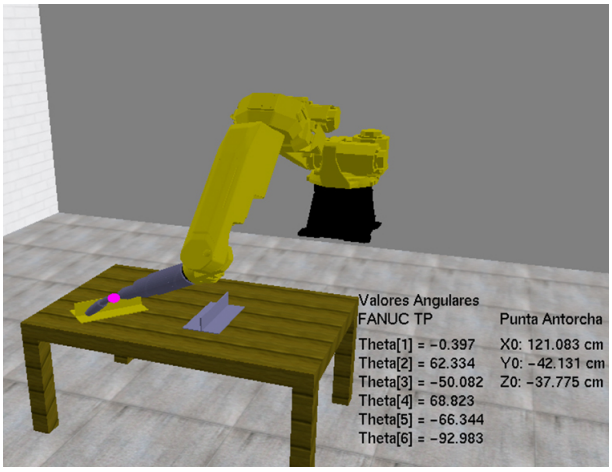


Figura 6.26. Robot virtual en n_2 de la tarea #2.

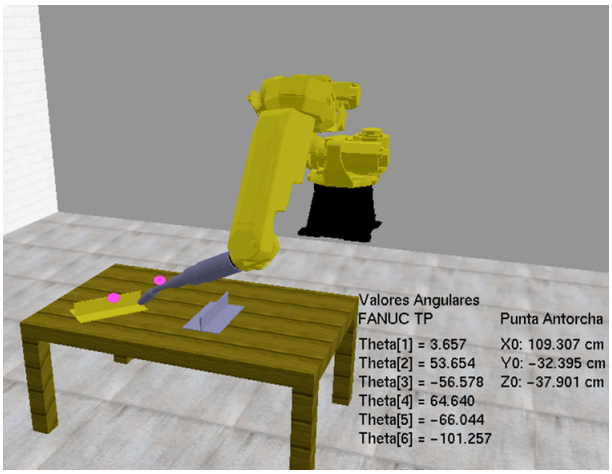


Figura 6.27. Robot virtual en n_3 de la tarea #2.

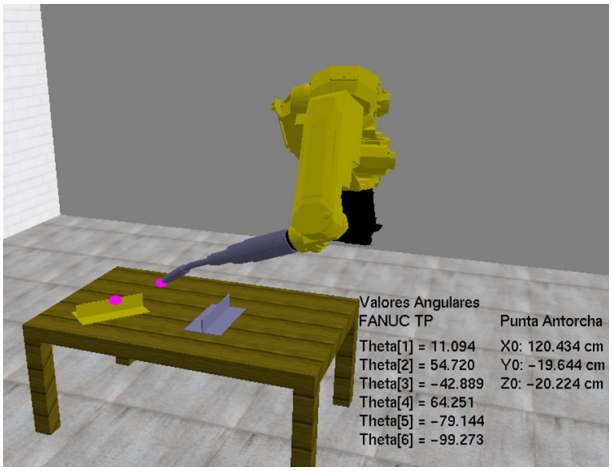


Figura 6.28. Robot virtual en n_4 de la tarea #2.

Cada segmento de la tarea experimental correspondiente se ejecutó en 1 segundo. Gráficamente los historiales de los valores angulares durante las tareas #1 y #2 están representados en las Figuras 6.29 y 6.31 respectivamente, donde están establecidos los límites articulares del robot Fanuc (Tabla 6) con líneas punteadas. Asimismo, en las Figuras 6.30 y 6.32 se presentan las posiciones de la punta de la antorcha de soldadura con respecto a la base del robot (Σ_0) durante las tareas #1 y #2 respectivamente. En todos los historiales se considera que la simulación comienza y finaliza con el robot virtual en la posición de *home*.

Como quedó establecido previamente, el robot virtual en posición de *home* presenta

los valores angulares de la Tabla 10, que se utilizan para calcular los valores angulares para los programas KAREL de acuerdo a la Tabla 26. Además, una vez calculada la matriz ${}^{0}_{6}T$ con estos valores angulares mediante cinemática directa, se puede determinar la posición de la punta de la antorcha de soldadura con respecto a Σ_0 utilizando la Ecuación 6.15 y extrayendo la información posicional de la matriz resultante, de tal forma que se obtienen los valores de $x_0 = 111.11$ cm, $y_0 = 0$ cm y $z_0 = 40.86$ cm.

$$ {}^{0}_{H}T = {}^{0}_{6}T \; {}^{6}_{H}T \tag{6.15} $$

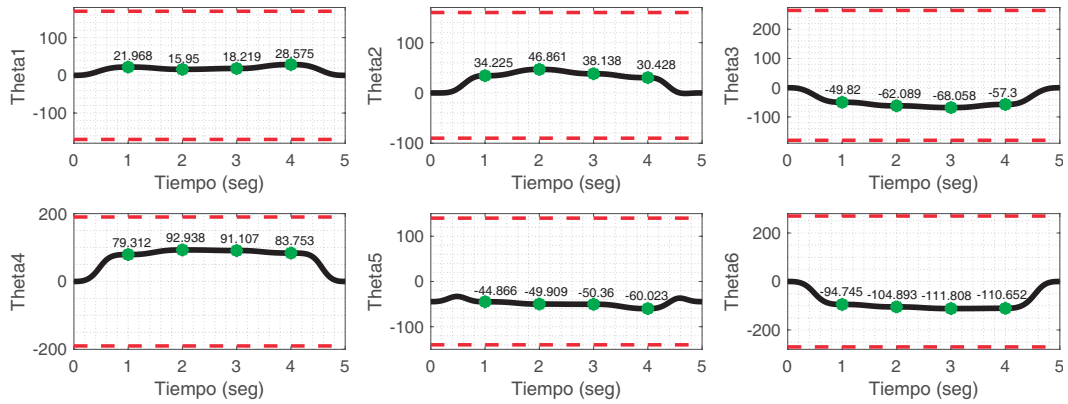


Figura 6.29. Historial de los valores angulares del robot virtual en la tarea #1.

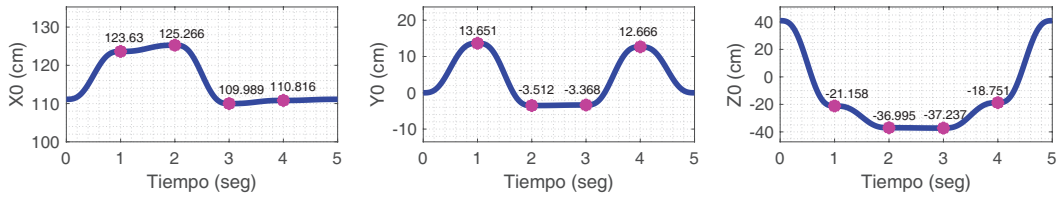


Figura 6.30. Historial de la punta de la antorcha virtual en la tarea #1.

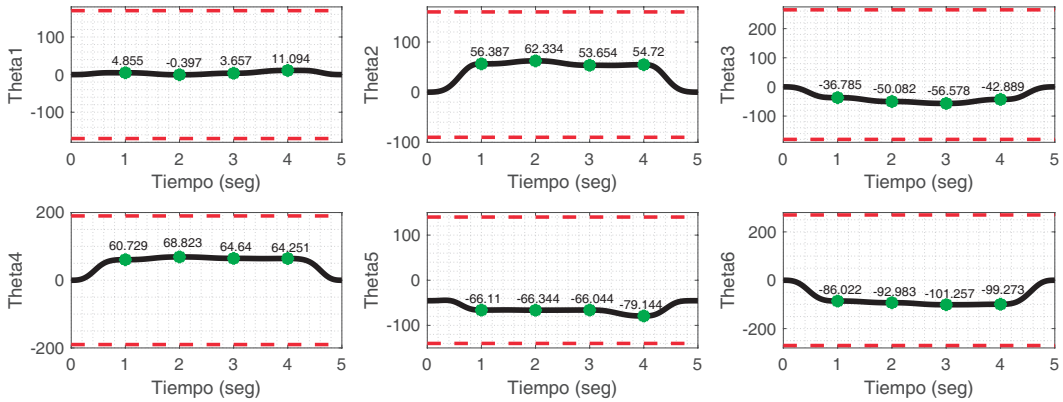


Figura 6.31. Historial de los valores angulares del robot virtual en la tarea #2.

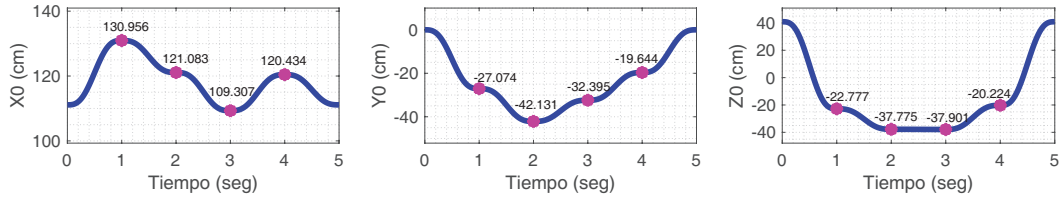


Figura 6.32. Historial de la punta de la antorcha virtual en la tarea #2.

Por otro lado, se efectuó la ejecución por pasos de cada programa experimental, para tomar fotografías del robot Fanuc después de que ejecutara cada instrucción de movimiento y detuviera los movimientos de sus articulaciones. Lo anterior indica que cuando los valores angulares de las articulaciones del robot sean los que se consignaron para cada punto nodo de la trayectoria dentro del programa, el robot suspenderá sus movimientos hasta que se le indique que continúe con la ejecución del programa. Las fotografías se obtuvieron desde una perspectiva similar a la presentada la simulación.

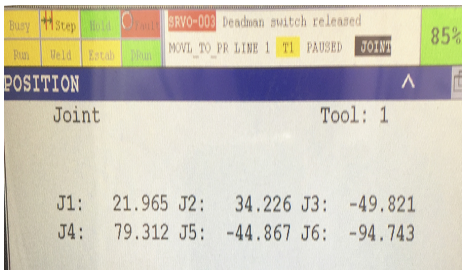
Las detenciones del robot Fanuc también permitieron tomar fotografías de la pantalla del *teach-pendant*; primero a los valores angulares de las articulaciones (modo JOINT) y después a la posición de la antorcha con respecto al sistema de coordenadas del mundo $\mathbf{\Sigma}_o$ (modo WORLD). A cada fotografía del robot Fanuc le corresponden una fotografía del modo JOINT y otra del modo WORLD.

A continuación se presentan las imágenes referentes a los experimentos de la ejecución

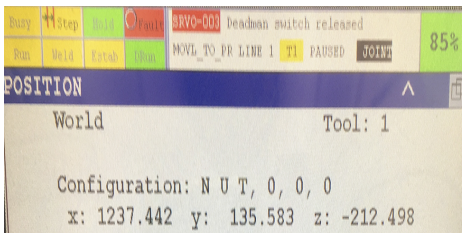
de los programas para las tareas #1 y #2. Cada figura mostrada contiene en su inciso (a) a la fotografía del robot Fanuc en el instante en que su antorcha de soldadura alcanza un punto nodo definido en el programa. En el inciso (b) de estas figuras se presentan los valores angulares en grados de las seis articulaciones del robot Fanuc ($J1$, $J2$, $J3$, $J4$, $J5$, $J6$), tomados de la pantalla del *teach-pendant* (modo JOINT) en el mismo momento de la fotografía del inciso (a). Las figuras también incluyen el inciso (c) que muestra la posición en milímetros de la punta de la antorcha de soldadura del robot Fanuc con respecto al marco del mundo (x, y, z), tomada también de la pantalla del *teach-pendant* (modo WORLD) en dicho momento. Son cuatro figuras por cada tarea experimental. De la Figura 6.33 a la Figura 6.36 corresponden a la tarea #1 y de de la Figura 6.37 a la Figura 6.40 pertenecen a la tarea #2.

- **Robot Fanuc en la tarea #1.**

(a)



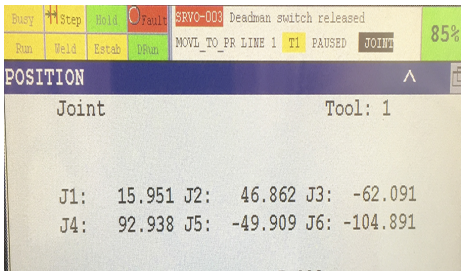
(b)



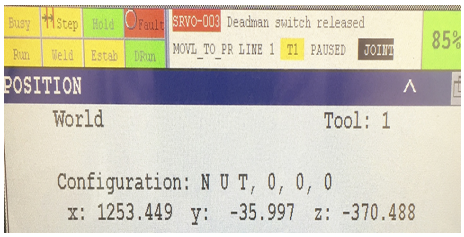
(c)

Figura 6.33. Robot Fanuc en n_1 de la tarea #1.

(a)



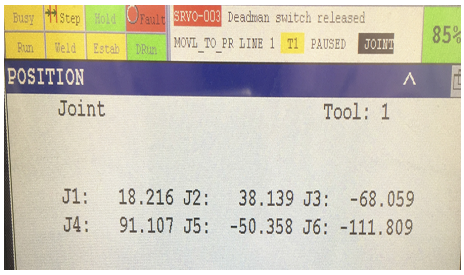
(b)



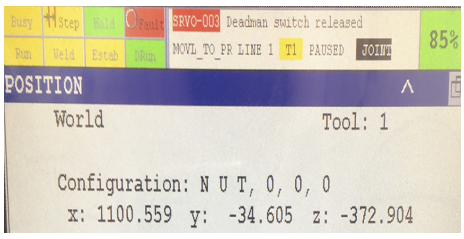
(c)

Figura 6.34. Robot Fanuc en n_2 de la tarea #1.

(a)

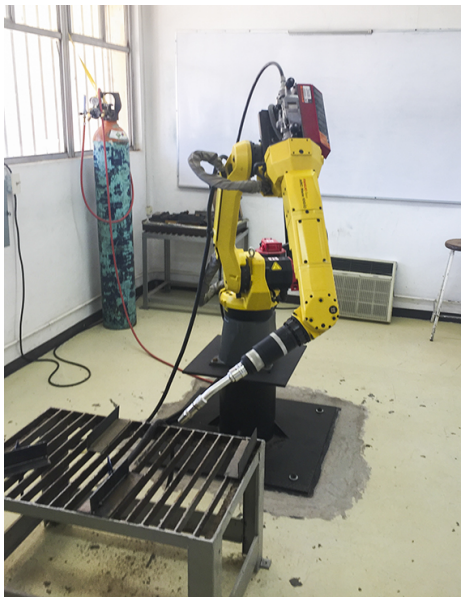


(b)

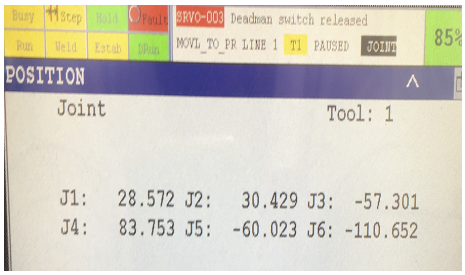


(c)

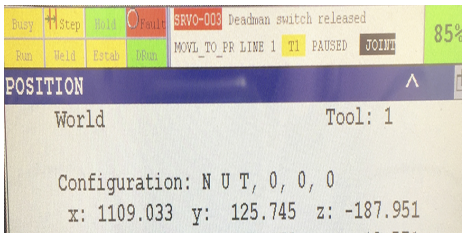
Figura 6.35. Robot Fanuc en n_3 de la tarea #1.



(a)



(b)

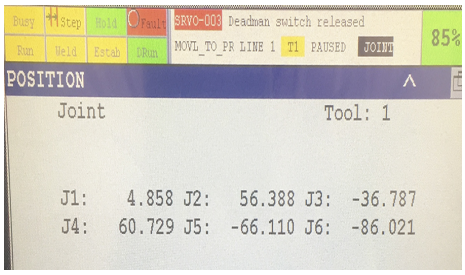


(c)

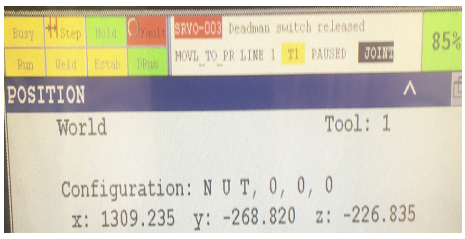
Figura 6.36. Robot Fanuc en n_4 de la tarea #1.

- **Robot Fanuc en la tarea #2.**

(a)



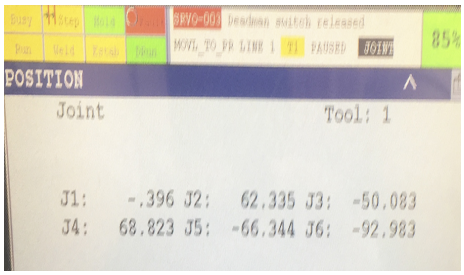
(b)



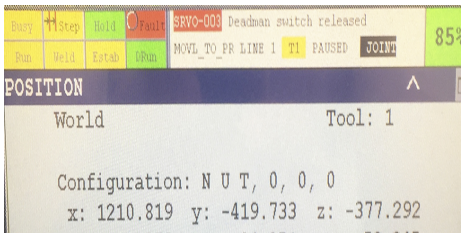
(c)

Figura 6.37. Robot Fanuc en n_1 de la tarea #2.

(a)



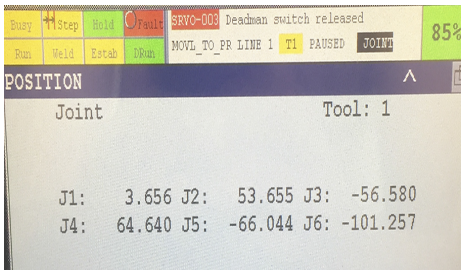
(b)



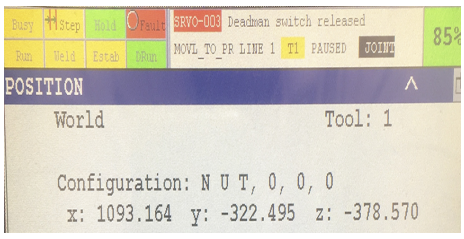
(c)

Figura 6.38. Robot Fanuc en n_2 de la tarea #2.

(a)

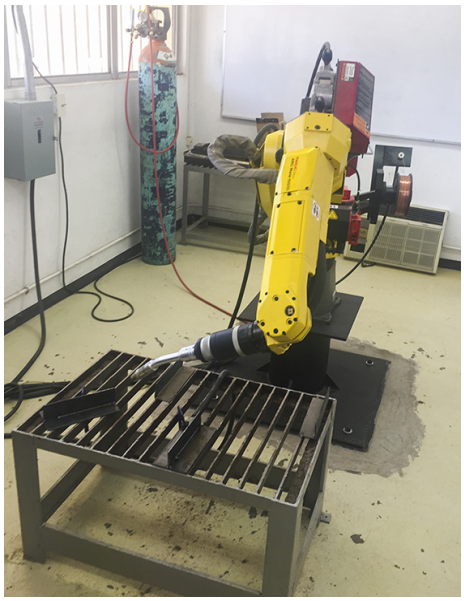


(b)

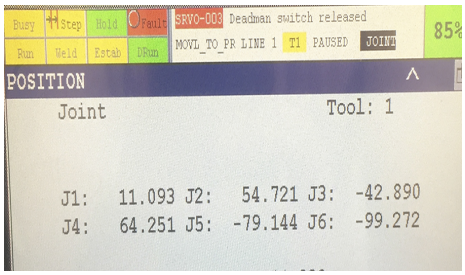


(c)

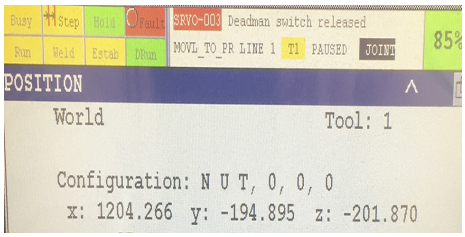
Figura 6.39. Robot Fanuc en n_3 de la tarea #2.



(a)



(b)



(c)

Figura 6.40. Robot Fanuc en n_4 de la tarea #2.

A manera de compendio, de la Tabla 28 a la Tabla 31 se exponen los resultados de los valores angulares. En tales tablas se agrupan los ángulos de las seis articulaciones que se obtuvieron durante la simulación de las dos tareas experimentales, junto con aquellos ángulos presentados en la ejecución de los dos programas experimentales en el robot Fanuc para los cuatro puntos nodo definidos. Asimismo se calcula el error existente entre los valores angulares en la ejecución y los valores angulares en la simulación.

Tabla 28. Resultados de valores angulares en n_1 (grados).

	Tarea #1			Tarea #2		
Articulación	Simulación	Ejecución	Error	Simulación	Ejecución	Error
$J1$	21.968	21.965	-0.003	4.855	4.858	0.003
$J2$	34.225	34.226	0.001	56.387	56.388	0.001
$J3$	-49.820	-49.821	-0.001	-36.785	-36.787	-0.002
$J4$	79.312	79.312	0.000	60.729	60.729	0.000
$J5$	-44.866	-44.867	-0.001	-66.110	-66.110	0.000
$J6$	-94.745	-94.743	0.002	-86.022	-86.021	0.001

Tabla 29. Resultados de valores angulares en n_2 (grados).

	Tarea #1			Tarea #2		
Articulación	Simulación	Ejecución	Error	Simulación	Ejecución	Error
$J1$	15.950	15.951	0.001	-0.397	-0.396	0.001
$J2$	46.861	46.862	0.001	62.334	62.335	0.001
$J3$	-62.089	-62.091	-0.002	-50.082	-50.083	-0.001
$J4$	92.938	92.938	0.000	68.823	68.823	0.000
$J5$	-49.909	-49.909	0.000	-66.344	-66.344	0.000
$J6$	-104.893	-104.891	0.002	-92.983	-92.983	0.000

Tabla 30. Resultados de valores angulares en n_3 (grados).

	Tarea #1			Tarea #2		
Articulación	Simulación	Ejecución	Error	Simulación	Ejecución	Error
$J1$	18.219	18.216	-0.003	3.657	3.656	-0.001
$J2$	38.138	48.139	0.001	53.654	53.655	0.001
$J3$	-68.058	-68.059	-0.001	-56.578	-56.580	-0.002
$J4$	91.107	91.107	0.000	64.640	64.640	0.000
$J5$	-50.360	-50.358	0.002	-66.044	-66.044	0.000
$J6$	-111.808	-111.809	-0.001	-101.257	-101.257	0.000

Tabla 31. Resultados de valores angulares en n_4 (grados).

	Tarea #1			Tarea #2		
Articulación	Simulación	Ejecución	Error	Simulación	Ejecución	Error
$J1$	28.575	28.572	-0.003	11.094	11.093	-0.001
$J2$	30.428	30.429	0.001	54.720	54.721	0.001
$J3$	-57.300	-57.301	-0.001	-42.889	-42.890	-0.001
$J4$	83.753	83.753	0.000	64.251	64.251	0.000
$J5$	-60.023	-60.023	0.000	-79.144	-79.144	0.000
$J6$	-110.652	-110.652	0.000	-99.273	-99.272	0.001

En el mismo sentido, de la Tabla 32 a la Tabla 35 se han integrado los valores de posición de la punta de la antorcha con respecto al marco de la base del robot durante

la simulación junto con los presentados por el robot Fanuc al ejecutar los programas. También, se ha calculado el error que hay entre los valores de posición en la ejecución y los valores de posición en la simulación.

Tabla 32. Resultados de valores de posición en n_1 (cm).

	Tarea #1			Tarea #2		
Coordenada	Simulación	Ejecución	Error	Simulación	Ejecución	Error
x	123.630	123.744	0.114	130.956	130.924	-0.032
y	13.651	13.558	-0.093	-27.074	-26.882	0.192
z	-21.158	-21.250	-0.092	-22.777	-22.684	0.093

Tabla 33. Resultados de valores de posición en n_2 (cm).

	Tarea #1			Tarea #2		
Coordenada	Simulación	Ejecución	Error	Simulación	Ejecución	Error
x	125.266	125.345	0.079	121.083	121.082	-0.001
y	-3.512	-3.600	-0.088	-42.131	-41.973	0.158
z	-36.995	-37.049	-0.054	-37.775	-37.729	0.046

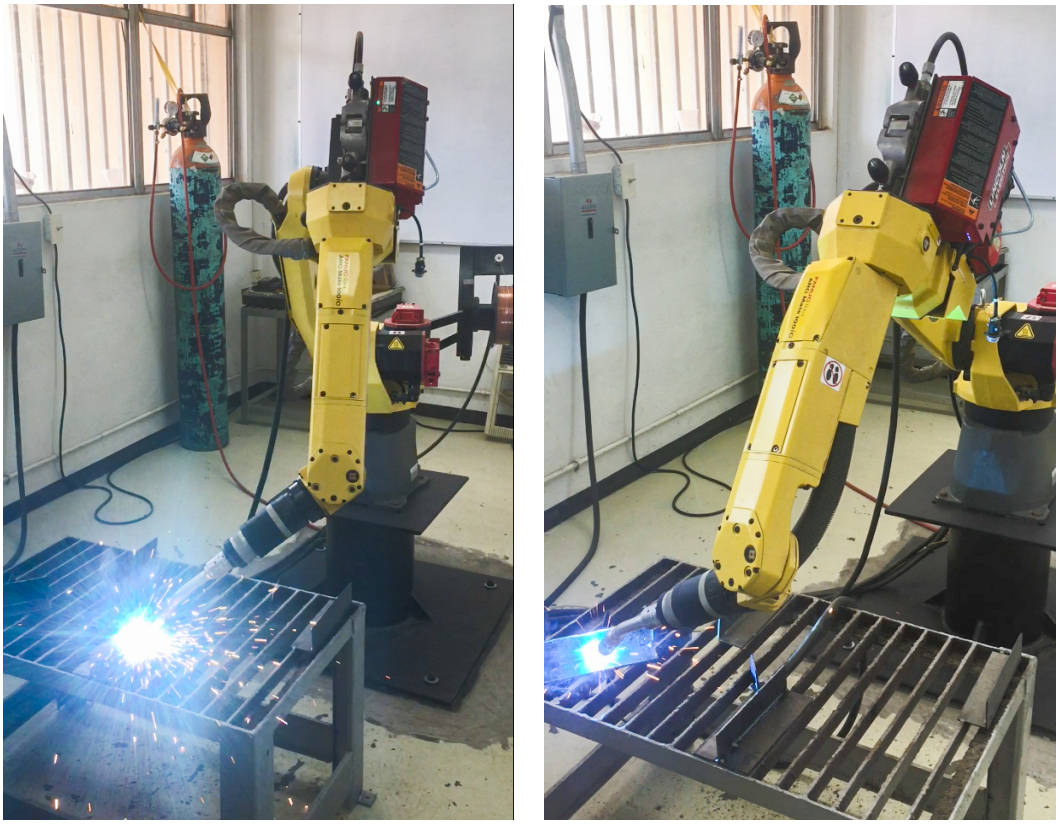
Tabla 34. Resultados de valores de posición en n_3 (cm).

	Tarea #1			Tarea #2		
Coordenada	Simulación	Ejecución	Error	Simulación	Ejecución	Error
x	109.989	110.056	0.067	109.307	109.316	0.009
y	-3.368	-3.461	-0.093	-32.395	-32.250	0.145
z	-37.237	-37.290	-0.053	-37.901	-37.857	0.044

Tabla 35. Resultados de valores de posición en n_4 (cm).

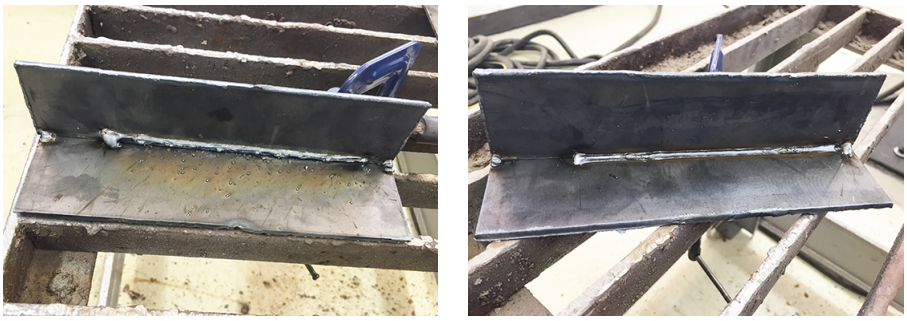
	Tarea #1			Tarea #2		
Coordenada	Simulación	Ejecución	Error	Simulación	Ejecución	Error
x	110.816	110.903	0.087	120.434	120.427	-0.007
y	12.666	12.575	-0.091	-19.644	-19.490	0.154
z	-18.751	-18.795	-0.044	-20.224	-20.187	0.037

Finalmente, el robot Fanuc efectuó los cordones de soldadura en las dos placas experimentales de acuerdo a las tareas programadas (Figura 6.41). En la Figuras 6.42 se observan dichos cordones aplicados en la junta de ambas placas.



(a) **(b)**

Figura 6.41. Robot aplicando los cordones de soldadura ((a) en las placas #1 y (b) en las placas #2).



(a) **(b)**

Figura 6.42. Cordones de soldadura aplicados ((a) en las placas #1 y (b) en las placas #2).

Capítulo 7

Análisis de resultados

Los resultados en acoplamiento virtual entre la AVC y la AVD muestran que la configuración de SRA donde se utilizan cuatro SRAL produjo un mejor rendimiento en los experimentos (Figura 5.10). Esta configuración presentó una menor diferencia de distancias entre ambas antorchas en los experimentos donde los usuarios manipularon la AVC con la interfaz háptica. De igual forma, en estos experimentos, las diferencias angulares entre las antorchas acopladas fueron las menores.

Por otro lado, con la configuración de SRA mencionada, se comprobó que cuando los usuarios utilizaron la interfaz háptica para realizar los movimientos de la AVC, se pudieron conseguir distancias menores a 0.7 cm como máximas distancias entre la AVC y la AVD (MDDP y MDDCM). Asimismo, se validó que en los promedios de las diferencias de las distancias entre ambas antorchas se pudieron alcanzar valores menores a 0.3 cm (PDDP y PDDCM). Además, se logró una diferencia máxima angular entre ellas de un grado (MDA), mientras que fue posible conseguir una diferencia angular promedio menor a 0.4 grados (DAP) (Tabla 16).

En el caso de la precisión de la antorcha virtual para colocar los puntos de soldadura, se tiene que de las cuatro condiciones empleadas en los experimentos de este tipo, la que arrojó mejores resultados fue la condición $F + C$ en los tres. Esta condición presenta sensación háptica y se maneja con un comportamiento dinámico en el

ambiente virtual.

En primer término, cuando a cada usuario se le pidió colocar un punto de soldadura en la placa horizontal (P_1 en la Figura 5.14), con la condición $F + C$ aplicada en la antorcha virtual, el 84.62% de los puntos de soldadura fueron puestos en la placa en sí. Mientras que el 15.38% de los usuarios registraron su punto de soldadura "en el aire" (flotantes); y dado que el comportamiento dinámico restringe el traslape de la antorcha virtual con la placa horizontal, la condición $F + C$ no permitió que los usuarios colocaran sus puntos de soldadura "en el interior" (adentro) de la misma (Tabla 20).

Algo similar al procedimiento experimental con la placa horizontal ocurrió cuando el usuario tenía que ubicar un punto de soldadura en la placa vertical (P_2 en la Figura 5.14). En este ocasión, con la condición $F + C$ en la antorcha virtual, los usuarios pudieron registrar sus puntos de soldadura en dicha placa el 92.31% de las veces y únicamente el 7.69% fueron puntos flotantes, dejando en 0% los puntos adentro de la placa (Tabla 21).

La condición $F + C$ también presentó los mejores resultados cuando se aplicó a la antorcha virtual y se efectuó el experimento donde cada usuario debía registrar un punto de soldadura en la junta de ambas placas (P_3 en la Figura 5.14). Los resultados para tal experimento mostraron que el 46.15% de los puntos de soldadura ubicados por los usuarios fueron en la junta de las placas. A diferencia de los anteriores procedimientos experimentales, en este caso hubo más puntos flotantes registrado, los cuales alcanzaron un 53.85%, sin registrarse ningún caso de puntos adentro de las placas (Tabla 22).

En la evaluación subjetiva que los usuarios hicieron sobre su experiencia al manipular la antorcha virtual con la interfaz háptica durante los experimentos, éstos consideraron que el comportamiento dinámico proporciona un mejor realismo durante la interacción al evaluarlo con un promedio de 4.46. Por su parte, juzgaron también que el comportamiento dinámico favorece la precisión de la ejecución de

las tareas, calificándolo con un promedio de 4.08. Sin embargo, cuando se les preguntó sobre la facilidad con que se ejecutan las tareas, tanto el retorno de fuerzas como el comportamiento dinámico obtuvieron la misma calificación promedio de 4.08 (Tabla 23). Por otra parte, el 76.92% de los usuarios calificó a $F + C$ como la mejor condición al percibirla como la que ofrece una mejor experiencia durante los experimentos (Figura 5.30).

Para la validación de la programación fuera de línea del robot de soldadura, se requirió hacer una comparación entre los valores angulares obtenidos en la simulación y los conseguidos durante la ejecución de las tareas programadas, la cual nos dio como resultado errores entre ±0.03 grados en los cuatro puntos nodo. Pero al contrastar los resultados entre los valores de posición de ambos escenarios para las dos tareas, se obtuvieron errores entre −0.093 cm y 0.192 cm, las cuales implican desviaciones entre lo que se reprodujo en el mundo real y lo que se efectuó en el ambiente virtual.

Se observa que las diferencias entre los valores angulares de la simulación y la ejecución son insignificantes. Esto se debe a que los valores angulares calculados en la simulación son los que se envían al programa KL. Estos valores son los que el robot Fanuc adopta sin hacer ningún otro cálculo, por lo que la correspondencia entre ellos es evidente.

En cambio, las diferencias entre los valores de posición son considerables. El motivo de tales diferencias radica en que la definición de matrices de transformación se debe llevar a cabo a base de vectores unitarios ortogonales (perpendiculares), de tal modo que bajo este escenario las operaciones con dicha matrices ofrecen resultados exactos. En caso contrario, si las matrices de transformación se obtienen a partir de sistemas de coordenadas con ejes no ortogonales, los resultados de las operaciones propias son inexactos. Tal fue el caso que ocurrió al definir los vectores unitarios en el proceso de calibración para las dos placas experimentales.

Se comprobó que los marcos de referencia calibrados para las dos placas experimentales no son ortogonales. Para las placas #1 se obtuvieron los vectores

unitarios ${}^0\hat{x}_{PR1}$ y ${}^0\hat{y}_{PR1}$ (Ecuación 6.9). El producto escalar entre ellos es igual a 0.01. Ahora bien, para las placas #2 se generaron los vectores unitarios ${}^0\hat{x}_{PR2}$ y ${}^0\hat{y}_{PR2}$ (Ecuación 6.10) y su producto escalar es igual -0.012. Dado que para que dos vectores sean ortogonales, el producto escalar entre ellos debe ser igual a 0, el proceso de calibración en ninguna de las dos placas experimentales brindó resultados exactos.

Aun cuando se presentaron los resultados de los cordones de soldadura aplicados a las placas experimentales, el análisis de éstos no es parte de este trabajo porque el presente trabajo está enfocado en la aplicación de la realidad virtual para generar programas fuera de línea para el robot de soldadura utilizando parámetros de soldadura preestablecidos.

Capítulo 8

Conclusiones y trabajo futuro

Se ha presentado un sistema para la programación fuera de línea de robots de soldadura donde el usuario tiene que manipular una antorcha de soldadura utilizando una interfaz háptica para definir tareas de soldadura dentro de un ambiente virtual.

En este sistema se utilizó una técnica en la que la antorcha virtual está representada por dos modelos acoplados con sistemas resorte-amortiguador. La diferencia de posiciones de los dos modelos de antorchas permite calcular la fuerza que se envía a la interfaz háptica de tal manera que la mano del usuario siente en tiempo real el impacto de la antorcha virtual. Por lo tanto, una rectificación del movimiento del dispositivo debería guiar al usuario hasta la ubicación correcta del órgano terminal. Gracias a este enfoque, la sensación de fuerza para el usuario facilita la especificación del movimiento deseado de la antorcha al permitir una colocación más precisa de los puntos de soldadura, asegurando que se coloquen en las placas para soldar y no "en el aire" o dentro de ellas.

Los sistemas resorte-amortiguador lineales provocaron que la antorcha virtual manifestara un comportamiento más estable durante su manipulación mediante la interfaz háptica que cuando se emplearon sistemas resorte-amortiguador torsionales. Sin embargo, se requiere que los usuarios se adiestren por más tiempo en el manejo de la interfaz háptica, puesto que el uso de sistemas resorte-amortiguador provoca

algunas vibraciones en el dispositivo que ocasionan cierta cautela en los usuarios, la cual impacta negativamente en el acoplamiento virtual de los dos modelos de antorcha. Paulatinamente los usuarios se van acostumbrando a estas oscilaciones y se obtienen mejores resultados en el acoplamiento virtual.

La antorcha virtual con comportamiento dinámico y retroalimentación háptica (condición $F + C$) permitió una colocación más precisa de los puntos de soldadura que aquellas antorchas sin estas dos características aplicadas a la vez. Asimismo, la antorcha con la condición $F + C$ fue la mejor evaluada por los usuarios en cuanto a su precisión en la definición de los puntos de soldadura y a la facilidad con que se ejecutaron las tareas experimentales.

La metodología de calibración propuesta es propensa a errores de medición y esto ocasiona diferencias entre lo que se representa en el ambiente virtual y lo que se ejecuta el mundo real. Estas diferencias son las que determinan si se coloca el cordón de soldadura en las placas o si la antorcha de soldadura choca con las mismas ocasionando daños en el robot de soldadura.

Quedaron varios puntos en el tintero, que por falta de tiempo no se pudieron conseguir, quedando éstos como trabajo futuro:

- Proponer un método de calibración que brinde una mayor exactitud.
- Experimentar con el acoplamiento virtual basado en control automático.
- Incluir funciones para definir tareas de soldadura con trayectorias circulares y otras formas geométricas.
- Generar los archivos *p-code* (.PC) sin utilizar la utilería que brinda Fanuc.
- Interconectar la PC con el controlador del robot para transmitirle los programas directamente.
- Evaluar el desempeño cinemático del robot al ejecutar tareas.

Anexos

A Hoja de evaluación de la experiencia del usuario

Descripción del experimento.

El sujeto (usuario) tiene que manipular la antorcha virtual utilizando la interfaz háptica con el fin de colocar en primer término un punto de soldadura en una placa horizontal, posteriormente otro punto de soldadura en una placa vertical y finalmente un punto de soldadura en la intersección de ambas placas. Tal procedimiento se efectúa bajo cuatro condiciones de la antorcha virtual. Antes de cada sesión experimental, el sujeto realiza un entrenamiento para familiarizarse con el manejo de la interfaz háptica.

Condiciones del experimento.

Se han contemplado dos condiciones generales que rigen la interacción de la antorcha dentro del ambiente virtual. La primera es que la fuerza de contacto entre la antorcha virtual y los objetos se envíe a la mano del sujeto a través de la interfaz háptica. La otra condición se basa en el uso de comportamiento dinámico, el cual contribuye a que no exista interpenetración visual entre los objetos del ambiente virtual.

Después de una sesión de entrenamiento, el sujeto realiza la colocación de los puntos de soldadura bajo las siguientes condiciones experimentales específicas en la antorcha virtual:

X	Sin retorno de fuerzas y sin comportamiento dinámico
F	Con retorno de fuerzas y sin comportamiento dinámico
C	Sin retorno de fuerzas y con comportamiento dinámico
$F + C$	Con retorno de fuerzas y con comportamiento dinámico

Procedimiento

Las cuatro condiciones experimentales se presentan a cada sujeto en un orden aleatorio. Dado que es una tarea sencilla cada sujeto ensaya la tarea durante cinco minutos y la hace válida sólo una vez. Después de los experimentos, el sujeto contesta el siguiente cuestionario indicando un nivel de importancia para cada criterio: los niveles de importancia varían de 0 a 5 (0 = nivel insignificante 5 = nivel muy importante que corresponde a una contribución muy significativa del criterio en cuestión). El sujeto también puede agregar información adicional en la sección de comentarios reservada para cada pregunta formulada.

Pregunta N° 1

¿Cuál es la contribución de la sensación de contacto entre la antorcha con las placas en

la percepción de realismo durante la interacción? 0 1 2 3 4 5

la precisión de la ejecución de la tarea? 0 1 2 3 4 5

la dificultad/facilidad de la tarea? 0 1 2 3 4 5

Comentarios: _____

_____

Pregunta N° 2

¿Cuál es el aporte del comportamiento dinámico en el ambiente virtual en

la percepción de realismo durante la interacción? 0 1 2 3 4 5

la precisión de la ejecución de la tarea? 0 1 2 3 4 5

la dificultad/facilidad de la tarea? 0 1 2 3 4 5

Comentarios: _____

_____

Pregunta N° 3

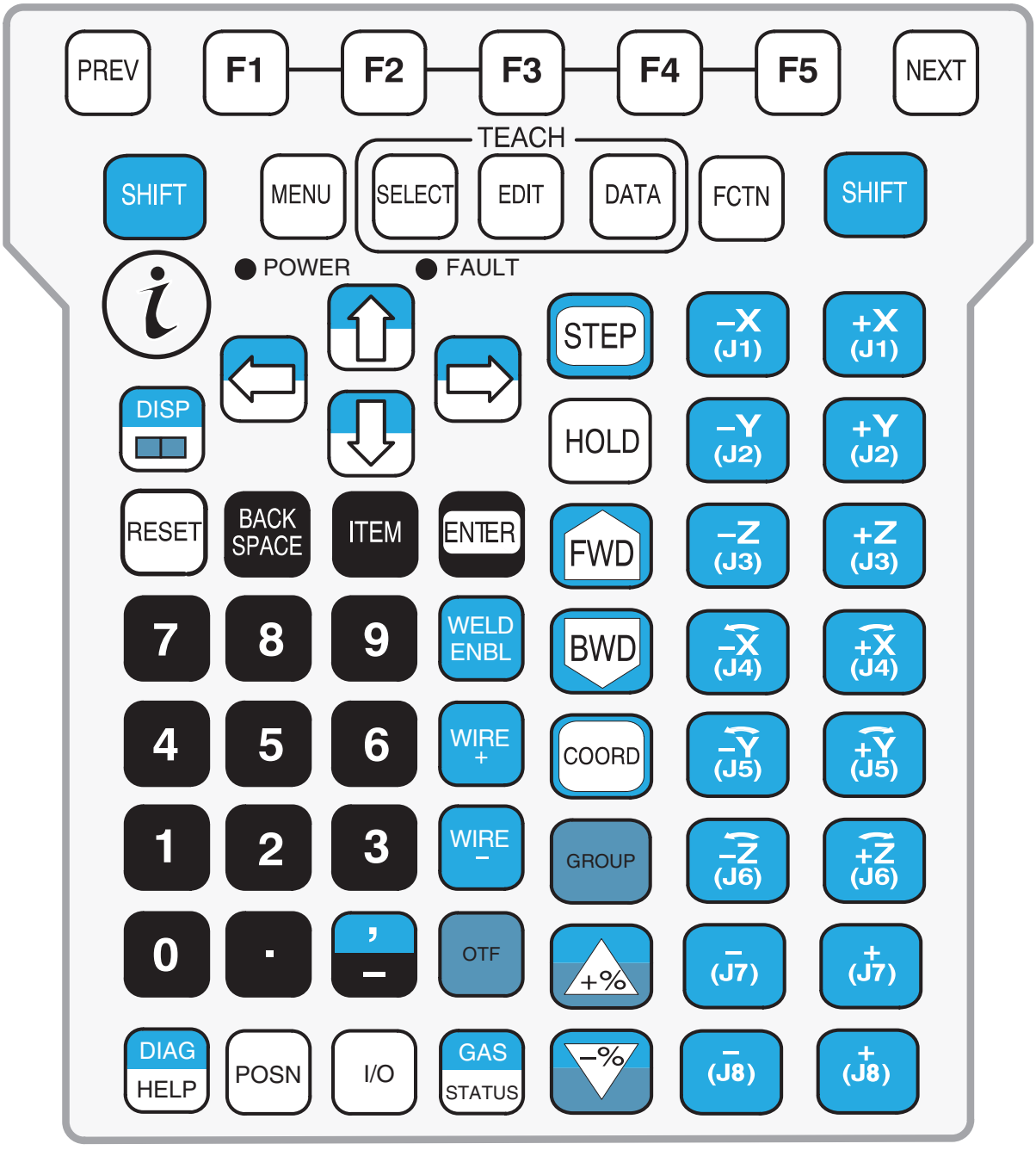
En términos de ejecución de la tarea, clasifica las condiciones experimentales X, C, F y $F + C$ de la más fácil a la menos fácil.

_____

Comentarios adicionales: _____

_____

B Botones del *teach-pendant*



C Programa en KAREL para las placas #1 (1/2)

```
PROGRAM offline25_1w

CONST
  NUM_AXES = 6
VAR
  p_joints : JOINTPOS6
  p_joints_a : ARRAY[NUM_AXES] OF REAL
  status: INTEGER

ROUTINE movl_to_pr FROM movl_to_pr
ROUTINE m_weldstart FROM m_weldstart
ROUTINE m_weldend FROM m_weldend

BEGIN
  p_joints_a[1] = 21.968
  p_joints_a[2] = 34.225
  p_joints_a[3] = -49.820
  p_joints_a[4] = 79.312
  p_joints_a[5] = -44.866
  p_joints_a[6] = -94.745

  CNV_REL_JPOS(p_joints_a, p_joints, status)
  SET_JPOS_REG(10, p_joints, status)
  movl_to_pr

  p_joints_a[1] = 15.950
  p_joints_a[2] = 46.861
  p_joints_a[3] = -62.089
  p_joints_a[4] = 92.938
  p_joints_a[5] = -49.909
  p_joints_a[6] = -104.893

  CNV_REL_JPOS(p_joints_a, p_joints, status)
  SET_JPOS_REG(11, p_joints, status)
  m_weldstart
```

Programa en KAREL para las placas #1 (2/2)

```
  p_joints_a[1] = 18.219
  p_joints_a[2] = 38.138
  p_joints_a[3] = -68.058
  p_joints_a[4] = 91.107
  p_joints_a[5] = -50.360
  p_joints_a[6] = -111.808

  CNV_REL_JPOS(p_joints_a, p_joints, status)
  SET_JPOS_REG(12, p_joints, status)
  m_weldend

  p_joints_a[1] = 28.575
  p_joints_a[2] = 30.428
  p_joints_a[3] = -57.300
  p_joints_a[4] = 83.753
  p_joints_a[5] = -60.023
  p_joints_a[6] = -110.652

  CNV_REL_JPOS(p_joints_a, p_joints, status)
  SET_JPOS_REG(10, p_joints, status)
  movl_to_pr

  p_joints_a[1] = 0.000
  p_joints_a[2] = 0.000
  p_joints_a[3] = 0.000
  p_joints_a[4] = 0.000
  p_joints_a[5] = -45.000
  p_joints_a[6] = 0.000

  CNV_REL_JPOS(p_joints_a, p_joints, status)
  SET_JPOS_REG(10, p_joints, status)
  movl_to_pr

END offline25_1w
```

Programa en KAREL para las placas #2 (1/2)

```
PROGRAM offline25_2w

CONST
  NUM_AXES = 6
VAR
  p_joints : JOINTPOS6
  p_joints_a : ARRAY[NUM_AXES] OF REAL
  status: INTEGER

ROUTINE movl_to_pr FROM movl_to_pr
ROUTINE m_weldstart FROM m_weldstart
ROUTINE m_weldend FROM m_weldend

BEGIN
  p_joints_a[1] = 4.855
  p_joints_a[2] = 56.387
  p_joints_a[3] = -36.785
  p_joints_a[4] = 60.729
  p_joints_a[5] = -66.110
  p_joints_a[6] = -86.022

  CNV_REL_JPOS(p_joints_a, p_joints, status)
  SET_JPOS_REG(10, p_joints, status)
  movl_to_pr

  p_joints_a[1] = -0.397
  p_joints_a[2] = 62.334
  p_joints_a[3] = -50.082
  p_joints_a[4] = 68.823
  p_joints_a[5] = -66.344
  p_joints_a[6] = -92.983

  CNV_REL_JPOS(p_joints_a, p_joints, status)
  SET_JPOS_REG(11, p_joints, status)
  m_weldstart
```

Programa en KAREL para las placas #2 (2/2)

```
  p_joints_a[1] = 3.657
  p_joints_a[2] = 53.654
  p_joints_a[3] = -56.578
  p_joints_a[4] = 64.640
  p_joints_a[5] = -66.044
  p_joints_a[6] = -101.257

  CNV_REL_JPOS(p_joints_a, p_joints, status)
  SET_JPOS_REG(12, p_joints, status)
  m_weldend

  p_joints_a[1] = 11.094
  p_joints_a[2] = 54.720
  p_joints_a[3] = -42.889
  p_joints_a[4] = 64.251
  p_joints_a[5] = -79.144
  p_joints_a[6] = -99.273

  CNV_REL_JPOS(p_joints_a, p_joints, status)
  SET_JPOS_REG(10, p_joints, status)
  movl_to_pr

  p_joints_a[1] = 0.000
  p_joints_a[2] = 0.000
  p_joints_a[3] = 0.000
  p_joints_a[4] = 0.000
  p_joints_a[5] = -45.000
  p_joints_a[6] = 0.000

  CNV_REL_JPOS(p_joints_a, p_joints, status)
  SET_JPOS_REG(10, p_joints, status)
  movl_to_pr

END offline25_2w
```

Bibliografía

[1] International Federation of Robotics, "2018 Industrial Robot Statistics http://www.ifr.org/industrial-robots/statistics/," 2018.

[2] I. Karabegović and R. Mirza, "Automation of the welding process by use of industrial robots," in *International Conference New Technologies, Development and Applications*, pp. 3–17, Springer, 2018.

[3] N. Larkin, A. Short, Z. Pan, and S. van Duin, "Automated programming for robotic welding," in *Transactions on Intelligent Welding Manufacturing*, pp. 48–59, Springer, 2018.

[4] J.-A. Lozano-Quilis, H. Gil-Gomez, J.-A. Gil-Gómez, S. Albiol-Perez, G. Palacios, H. M. Fardoum, and A. S. Mashat, "Virtual reality system for multiple sclerosis rehabilitation using KINECT," in *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2013 7th International Conference on*, pp. 366–369, IEEE, 2013.

[5] P. Almajano, M. L. Sanchez, I. Rodriguez, and E. Mayas, "Including Conversational Agents into Structured Hybrid 3D Virtual Environments," *IEEE Latin America Transactions*, vol. 13, no. 2, pp. 523–531, 2015.

[6] Y. Gan, X. Dai, and D. Li, "Off-line programming techniques for multirobot cooperation system," *International Journal of Advanced Robotic Systems*, vol. 10, no. 7, p. 282, 2013.

[7] W. Dong, H. Li, and X. Teng, "Off-line programming of Spot-weld Robot for Car-body in White Based on Robcad," in *Mechatronics and Automation, 2007. ICMA 2007. International Conference on*, pp. 763–768, IEEE, 2007.

[8] Z. M. Bzymek, M. Nunez, M. Li, and S. Powers, "Simulation of a machining sequence using delmia/quest software," *Computer-Aided Design and Applications*, vol. 5, no. 1-4, pp. 401–411, 2008.

[9] J. Breat, F. Clement, P. Jadeau, A. Ijel, and R. Macia, "ACT WELD: a unique off-line programming software tailored for robotic welding applications," in *1994 International Conference of International Institute of Welding*, Souder, 1994.

[10] A. Robotics, "Operating Manual RobotStudio," *Västerås, Sweden*, 2007.

[11] K. Vollmann, "A new approach to robot simulation tools with parametric components," in *Industrial Technology, 2002. IEEE ICIT'02. 2002 IEEE International Conference on*, vol. 2, pp. 881–885, IEEE, 2002.

[12] L. Li, X. Li, X. Zhou, and J. Yue, "Study of Off-Line Programming System of Arc Robot Based on the Software of ROBOGUIDE," *Robotic Welding, Intelligence and Automation*, pp. 401–408, 2007.

[13] S. Derby, "Simulating motion elements of general-purpose robot arms," *The International journal of robotics research*, vol. 2, no. 1, pp. 3–12, 1983.

[14] D. Lee and W. ElMaraghy, "ROBOSIM: a CAD-based off-line programming and analysis system for robotic manipulators," *Computer-aided engineering journal*, vol. 7, no. 5, pp. 141–148, 1990.

[15] S. Zeghloul, B. Blanchard, and M. Ayrault, "SMAR: A robot modeling and simulation system," *Robotica*, vol. 15, no. 01, pp. 63–73, 1997.

[16] H. Wu, H. Deng, C. Yang, Y. Guan, H. Zhang, and H. Li, "A robotic off-line programming system based on SolidWorks," in *Robotics and Biomimetics (ROBIO), 2015 IEEE International Conference on*, pp. 1711–1716, IEEE, 2015.

[17] P. Neto, J. N. Pires, and A. P. Moreira, "CAD-based off-line robot programming," in *Robotics Automation and Mechatronics (RAM), 2010 IEEE Conference on*, pp. 516–521, IEEE, 2010.

[18] P. Neto and N. Mendes, "Direct off-line robot programming via a common CAD package," *Robotics and Autonomous Systems*, vol. 61, no. 8, pp. 896–910, 2013.

[19] Z. Yin, Y. Guan, S. Chen, W. Wu, and H. Zhang, "Off-line programming of robotic system based on DXF files of 3D models," in *Information and Automation (ICIA), 2013 IEEE International Conference on*, pp. 1296–1301, IEEE, 2013.

[20] V. Bottazzi and J. Fonseca, "Off-line programming industrial robots based in the information extracted from neutral files generated by the commercial CAD tools," in *Industrial Robotics: Programming, Simulation and Applications*, InTech, 2006.

[21] S. Deng, Z. Cai, D. Fang, H. Liao, and G. Montavon, "Application of robot offline programming in thermal spraying," *Surface and Coatings Technology*, vol. 206, no. 19, pp. 3875–3882, 2012.

[22] G. Erdős, Z. Kemény, A. Kovács, and J. Váncza, "Planning of remote laser welding processes," *Procedia CIRP*, vol. 7, pp. 222–227, 2013.

[23] J. Świder, K. Foit, G. Wszołek, and D. Mastrowski, "The system for simulation and offline, remote programming of the Mitsubishi Movemaster RV-M1 robot," *Journal of Achievements in Materials and Manufacturing Engineering*, vol. 25, no. 1, pp. 7–14, 2007.

[24] J. Cai, Z. Ding, Y. Zhang, and M. Liu, "Trajectory planning and simulation for intersecting line cutting of the industry robot," in *Intelligent Control and Automation (WCICA), 2014 11th World Congress on*, pp. 63–68, IEEE, 2014.

[25] L. A. Ferreira, Y. L. Figueira, I. F. Iglesias, and M. Á. Souto, "Offline CAD-based Robot Programming and Welding Parametrization of a Flexible and Adaptive Robotic Cell Using Enriched CAD/CAM System for Shipbuilding," *Procedia Manufacturing*, vol. 11, pp. 215–223, 2017.

[26] J. A. Pámanes, R. A. Fematt, G. Franco, E. Juárez, and E. Guajardo, "Mejoras a los subsistemas mecánico y de comunicación del robot experimental SALVIATI," *Memorias del I Congreso de la Sociedad Mexicana de Ingeniería Mecánica*, pp. 157–162, 1995.

[27] R. A. Fematt, E. Juárez, J. Pámanes, and G. Franco, "Desarrollo de un Sistema de Simulación y Programación para el Robot Experimental SALVIATI," *Memorias del I Congreso Nacional de Robótica*, pp. 95–100, 1996.

[28] J. A. Pámanes, A. Dzul, and E. Marín, "Programación fuera de Línea de Tareas de Soldadura en un Robot Manipulador Experimental," *Memorias del III Congreso Iberoamericano de Ingeniería Mecánica*, 1997.

[29] J. Pámanes, L. Domínguez, and A. Barrón, "A Task Planning System for an Experimental Robotic Manipulator," *Memorias del I Congreso Mexicano de Robótica de la AMRob*, pp. 17–24, 1999.

[30] J. A. Pámanes, A. Barrón, and C. Pinedo, "Constrained optimization in redundancy resolution of robotic manipulators," in *10th World Congress on the Theory of Machines and Mechanisms*, pp. 1057–1066, 1998.

[31] O. Alba and J. Pámanes, "Actualización y Mejoramiento del Desempeño del Paquete PLASIRS para la Simulación Gráfica de un Robot Manipulador Experimental," *Memorias del V Congreso Mexicano de Robótica de la AMRob*, pp. 72–77, 2003.

[32] J. Pamanes, E. Cuan-Durón, and S. Zeghloul, "Single and multi-objective optimization of path placement for redundant robotic manipulators," *INGENIERÍA Investigación y Tecnología*, vol. 9, no. 3, pp. 231–257, 2008.

[33] W. Khalil, "SYMORO: système pour la modélisation des robots," *Support technique-Notice d'utilisation, ENSM-LAN, Nantes*, 1989.

[34] O. Alba, A. Urquizo, J. A. Pámanes, J. Pámanes, M. Nieto, C. Ocón, and G. Julián, "Sistema de Programación Fuera de Línea Para Robots de Soldadura. Etapa de Planificación de Movimientos," *Memorias del XVIII Congreso Internacional Anual de la SOMIM*, pp. 961–970, 2012.

[35] U. Zaldívar-Colado, L. Arias, J. A. Pámanes, M. Nieto, R. Ávila-Rondón, and J. P. Nieto, "Programación Fuera de Línea de Robots de Soldadura en el Paquete WEROP Usando una Interfaz Háptica," *Memorias del XX Congreso Internacional Anual de la Sociedad Mexicana de Ingeniería Mecánica*, 2014.

[36] U. Zaldívar-Colado, D. Murillo, X. Zaldívar, E. Osuna, and V. Nalda, "Interaction Technique for Virtual Robot Stabilization with Dynamic Behavior," *Congreso Internacional de Ingeniería Electrónica, Biomédica, Computación e Informática*, pp. 335–349, 2009.

[37] U. Zaldívar-Colado, D. Murillo-Campos, X. Zaldívar-Colado, C. Marmolejo-Rivas, and R. Bernal-Guadiana, "Realistic manipulation of virtual robot in dynamic virtual environment," in *ASME 2010 World Conference on Innovative Virtual Reality*, pp. 341–349, American Society of Mechanical Engineers, 2010.

[38] S. Garbaya and U. Zaldívar-Colado, "The affect of contact force sensations on user performance in virtual assembly tasks," *Virtual Reality, Springer*, vol. 11, no. 4, pp. 287–299, 2007.

[39] I. Karabegović, E. Karabegović, S. Pašić, S. Isić, *et al.*, "Worldwide trend of the industrial robot applications in the welding processes," *EUROPE*, vol. 34, no. 20.483, pp. 30–630, 2012.

[40] Z. Pan, J. Polden, N. Larkin, S. Van Duin, and J. Norrish, "Recent progress on programming methods for industrial robots," *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 2, pp. 87–94, 2012.

[41] S. Mitsi, K.-D. Bouzakis, G. Mansour, D. Sagris, and G. Maliaris, "Off-line programming of an industrial robot for manufacturing," *The International Journal of Advanced Manufacturing Technology*, vol. 26, no. 3, pp. 262–267, 2005.

[42] M. L. Hornick and B. Ravani, "Computer-aided off-line planning and programming of robot motion," *The International journal of robotics research*, vol. 4, no. 4, pp. 18–31, 1986.

[43] N. Xiao, J. Guo, S. Guo, and T. Tamiya, "A robotic catheter system with real-time force feedback and monitor," *Australasian Physical & Engineering Sciences in Medicine*, vol. 35, no. 3, pp. 283–289, 2012.

[44] G. C. Burdea, "Invited review: the synergy between virtual reality and robotics," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 3, pp. 400–410, 1999.

[45] N. V. Navkar, Z. Deng, D. J. Shah, K. E. Bekris, and N. V. Tsekos, "Visual and force-feedback guidance for robot-assisted interventions in the beating heart with real-time MRI," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 689–694, IEEE, 2012.

[46] C. W. Borst and A. P. Indugula, "Realistic virtual grasping," in *Virtual Reality, 2005. Proceedings. VR 2005. IEEE*, pp. 91–98, IEEE, 2005.

[47] R. S. Wright Jr, N. Haemel, G. M. Sellers, and B. Lipchak, *OpenGL SuperBible: comprehensive tutorial and reference*. Pearson Education, 2010.

[48] M. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. John Wiley & Sons Inc, 2006.

[49] K. S. Fu, R. C. Gonzalez, and C. S. G. Lee, *ROBÓTICA: Control, detección, visión e inteligencia*. McGraw Hill. México., 1988.

[50] W. Khalil and J. Kleinfinger, "A new geometric notation for open and closed-loop robots," in *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, vol. 3, pp. 1174–1179, IEEE, 1986.

[51] R. P. Paul, *Robot manipulators: mathematics, programming, and control: the computer control of robot manipulators*. MIT Press, 1981.

[52] Y. Wang and N. Chi, "Path planning optimization for teaching and playback welding robot," *TELKOMNIKA Indonesian Journal of Electrical Engineering*, vol. 11, no. 2, pp. 960–968, 2013.

[53] Y. Xu, H. Yu, J. Zhong, T. Lin, and S. Chen, "Real-time seam tracking control technology during welding robot GTAW process based on passive vision sensor," *Journal of Materials Processing Technology*, vol. 212, no. 8, pp. 1654–1662, 2012.

[54] L. Jingwei, T. Yifei, W. Shaofeng, T. Qingmeng, and L. Dongbo, "Welding Robot Kinematics Analysis and Trajectory Planning," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 14, no. 2A, pp. 92–100, 2016.

[55] C. Chen, S. Hu, D. He, and J. Shen, "Kinematic analysis and trajectory planning of J-groove welding robot," *Transactions of Tianjin University*, vol. 18, no. 5, pp. 350–356, 2012.

[56] J. N. Pires, A. Loureiro, and G. Bölmsjo, *Welding robots: Technology, system issues and application*. Springer Science & Business Media, 2006.

[57] H. Cary and S. Helzer, "Modern Welding Technology, 2005," 2005.

[58] W. Dai and M. Kampker, "User oriented integration of sensor operations in a offline programming system for welding robots," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 2, pp. 1563–1567, IEEE, 2000.

[59] M. Dinham and G. Fang, "Weld seam detection using computer vision for robotic arc welding," in *Automation Science and Engineering (CASE), 2012 IEEE International Conference on*, pp. 771–776, IEEE, 2012.

[60] C.-S. Kim, K.-S. Hong, H. Y.-S. Han, S.-H. Kim, and S.-C. Kwon, "PC-based off-line programming using VRML for welding robots in shipbuilding," in *Robotics, Automation and Mechatronics, 2004 IEEE Conference on*, vol. 2, pp. 949–954, IEEE, 2004.

[61] J. N. Pires, T. Godinho, and P. Ferreira, "CAD interface for automatic robot welding programming," *Industrial Robot: An International Journal*, vol. 31, no. 1, pp. 71–76, 2004.

[62] X. Tang and D. Paul, "3D-visualized offline-programming and simulation system for industrial robots," *Transactions of the China Welding Institution*, vol. 26, no. 2, pp. 64–68, 2005.

[63] J. A. Pámanes and A. Fernández, "Análisis del Desempeño de un Robot Manipulador en la Ejecución de una Operación de Soldadura," *Memorias del I Congreso Nacional de Robótica*, pp. 71–79, 1996.

[64] K. Brink, M. Olsson, and G. Bolmsjö, "Increased autonomy in industrial robotic systems: A framework," *Journal of Intelligent and Robotic Systems*, vol. 19, no. 4, pp. 357–373, 1997.

[65] S. Chen and T. Lin, "Intelligent welding robot technology," *Chinese Mechanical Industry Press, Beijing.*, 2006.

[66] Z. Pan, J. Polden, N. Larkin, S. van Duin, and J. Norrish, "Automated offline programming for robotic welding system with high degree of freedoms," in *Advances in Computer, Communication, Control and Automation*, pp. 685–692, Springer, 2011.

[67] Y. F. Yong, J. A. Gleave, J. L. Green, and M. C. Bonney, *Off-line programming of robots*. Wiley, 1985.

[68] E. Freund, D. Rokossa, and J. Roßmann, "Process-oriented approach to an efficient off-line programming of industrial robots," in *Industrial Electronics Society, 1998. IECON'98. Proceedings of the 24th Annual Conference of the IEEE*, vol. 1, pp. 208–213, IEEE, 1998.

[69] G. Wittenberg, "Developments in offline programming: an overview," *Industrial Robot: An International Journal*, vol. 22, no. 3, pp. 21–23, 1995.

[70] H. Gurocak, S. Jayaram, B. Parrish, and U. Jayaram, "Weight sensation in virtual environments using a haptic device with air jets," *Journal of Computing and Information Science in Engineering*, vol. 3, no. 2, pp. 130–135, 2003.

[71] T. Asano and Y. Ishibashi, "Guidance services for a haptic museum in distributed virtual environments," in *Proc. ICAT2004 Workshop on VR and Entertainment Technology*, pp. 39–42, 2004.

[72] I.-B. Pavaloiu, R. Ioanitescu, G. Dragoi, S. Grigorescu, and S. A. Sandu, "Virtual reality for education and training in dentistry," in *The International Scientific Conference eLearning and Software for Education*, vol. 1, p. 367, " Carol I" National Defence University, 2016.

[73] T. Lim, J. M. Ritchie, R. G. Dewar, J. R. Corney, P. Wilkinson, M. Calis, M. Desmulliez, and J.-J. Fang, "Factors affecting user performance in haptic assembly," *Virtual reality*, vol. 11, no. 4, pp. 241–252, 2007.

[74] B. Petzold, M. F. Zaeh, B. Faerber, B. Deml, H. Egermeier, J. Schilp, and S. Clarke, "A study on visual, auditory, and haptic feedback for assembly tasks," *Presence: teleoperators and virtual environments*, vol. 13, no. 1, pp. 16–21, 2004.

[75] S. Jeong, N. Hashimoto, and S. Makoto, "A novel interaction system with force feedback between real-and virtual human: an entertainment system: virtual catch ball," in *Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pp. 61–66, ACM, 2004.

[76] M. Hosseini, F. Malric, and N. D. Georganas, "A haptic virtual environment for industrial training," in *Haptic Virtual Environments and Their Applications, IEEE International Workshop 2002 HAVE*, pp. 25–30, IEEE, 2002.

[77] R. J. Adams and B. Hannaford, "Stable haptic interaction with virtual environments," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 3, pp. 465–474, 1999.

[78] A. El Saddik, "The potential of haptics technologies," *IEEE Instrumentation & Measurement Magazine*, vol. 10, no. 1, pp. 10–17, 2007.

[79] J. E. Colgate, P. E. Grafing, M. C. Stanley, and G. Schenkel, "Implementation of stiff virtual walls in force-reflecting interfaces," in *Virtual Reality Annual International Symposium, 1993., 1993 IEEE*, pp. 202–208, IEEE, 1993.

[80] C. W. Borst and A. P. Indugula, "A spring model for whole-hand virtual grasping," *Presence: Teleoperators & Virtual Environments*, vol. 15, no. 1, pp. 47–61, 2006.

[81] B. Shneiderman, *Designing the user interface: strategies for effective human-computer interaction*. Addison-Wesley, 3rd ed., 1998.

[82] E. Angel and D. Shreiner, *Interactive computer graphics: A top-down approach with shader-based OpenGL*, vol. 1. Pearson, 6th ed., 2012.

[83] U. Zaldívar-Colado, *Planification d'Assemblage en Environnement Virtuel*. PhD thesis, Université de Versailles Saint Quentin-en-Yvelines, France, 2009.

[84] M. Botsch, M. Pauly, C. Rossl, S. Bischoff, and L. Kobbelt, "Geometric modeling based on triangle meshes," in *ACM SIGGRAPH 2006 Courses*, p. 1, ACM, 2006.

[85] B. M. Howard and J. M. Vance, "Desktop haptic virtual assembly using physically based modelling," *Virtual Reality*, vol. 11, no. 4, pp. 207–215, 2007.

[86] Sensable Technologies , *OpenHaptics Toolkit, Programmers Guide*, 2009.

[87] Fanuc America Corporation, "Intelligent Arc Welding Robot: Fanuc Arc Mate 100ic," 2015.

[88] W. Khalil and P. Lemoine, "SYMORO+ Symbolic Modeling of Robots: User Guide," tech. rep., IRCCyN (Institut de Recherche en Communication et Cybernétique de Nantes), Nantes, Francia, 2003.

[89] P. Neto, J. N. Pires, and A. P. Moreira, "Robot path simulation: a low cost solution based on CAD," in *Robotics Automation and Mechatronics (RAM), 2010 IEEE Conference on*, pp. 333–338, IEEE, 2010.

[90] M. Soron and I. Kalaykov, "Generation of continuous tool paths based on CAD models for Friction Stir Welding in 3D," in *Control & Automation, 2007. MED'07. Mediterranean Conference on*, pp. 1–5, IEEE, 2007.

[91] S. C. Mondesire, D. B. M. J. Stevens, S. Zielinski, and G. A. Martin, "Physics engine benchmarking in three-dimensional virtual world simulation," *MODSIM World*, pp. 5–8, 2016.

[92] K.-S. Choi, S.-H. Chan, and W.-M. Pang, "Virtual suturing simulation based on commodity physics engine for medical learning," *Journal of medical systems*, vol. 36, no. 3, pp. 1781–1793, 2012.

[93] AGEIA Technologies Inc, *PhysX SDK 2.3.1 Documentation*, 2005.

[94] A. Sanchez-Diaz, U. Zaldivar-Colado, J. A. Pamanes-Garcia, and X. Zaldivar-Colado, "Operation of a haptic interface for offline programming of welding robots by applying a spring-damper model," *International Journal of Computer Integrated Manufacturing*, pp. 1–19, 2019.

[95] J. E. Colgate, M. C. Stanley, and J. M. Brown, "Issues in the haptic display of tool use," in *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, vol. 3, pp. 140–145, IEEE, 1995.

[96] Fanuc Robotics America Inc., "ArcTool Application Software," 2011.

[97] E. Karadeniz, U. Ozsarac, and C. Yildiz, "The effect of process parameters on penetration in gas metal arc welding processes," *Materials & design*, vol. 28, no. 2, pp. 649–656, 2007.

[98] FANUC Robotics America Corporation, *ArcTool Operations & Programming. Student Manual.*, 2013.

[99] Y. Guerrero Ávila, "Contribución al estudio de la selección de parámetros de la estación de soldadura robotizada Fanuc 100 iC," Master's thesis, Tecnológico Nacional de México, Instituto Tecnológico de la Laguna, 2017.

[100] FANUC Robotics America Corporation, *FANUC Robotics SYSTEM R-30iA and R-30iB Controller KAREL Reference Manual.*, 2013.