Planeación de trayectorias en vuelo de un manipulador industrial para el Laboratorio Fábrica Experimental UN

Julio César García Carrero

Universidad Nacional de Colombia
Facultad de Ingeniería
Bogotá, Colombia
2017

Planeación de trayectorias en vuelo de un manipulador industrial para el Laboratorio Fábrica Experimental UN

Julio César García Carrero

Tesis o trabajo de grado presentada(o) como requisito parcial para optar al título de:
Magister en Ingeniería - Automatización Industrial

Director:
Prof. Ernesto Córdoba Nieto

Línea de Investigación:
Robótica y Manufactura
Grupo de Investigación:
Grupo de Trabajo en Nuevas Tecnologías de Diseño y Manufactura - Automatización

Universidad Nacional de Colombia
Facultad de Ingeniería
Bogotá, Colombia
2017

Dedicatoria

A mis padres por su amor y apoyo incondicional

A Elizabeth, mi amor, mi amiga

Agradecimientos

Quiero expresar mi agradecimiento al profesor Ernesto Córdoba Nieto por su orientación y aporte en la realización de este trabajo.

A la empresa IMOCOM S.A. por brindarme la oportunidad de desarrollarme profesionalmente en conjunto con la actividad académica.

A Ted Miller de Yaskawa America y Gijs van der Hoorn de TU Delft por su continuo interés y valiosa ayuda durante las pruebas de comunicación.

A mis padres y a Elizabeth por su apoyo en los momentos difíciles.

A todas las personas que han participado en esta investigación con su conocimiento, observaciones y apoyo.

Resumen

El presente trabajo de investigación se enfoca en la planeación de trayectorias para un sistema robótico industrial de ocho articulaciones utilizando la plataforma ROS para el desarrollo de un modelo virtual que permita comandar movimientos hacia el robot y sus ejes externos. Este sistema incluye una configuración única que no ha sido explorada previamente a través de herramientas de software similares.

Para la validación del modelo virtual y de comunicación desarrollado se plantea la exploración experimental del sistema en aplicaciones de manufactura y de interacción básica con robots móviles. En el primer caso se aborda la generación de trayectorias utilizando 7 y 8 DOF para el mecanizado de superficies con forma libre o definición paramétrica. Las pruebas correspondientes se realizan sobre materiales blandos, adaptando una herramienta motorizada al manipulador industrial Motoman MH6 y usando una fresa cilíndrica de punta plana. Durante la trayectoria se lleva a cabo la interpolación de todos los ejes involucrados para alcanzar una orientación normal de la herramienta con respecto a cada punto de la superficie, proporcionando resultados satisfactorios en la calidad de la pieza. Igualmente se efectúan mediciones de velocidad y torque a los que se encuentran sometidos cada articulación, así como mediciones de precisión en el mecanizado de probetas fabricadas para validar las condiciones operativas del proceso. Por otra parte, en las pruebas básicas de interacción con un robot móvil (AGV) que se desplaza de manera autónoma de acuerdo a su algoritmo de navegación, se realiza una planeación continua de la trayectoria para mover el robot industrial hacia un objetivo dinámico correspondiente a la posición del AGV en cada instante. Esta situación evidencia un efecto de seguimiento en los robots definiendo una distancia de separación entre ellos.

Palabras clave: ROS, planeación de trayectorias, mecanizado robotizado, superficies de forma libre, robótica colaborativa.

Abstract

This research focuses on path planning for an industrial robotic system with eight joints using ROS in order to develop a virtual model to command the robot and its external axes movements. This system has a particular setup which has not been explored with similar software tools.

For virtual and communication models validation it is proposed an experimental exploration of the system in manufacturing applications and basic interaction with mobile robots. In the first case, the trajectories generation with 7 and 8 DOF is approached for machining of free-shaped or parametric surfaces. The corresponding tests are performed on soft materials, with adapting a mototool to Motoman MH6 industrial manipulator and using a flat-cylindrical milling tool. All joints interpolate along the trajectory to achieve normal tool orientation for every point in the surface. It provides suitable results for part quality. Further, speed and torque measurements for each joint are registered, as well as machining precision measurements over test pieces to verify operating conditions in the process. Additionally, in the basic tests for interaction with a mobile robot (AGV) which moves autonomously according with its navigation algorithm a path is planned to move the industrial robot toward a dynamic target defined by the AGV's instant position. This situation shows a tracking effect among the robots with a fixed distance between them.

Keywords: ROS, path planning, robotic machining, free-shaped surfaces, collaborative robotics

Contenido

Lista de símbolos

Símbolos con letras latinas

Símbolo	Término	Unidad SI	Definición
X	Eje X cartesiano		
Y	Eje Y cartesiano		
Z	Eje Z cartesiano		

Abreviaturas

Abreviatura	Término
DOF	*Degrees of Freedom*
OLP	*Off-line Programming*
CAD	*Computer Aided Design*
CAM	*Computer Aided Manufacturing*
ROS	*Robot Operating System*
ROS-I	*ROS-Industrial*
GUI	*Graphical User Interface*
UI	*User Interface*
AGV	*Automatic Guided Vehicle*
RRT	*Rapidly-exploring Random Trees*
PRM	*Probabilistic Roadmap Method*
RRT-GD	*RRT-Goal Directed*
PRMwO	*Probabilistic Road Map with Obstacles*
OMPL	*Open Motion Planning Library*

Abreviatura	**Término**
KPIECE	*Kinematic Planning by Interior-Exterior Cell Exploration*
IFR	*International Federation of Robotics*
CNC	*Computer Numerical Control*
FEA	*Finite Element Analysis*
MSA	*Matrix Structural Analysis*
VJM	*Virtual Joint Modeling*
TCP	*Tool Center Point*
BSD	*Berkeley Software Distribution*
TCP/IP	*Transmission Control Protocol/Internet Protocol*
JSON	*JavaScript Object Notation*
RWT	*Robot Web Tools*
AR	*Augmented Reality*
VTK	*Visualization Tool Kit*
SDK	*Software Development Kit*
LabFabEx	Laboratorio Fábrica Experimental
VMD	*Virtual Manufacturing Device*
URDF	*Unified Robot Description Format*
IK	*Inverse Kinematics*
PLY	*Polygon File Format*
STL	*STereoLithography*

Lista de Figuras

Lista de Tablas

1. Introducción

La automatización mediante el uso de robots industriales constituye una vía interesante de solución en cuanto a productividad y flexibilidad. Sin embargo, la programación de estos sistemas requiere bastante tiempo, particularmente cuando la aplicación específica aumenta en su complejidad, e implica adicionalmente un conocimiento preliminar respecto al funcionamiento del equipo [Kohrt et al., 2013]. Existen dos métodos principales para la programación de robots, los cuales corresponden a programación online (que incluye la enseñanza mediante un *teach pendant* de los movimientos que conforman la trayectoria) y programación offline (OLP), que a partir de modelos 3D de la pieza permite generar y simular los programas del robot, validando de esta manera la correcta ejecución del proceso (sin colisiones) para transmitirlos posteriormente al controlador [Andersson et al., 2016].

Adicionalmente, debido al auge de robots autónomos y manufactura virtual en los últimos años, el problema de planeación de trayectorias se ha convertido en un tema de amplio interés en investigación [Carlson et al., 2013]. Sin embargo, considerando que en general los algoritmos implementados al respecto implican alta complejidad computacional y tiempo de procesamiento [Liu et al., 2016], no es una tarea trivial calcular trayectorias, especialmente de forma rápida [Guernane and Achour, 2011].

De la misma manera, el entorno de los manipuladores se torna progresivamente más sofisticado mediante la aparición de posibles obstáculos o la presencia de otros robots compartiendo el espacio de trabajo. Por ello, la planeación de movimientos evadiendo efectivamente las colisiones es necesaria para el desarrollo de aplicaciones robóticas complejas. Los algoritmos desarrollados para este tipo de tareas mejoran el desempeño de los manipuladores a la vez que reducen la cantidad de cálculos en la generación de la trayectoria [Ma et al., 2016].

Por otra parte, con el desarrollo de procesos automatizados de manufactura, los robots industriales han sido ampliamente utilizados en aplicaciones alternativas a las tareas convencionales de manipulación y soldadura, ya que permiten mejorar significativamente la eficiencia del proceso, garantizar calidad de los productos y reducir costos [Liu et al., 2016]. Específicamente, los robots industriales de 6 DOF se han convertido en una importante herramienta para diferentes procesos de mecanizado por su capacidad de realizar cualquier movimiento en un espacio de trabajo extendido, al igual que la posibilidad de medición y realimentación en tiempo real de los torques experimentados para llevar a cabo control de las fuerzas de

contacto causadas por la interacción entre la herramienta y la pieza [Klimchik et al., 2016].

Sin embargo; a partir de su estructura basada en eslabones, la rigidez de un robot articulado es usualmente cincuenta veces menor que en las máquinas herramientas. Este aspecto, presentado por la flexibilidad de sus articulaciones induce vibraciones en el efector final, lo cual afecta la precisión y las condiciones de mecanizado [Olabi et al., 2010]. Por lo anterior, las aplicaciones de remoción de material con robots se encuentran limitadas a piezas de mayor tolerancia geométrica o al uso de materiales menos resistentes [Kubela et al., 2015].

En línea con la tendencia de mecanizado robotizado, fabricantes de software CAD/CAM han implementado post-procesadores para la generación de trayectorias y simulación de aplicaciones de mecanizado con robots. Para esto, el movimiento nominal de la herramienta es calculado con respecto a un marco de referencia establecido y posteriormente transformado al lenguaje nativo del correspondiente controlador. Sin embargo, ninguna de estas herramientas de software se involucra realmente con la generación de trayectorias para el robot [Halbauer et al., 2013].

El desarrollo de software específico para plataformas robóticas ha presentado un creciente interés, generando significativos avances en este campo. De manera similar a los sistemas operativos, estas herramientas gestionan interfaces entre el hardware y software para proporcionar drivers de los dispositivos, estructuras de datos, herramientas de visualización y transmisión de mensajes, entre otros. Esto ha generado un incremento en la velocidad a la cual los investigadores pueden crear aplicaciones para diversos tipos de robots [Toris et al., 2015]; a la vez que establece una infraestructura que facilita la interoperabilidad y la reutilización de código, necesarias para el uso de robots a gran escala [Crick et al., 2017]. Así mismo permiten obtener una integración sencilla y completa entre sistemas robóticos en entornos complejos, de forma que estos pueden diferir en capacidades o funcionalidades pero trabajan de manera conjunta en aplicaciones dinámicas altamente flexibles [Ferrati et al., 2016], o proporcionan acceso a los robots a través de Internet mediante la creación de interfaces de usuario intuitivas [Crick et al., 2017].

La idea descrita sugiere entonces la posibilidad de crear laboratorios robóticos virtuales que eliminen la barrera de acceso a recursos físicos y faciliten a los investigadores realizar experimentos y comparar resultados [Crick et al., 2017].

Este trabajo de investigación plantea el desarrollo de una plataforma robótica a través de ROS, mediante la cual se obtiene interacción completa con un robot de 6 DOF que incluye dos ejes externos y en conjunto forman un sistema robótico de 8 DOF. A partir de lo anterior, es posible realizar tareas de planeación de trayectorias en vuelo libres de colisiones y comandar la ejecución de los movimientos generados hacia el robot. El modelo de comunicación es

implementado con base en un driver de ROS-I creado para Motoman, el cual se conecta al software MotoROS ejecutándose en el controlador. Por tratarse de una configuración física única, en la literatura no existe registro de aplicaciones de este tipo.

Para la validación de la plataforma se aborda el desarrollo experimental de procesos de mecanizado robotizado en materiales blandos para la obtención de diferentes superficies con formas libres. En este caso, la generación de trayectorias se lleva a cabo mediante herramientas de computación gráfica aplicadas a un archivo CAD correspondiente a la pieza deseada. Debido a la conformación del sistema robótico, las pruebas de mecanizado se realizan con la pieza fija en una mesa (7 DOF) o en la estructura del posicionador incluyendo movimientos interpolados (8 DOF). Por otra parte se aplica una orientación a la herramienta para hacerla coincidente con la normal a cada punto en la superficie, lo cual constituye un concepto fundamental en el mecanizado multi-ejes.

Adicionalmente, se realizan pruebas de mecanizado mediante la definición de superficies definidas matemáticamente. Lo anterior incluye el uso de formas descritas a partir de ecuaciones parametrizadas o construidas mediante entidades geométricas básicas. Durante la ejecución de cada tarea son registrados los valores de torque y velocidad alcanzados por cada articulación.

Finalmente, se plantea una aproximación hacia el desarrollo de aplicaciones que incluyen interacción entre el robot Motoman y un robot móvil AGV, definiendo una tarea de seguimiento entre los dispositivos. Para ello, el robot móvil se desplaza en el laboratorio mediante su algoritmo de navegación y una vez ingresa al espacio de trabajo alcanzable del Motoman, una trayectoria libre de colisiones es planeada y ejecutada a partir de la posición estimada del AGV en cada instante.

La estructura general del trabajo desarrollado en esta tesis se presenta en la Figura **1-1**.

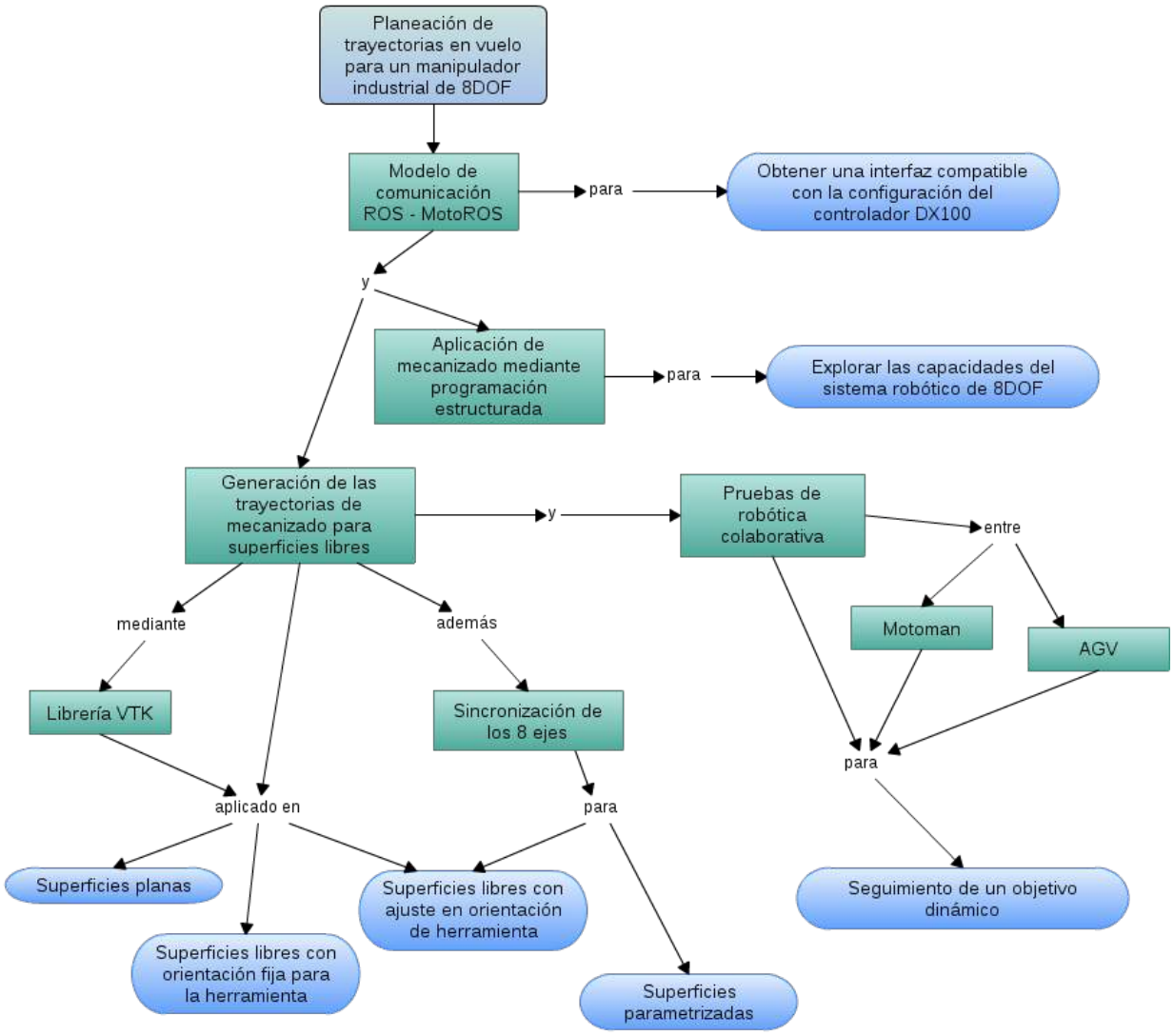


Figura 1-1.: Estructura general de la tesis

2. Estado del arte

2.1. Planeación de trayectorias para robots

En los últimos años debido al auge de robots autónomos y manufactura virtual, el problema de planeación de trayectorias ha presentado amplio interés. De manera general, el principio funcional para los algoritmos desarrollados a partir de esta temática es buscar segmentos conectados de trayectorias desde un punto de inicio hasta un objetivo establecido, mientras son evadidas las colisiones con elementos en el entorno o manteniendo una distancia de separación con respecto a un objeto [Carlson et al., 2013]. Adicionalmente, estos algoritmos implican alta complejidad y consumo de tiempo especialmente a medida que aumentan los grados de libertad involucrados en el robot [Liu et al., 2016].

Algunas de las técnicas más efectivas en la planeación de trayectorias son aquellas basadas en muestreos, las cuales pueden ser clasificadas en determinísticas y probabilísticas dependiendo de la forma en la cual son generadas las muestras. Ejemplos típicos de aproximaciones determinísticas son los algoritmos A* y los campos potenciales. Para el caso probabilístico los más relevantes corresponden a métodos *Rapidly-exploring Random Trees* (RRT) y *Probabilistic Roadmap Method* (PRM) [Rodríguez et al., 2014]. Conceptualmente la planeación de movimientos basada en muestreo es sencilla, pero la implementación de los algoritmos no es una tarea trivial [Sucan et al., 2012].

Este tipo de métodos son adecuados en la planeación de trayectorias libres de colisiones para el caso de robots móviles y manipuladores industriales de múltiples grados de libertad. La idea fundamental es construir un grafo con puntos o nodos libres de obstáculos dentro del espacio alcanzable del robot, y con aristas como segmentos válidos de trayectoria, mediante un muestreo aleatorio (Ver Figura **2-1**) [Ge et al., 2016]. Si no se presenta ninguna colisión del robot en su desplazamiento hacia el nuevo nodo, entonces éste se considera válido y es agregado al árbol. La representación en el espacio de articulaciones obtenida a través del cálculo de cinemática inversa, debe igualmente ser libre de colisiones. En caso contrario, el punto es desechado y debe planearse de nuevo. Este procedimiento es ejecutado iterativamente hasta alcanzar el objetivo y como resultado, la trayectoria obtenida es un conjunto discreto de puntos [Ma et al., 2016]. Este método no requiere un modelo preciso de los obstáculos; en contraste, solamente requiere establecer si un punto se encuentra definido en espacio libre y alcanzable [Ge et al., 2016].

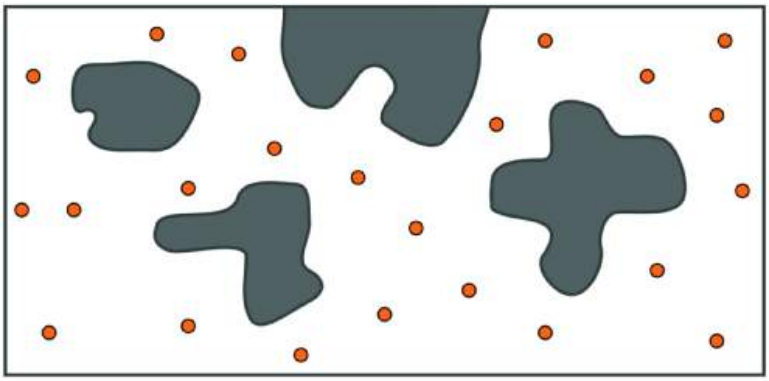


Figura 2-1.: Conjunto de muestras aleatorias en el espacio libre [Kavraki Lab, 2014]

Las trayectorias obtenidas con los algoritmos mencionados pueden ser optimizadas utilizando métodos de post procesamiento [Rodríguez et al., 2014].

Diversas modificaciones son propuestas a los algoritmos de planeación desarrollados. Una implementación del método RRT con direccionalidad del objetivo (RRT-GD) es sugerido por [Ge et al., 2016], a partir de definir la dirección de crecimiento del árbol hacia el objetivo.

Un enfoque en la planeación de trayectorias para un sistema robótico de dos brazos es abordado en [Rodríguez et al., 2014], considerando la posibilidad de utilizar los propios robots para retirar potenciales obstáculos y obtener de esta manera un acceso libre a los objetos que se desean manipular. Para ello, hace uso de un algoritmo que ha denominado *Probabilistic Road Map with Obstacles* (PRMwO); el cual retorna la trayectoria para un robot hacia su objetivo, junto con una lista de los obstáculos que deben ser removidos.

Por otra parte, este problema es igualmente abordado mediante el uso de técnicas de inteligencia artificial como en [Ming and Yong, 2015], el cual presenta un control difuso basado en algoritmo de colonia de hormiga para calcular la solución de cinemática inversa y aborda esta estrategia como un método de planeación de trayectorias para un manipulador serial.

Finalmente, a la funcionalidad principal de planeación se incluyen restricciones o criterios de optimización. [Rubio et al., 2016] utiliza el tiempo como una función objetivo para obtener trayectorias, así como especificaciones asociadas al comportamiento mecánico del robot, tales como potencia máxima, máximos torques y *jerks*. Adicionalmente, [Valente et al., 2017] presenta un modelo de generación de trayectorias suavizadas, con optimización simultánea del tiempo de ejecución para todas las articulaciones, mediante una aproximación multi-variable.

2.1.1. The Open Motion Planning Library

OMPL es una librería de código abierto para la planeación de trayectorias basadas en muestreo, la cual contiene la implementación en C++ de diversos algoritmos; a la vez que facilita la inclusión de nuevos métodos, proporcionando una conveniente interconexión con otras herramientas de software [Sucan et al., 2012].

Usualmente es desafiante demostrar que un nuevo algoritmo de planeación o una modificación propuesta es realmente una mejora sobre los ya existentes, considerando algún parámetro de interés. Lo anterior, debido a que implica no solamente una implementación del nuevo modelo sino también de diversos algoritmos con el fin de efectuar una comparación. Por lo tanto, OMPL constituye una base para este tipo de propósitos [Moll et al., 2015].

Entre algunos algoritmos de planeación implementados en OMPL se incluyen:

- Rapidly-exploring Random Trees (RRT) [LaValle and Kuffner, 1999]
  Es un método (basado en árboles) con un principio similar al de campos potenciales, el cual explora rápida y uniformemente el espacio [Valle, 2011]. El fundamento de los algoritmos basados en árboles se presenta en la Figura **2-2**.
  - RRT Connect (RRTConnect)
  - Lazy RRT (LazyRRT)

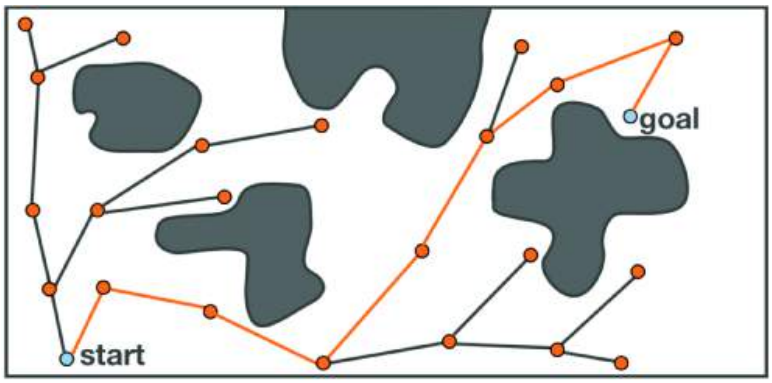


Figura 2-2.: Despliegue de un algoritmo basado en árboles [Kavraki Lab, 2014]

- Kinematic Planning by Interior-Exterior Cell Exploration (KPIECE)
  Es un algoritmo basado en árboles el cual utiliza una discretización a diferentes capas o niveles para guiar la exploración en el espacio [Şucan and Kavraki, 2012]. Al utilizar árboles, su funcionamiento es similar a la Figura **2-2** y la representación multinivel es como se muestra en la Figura **2-3**, en la que se evidencia un aumento en la resolución con el cambio de etapa.
  - Bi-directional KPIECE (BKPIECE)

- Lazy Bi-directional KPIECE (LBKPIECE)

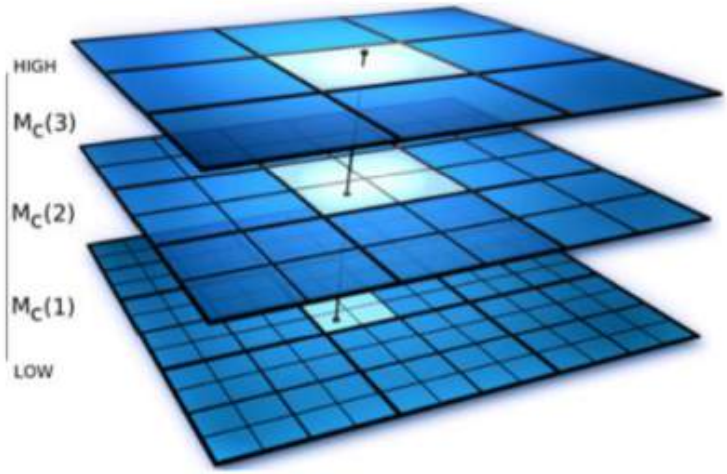


Figura 2-3.: Ejemplo de discretización a tres niveles [Şucan and Kavraki, 2012]

- Probabilistic Roadmap Method (PRM) [Kavraki et al., 1996]
  PRM es una de las primeras estrategias desarrolladas para la planeación de trayectorias. Este algoritmo intenta conectar nodos cercanos a partir del muestreo aleatorio. Una vez el mapa se encuentra completo, se busca una conexión entre el punto inicial y el objetivo. La Figura **2-4** ilustra el comportamiento de este algoritmo [Kavraki Lab, 2014].
  - LazyPRM

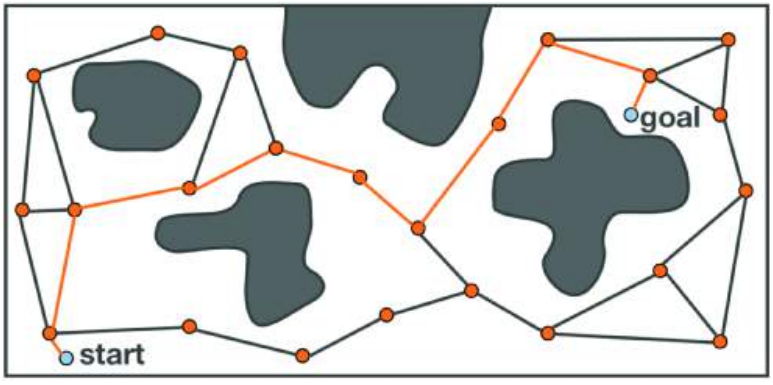


Figura 2-4.: Despliegue de un algoritmo basado en PRM [Kavraki Lab, 2014]

El desarrollo de OMPL incluye una interfaz gráfica sencilla que proporciona una aproximación a su uso para planeación de trayectorias. Adicionalmente, ha sido implementado en software robótico como ROS [Sucan et al., 2012].

2.2. Procesos de manufactura mediante el uso de robots industriales

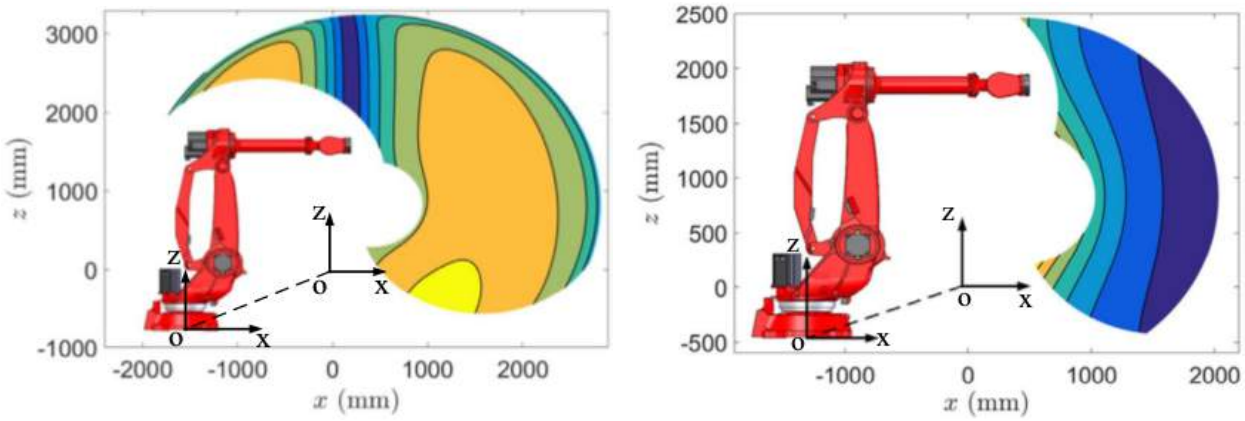
Los robots industriales han sido ampliamente utilizados para procesos de soldadura y manipulación. De acuerdo con la Federación Internacional de Robótica (IFR), más del 70 % de todos los robots industriales son usados en estas áreas. Sin embargo, estos cuentan con un amplio potencial en operaciones de manufactura tales como rectificado, desbarbado, pulido, corte, taladrado o mecanizado, representando menos del 5 % del mercado pero con un continuo crecimiento [Kubela et al., 2015].

Con respecto a las máquinas CNC estándar, los robots presentan espacios de trabajos más grandes y que se hacen fácilmente extendibles o ajustables a partir de la inclusión de ejes externos. Adicionalmente, los centros de mecanizado pueden alcanzar hasta 5DOF mientras que los robots disponen generalmente de al menos 6DOF, dicha redundancia puede ser utilizada para optimizar los movimientos de la herramienta. De esta manera, las trayectorias de mecanizado son más flexibles y versátiles pues facilitan la obtención de formas tridimensionales complejas [Klimchik et al., 2017]. Sin embargo, los robots industriales presentan menor precisión en el posicionamiento y debido a su cinemática serial proporcionan menor rigidez, particularmente en la muñeca [Kubela et al., 2015].

Específicamente, el manejo de singularidades para el codo y la muñeca del robot constituye un problema durante los movimientos relativos entre la herramienta de corte y la pieza, con el interés de evitar cambios de dirección repentinos en el movimiento de los ejes. Estos movimientos relativos junto con aspectos como la estrategia de mecanizado, el material de la pieza, la herramienta y los parámetros de corte pueden generar vibraciones en el sistema, lo cual conlleva a afectar la calidad de la superficie mecanizada, así como la integridad de la herramienta y del robot. Se trata de un problema complejo y fundamental ya que el riesgo de vibración depende de muchos factores [Halbauer et al., 2013].

Debido al efecto de la postura del robot en el comportamiento de la rigidez, [Lin et al., 2017] lleva a cabo un estudio que plantea la conveniencia de realizar los procesos de mecanizado en las regiones del espacio donde la cinemática y la dinámica alcanzan su mejor desempeño. Particularmente, para un manipulador de 6DOF obtiene que el mejor desempeño dinámico se presenta en el espacio de trabajo indicado en la Figura **2-5**a, mientras que para el caso de la rigidez corresponde a la Figura **2-5**b.

En contraste, [Klimchik et al., 2017] considera la deflexión Φ ocasionada por las fuerzas de corte en el proceso de mecanizado, la cual puede ser calculada a partir de la matriz de rigidez



(a) Espacio con mejor dinámica (b) Espacio con mejor rigidez

Figura 2-5.: Espacio de trabajo con mejor desempeño dinámico y de rigidez [Lin et al., 2017]

de un manipulador K_c, de acuerdo con la ecuación 2-1.

$$\Phi = K_c^{-1} \cdot F \tag{2-1}$$

para una fuerza o torque externo F, que en el mecanizado corresponde a las fuerzas de corte. K_c es definida positiva y varía a lo largo del espacio de trabajo del robot. Entre algunos métodos que permiten obtener esta matriz se incluyen el análisis de elementos finitos (FEA), matriz de análisis estructural (MSA) y el modelamiento virtual de articulaciones (VJM). Finalmente, a partir de lo anterior se define el espacio de trabajo de mayor conveniencia para ubicar la pieza que será mecanizada. Ver Figura **2-6**

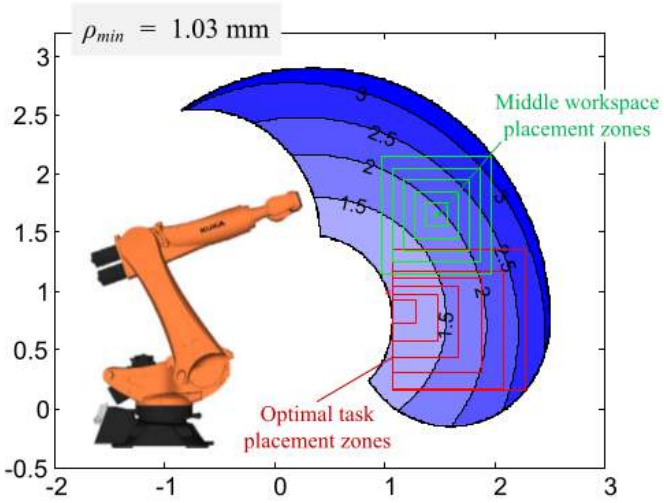
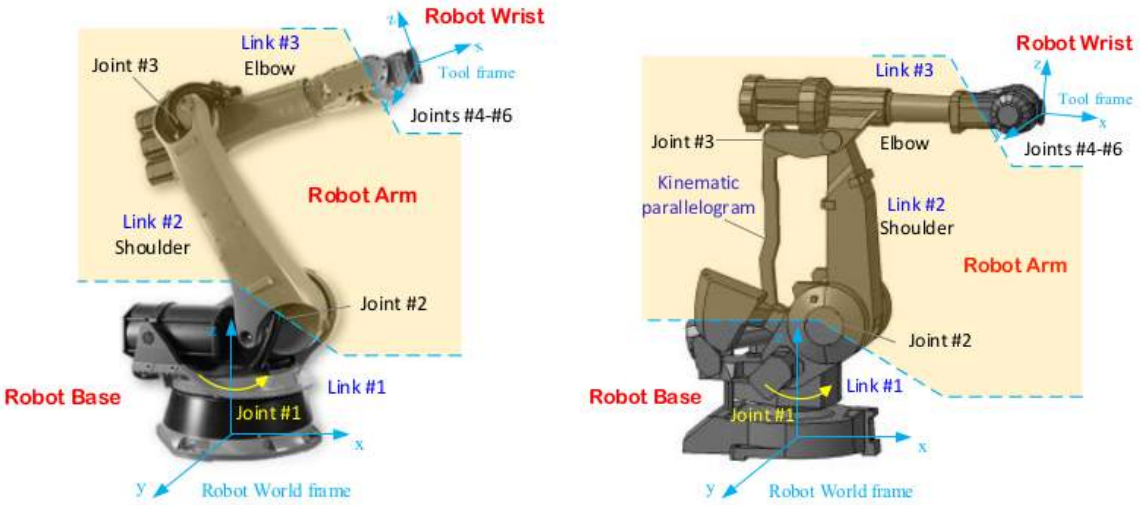


Figura 2-6.: Espacio de trabajo óptimo para ubicación de la pieza [Klimchik et al., 2017]

Adicionalmente, [Kubela et al., 2015] aborda el efecto del error generado por el *backlash* co-

mo un aspecto esencial para alcanzar mejor precisión en robots industriales y propone una ley de compensación implementada mediante un control de velocidad. Para este caso, realimentación de posición y velocidad del robot en tiempo real son requeridas; sin embargo, el progreso alcanzado no es significativo. Y [Atmosudiro et al., 2014] lleva a cabo una comparación de planeación y ejecución de trayectorias en el espacio de articulaciones y en el dominio cartesiano, evidenciando que en el primer caso el TCP alcanza mayor velocidad (debido a menores requerimientos de procesamiento) y en consecuencia ofrece una reducción en tiempos de producción. Sin embargo, esta condición genera mayores fluctuaciones de la herramienta. En contraste con un espacio cartesiano, en el cual el TCP presenta una velocidad casi constante.

Por otra parte, [Klimchik et al., 2016] evalúa la influencia de la arquitecura de un manipulador en la precisión del mecanizado, a partir del error causado por las fuerzas del proceso. Para ello, considera el modelo serial y quasi-serial en robots industriales, donde el primero corresponde a una estructura estándar (Figura **2-7**a), mientras que el segundo contiene una cadena cinemática cerrada adicional (Figura **2-7**b), que incrementa sus propiedades dinámicas. Con lo cual obtiene que los manipuladores quasi-seriales son preferibles para tareas de largas dimensiones al presentar mejor comportamiento en los límites de su espacio de trabajo que un robot serial, los cuales son más convenientes para el caso de medianas y pequeñas dimensiones.



(a) Arquitectura serial (b) Arquitectura quasi-serial

Figura 2-7.: Arquitectura típica de un robot industrial [Klimchik et al., 2016]

Otras aplicaciones de manufactura abordadas incluyen pulido [Liu et al., 2016] y pintura [Chen et al., 2008]. Para el caso de pulido resalta la planeación de trayectorias incluyendo control de fuerza y el uso de su componente normal a la superficie para compensar desviaciones en la pieza. En pintura se presenta una restricción en el movimiento, debido a la necesidad de aplicación uniforme del producto.

2.3. Herramientas de Software

2.3.1. Robot Operating System

ROS es un entorno de trabajo para el desarrollo de aplicaciones robóticas, el cual incluye un amplio conjunto de librerías y herramientas que facilitan la creación de tareas complejas y robustas en una extensa variedad de plataformas robóticas [Lentin, 2015]. Progresivamente diversas instituciones de investigación agregan nuevas opciones de hardware, a la vez que compañías de robótica adaptan sus productos para ser utilizados con ROS [Fernández et al., 2015].

ROS es publicado bajo los términos de licencia BSD y constituye un software de código abierto; basado en arquitectura de grafos con una topología centralizada, cuya filosofía es el desarrollo de herramientas de software que pueden utilizarse en diferentes robots con pequeños cambios de código [Fernández et al., 2015].

La ejecución de ROS se halla ligada a una red donde todos los procesos se encuentran conectados, de forma que cualquier nodo en el sistema puede acceder a cada elemento de dicha red, interactuar con otros nodos, ver la información que se está enviando y transmitir datos [Lentin, 2015].

2.3.2. ROS-Industrial

ROS-I es un proyecto que extiende las capacidades de ROS hacia aplicaciones de manufactura, con la implementación de herramientas y drivers para hardware de tipo industrial, permitiendo incrementar su flexibilidad en gran medida. Lo anterior mediante la inclusión de sistemas de visión 2D y 3D, así como la planeación de movimientos libres de colisiones [Lentin, 2015].

A partir de su comunidad de desarrollo, el conjunto de librerías es mejorado continuamente, considerando los requerimientos funcionales para la industria, tales como robustez, confiabilidad y seguridad [Munaro et al., 2016].

La arquitectura de ROS-I [Lentin, 2015] es presentada en la Figura **2-8**.

- **ROS GUI:** Incluye *plugins* para las herramientas de interfaz gráfica, tales como RViz, rqt_gui, entre otras.
- **ROS-I GUI:** son UI industriales que serán implementadas en el futuro.
- **ROS Layer:** es la capa en la cual se efectúa toda la comunicación.

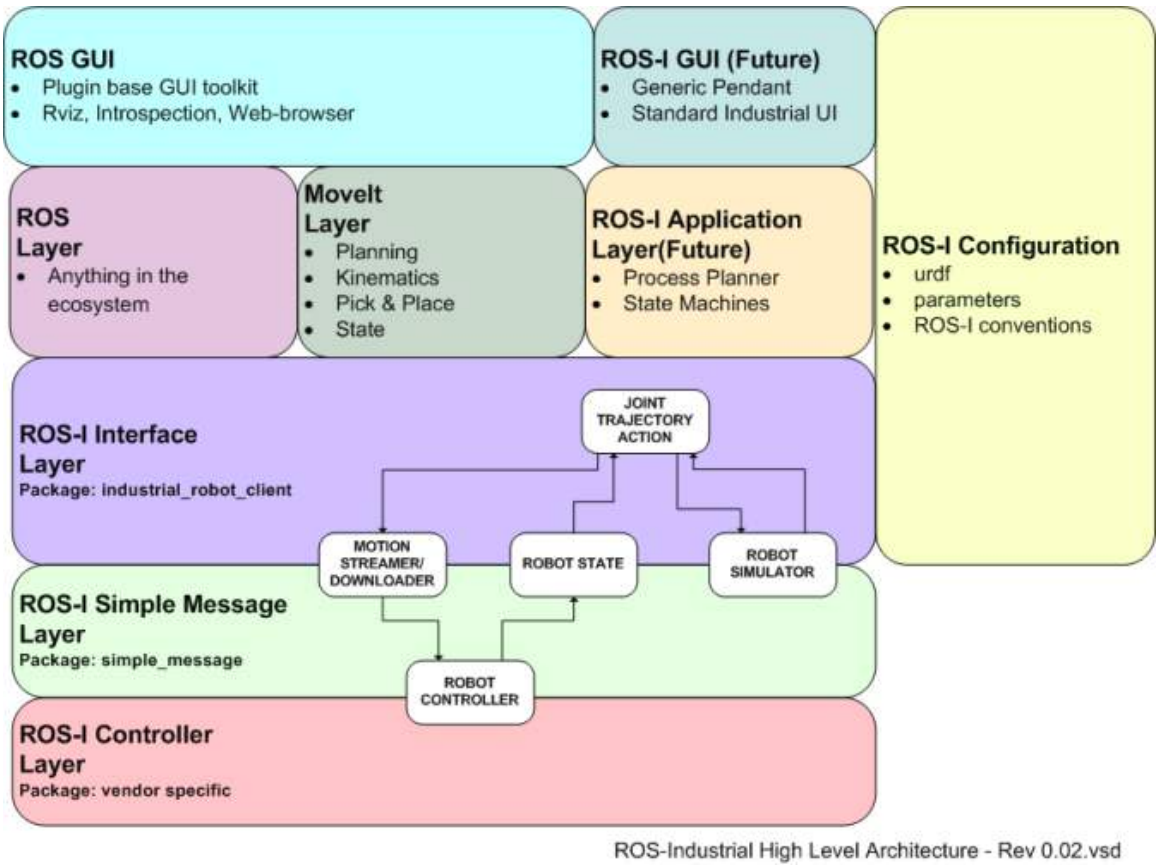


Figura 2-8.: Arquitectura de Ros-Industrial [Lentin, 2015]

- **MoveIt! Layer:** capa que proporciona un conjunto de librerías para cálculos de cinemática, dinámica y planeación de trayectorias del robot.
- **ROS-I Application Layer:** es una funcionalidad que será implementada en el futuro e incluye herramientas para planeación del proceso (qué, cómo, recursos).
- **ROS-I Interface Layer:** corresponde a la aplicación cliente del robot industrial, la cual se conecta al controlador a través de un protocolo de mensajes simples.
- **ROS-I Simple Message Layer:** es la capa de comunicación del robot industrial, la cual consiste de un protocolo que permite el intercambio de información bidireccional entre la aplicación cliente y el controlador.
- **ROS-I Controller Layer:** hace referencia al hardware del controlador específico.

2.3.3. MotoROS

MotoROS es un driver creado a través del software *MotoPlus* e instalado en el controlador del robot, el cual permite que un nodo de ROS-I comande directamente el movimiento

de un robot Motoman, facilitando el envío de instrucciones incrementales a una alta tasa [Motoman, 2017].

Inicialmente a través de ROS es definida la trayectoria del robot, a partir de lo cual ROS-I se encarga de la generación de los puntos de dicha trayectoria para enviarlos posteriormente a través de TCP/IP hacia MotoROS en el controlador del robot. Una vez es recibido cada punto, una confirmación se transmite al PC ejecutando ROS, informando que se encuentra listo para la transmisión de los siguientes puntos de la trayectoria, a la vez que comanda el movimiento del robot [Motoman, 2017]. La Figura **2-9** ilustra la arquitectura de MotoROS.

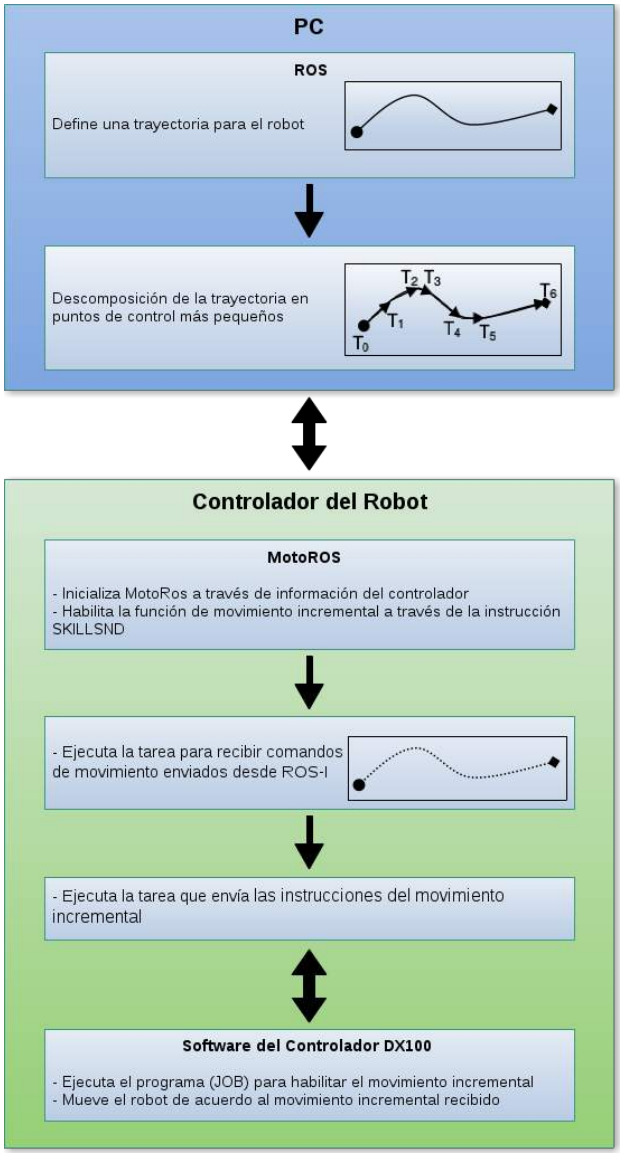


Figura 2-9.: Arquitectura de MotoROS. Adaptado de [Motoman, 2017]

2.3.4. Secuencia de comunicación

Esta sección es descrita a partir de [Motoman, 2017], con la inclusión de algunas modificaciones requeridas para su correcta implementación con el sistema robótico objeto del presente trabajo de investigación.

La secuencia de comunicación inicia con el encendido del controlador y posterior ejecución de la tarea principal (*Main task*), con lo cual se crea una instancia de la estructura del controlador a partir de la lectura de parámetros y de información adicional correspondiente a la configuración del equipo. Tras lo anterior, se ejecuta la tarea *Connection Server*, que mantiene a la aplicación en un ciclo infinito para monitorear los estados del controlador (alarmas, errores, estado de servomotores, etc). La Figura **2-10** presenta la operación de la tarea principal y su interacción con el controlador.

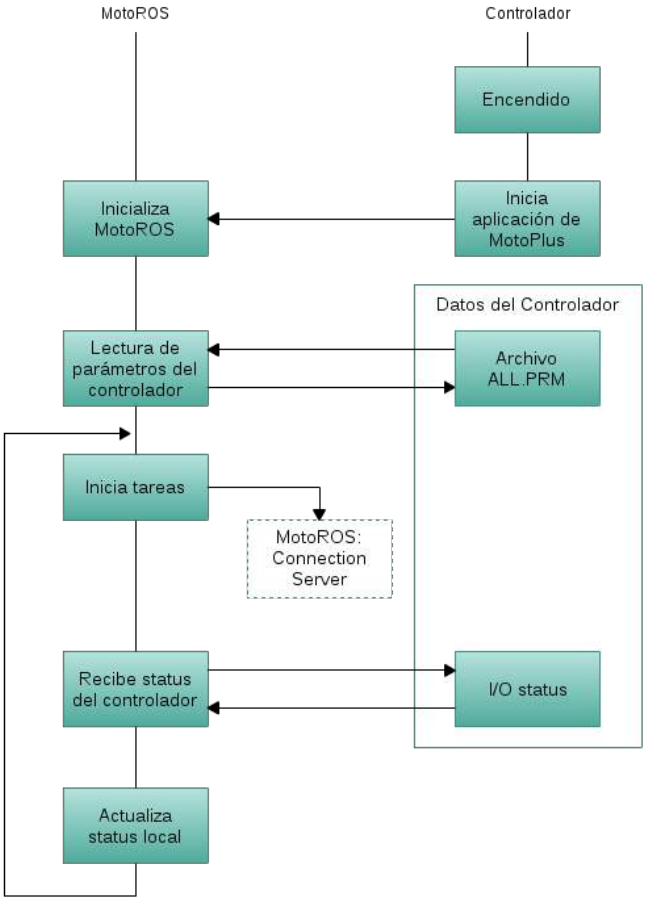


Figura 2-10.: Ejecución de la tarea principal al encendido. Adaptado de [Motoman, 2017]

Al ejecutarse la tarea *Connection Server*, son habilitados dos puertos de comunicación y el controlador espera a que una aplicación cliente se conecte a cualquiera de estos. El primero

de ellos corresponde al puerto de estado (50241) proporcionando la conexión que permite leer la posición actual del robot, así como sus señales de estado. Por otra parte; con el fin de comandar movimientos, el PC debe establecer una segunda conexión TCP/IP hacia el puerto 50240. Cuando alguna conexión es detectada, una nueva tarea se ejecuta de acuerdo con el puerto de comunicación correspondiente. La Figura **2-11** ilustra la ejecución de la tarea *Connection Server*.

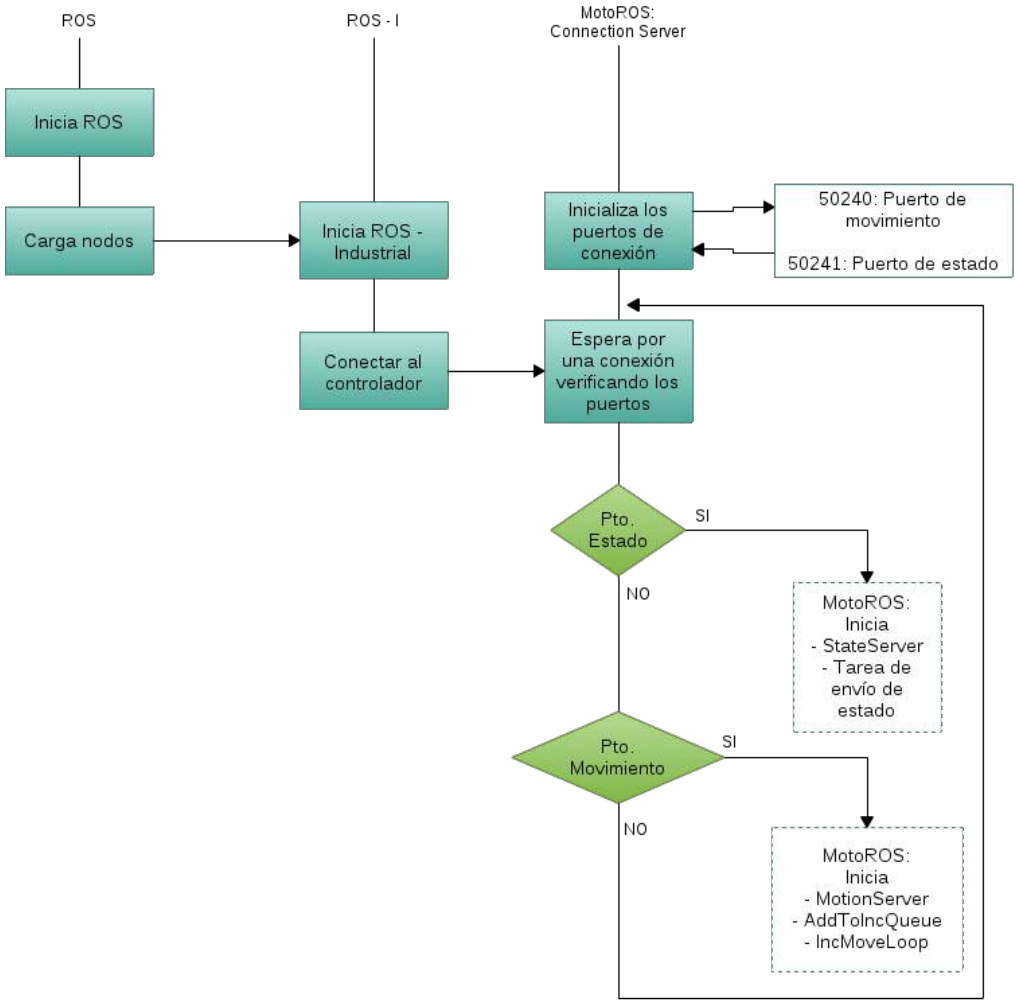


Figura 2-11.: Ejecución de la tarea *Connection Server*.Adaptado de [Motoman, 2017]

Si la conexión es efectuada con el puerto de estado del controlador, entonces la tarea *State Server* es iniciada y ejecutada de acuerdo a la Figura **2-12**. Esta tarea recibe la información correspondiente a la posición actual y estados del robot (suministrados por el controlador) y los transmite hacia el PC a través de ROS-I para su respectiva visualización.

Desde el momento en el cual la comunicación es establecida, se inicia de manera automática

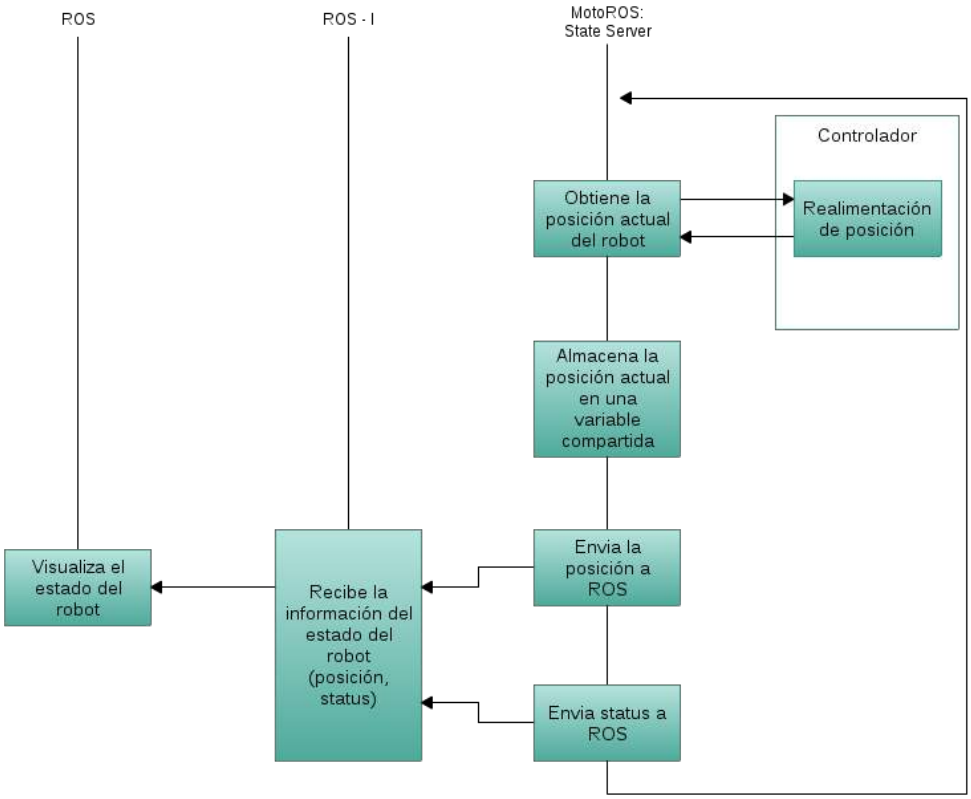


Figura 2-12.: Ejecución de la tarea *State Server*. Adaptado de [Motoman, 2017]

el envío de información a un intervalo de tiempo regular, lo cual permite que la aplicación cliente acceda a los datos cuando lo requiera, sin necesidad de realizar alguna petición. Es posible entonces que existan diferentes clientes recibiendo los datos; sin embargo, exisitirá una única instancia del servidor.

Por otra parte, cuando se presenta una conexión hacia el puerto de movimiento, una serie de tareas son ejecutadas. La primer tarea es *Motion Server*, que se inicia con cada conexión al puerto, teniendo en cuenta que no es posible que múltiples conexiones intenten comandar el mismo grupo de control.

Esta tarea espera por un mensaje de movimiento proveniente de la aplicación cliente, para posteriormente identificar su tipo y propósito, y de esta manera efectuar la acción correspondiente. Básicamente, los mensajes enviados para incluir puntos a la trayectoria serán inicialmente validados y dirigidos hacia una cola de movimientos incrementales mientras que otros que afectan el controlador (I/O, control de servomotores, etc) son procesados inmediatamente a través de un hilo que recibe el mensaje.

Si la cola de movimientos incrementales se encuentra llena, la respuesta puede presentar retraso. Lo anterior genera que posteriores mensajes no sean procesados inmediatamente e instrucciones que detengan el movimiento del robot podrían no funcionar apropiadamente y generar inconvenientes de operación y posibles situaciones de peligro. Por ello, el mensaje es validado y el punto de la trayectoria copiado temporalmente para ser procesado por una tarea en segundo plano (*AddToIncQueueProcess*), manteniendo disponible la comunicación con el *socket*. Cabe señalar que una respuesta exitosa en este instante corresponde solamente a que el mensaje ha sido aceptado, pero no que el movimiento ha sido completamente procesado. Ver Figura **2-13**.

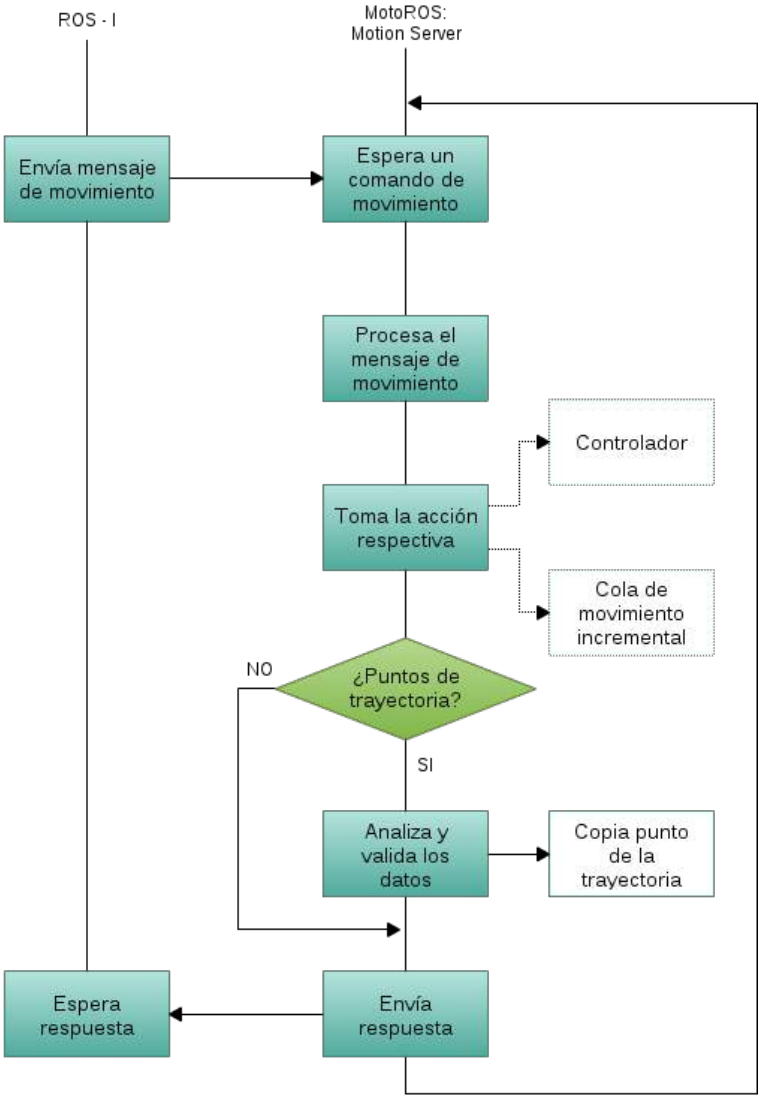


Figura 2-13.: Ejecución de la tarea *Motion Server*. Adaptado de [Motoman, 2017]

La tarea *AddToIncQueueProcess* presentada en la Figura **2-14** se ejecuta en segundo plano,

y es la encargada de dividir la trayectoria en movimientos incrementales a partir de una segmentación dependiente del período de interpolación del controlador. Dicho período varía dependiendo de la configuración del controlador; tal que para un único robot sin ejes externos es de 4 milisegundos. Sin embargo, en este caso debido a los ejes adicionales el período se incrementa aproximadamente a 8 milisegundos.

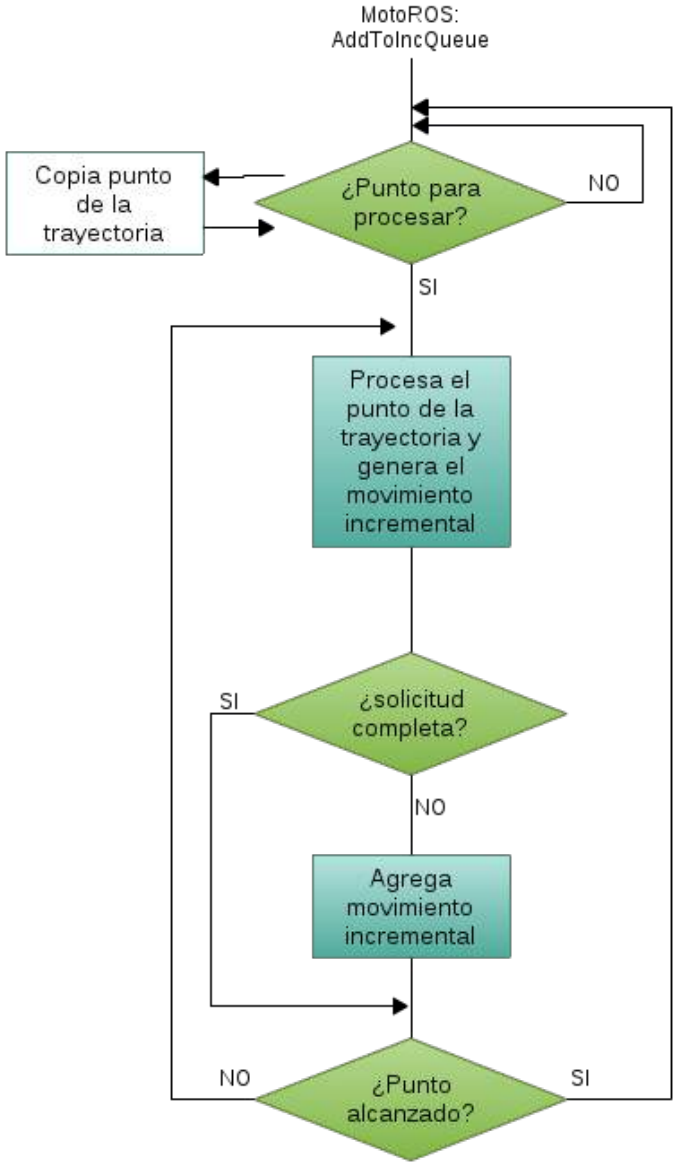


Figura 2-14.: Ejecución de la tarea *AddToIncQueueProcess*. Adaptado de [Motoman, 2017]

Finalmente, la tarea de movimiento incremental (*IncMoveLoop*) en la Figura **2-15** se sincroniza con el reloj interno del controlador y envía el movimiento correspondiente para cada período de interpolación (8ms). Es esta tarea la que establece la velocidad de movimiento

correspondiente.

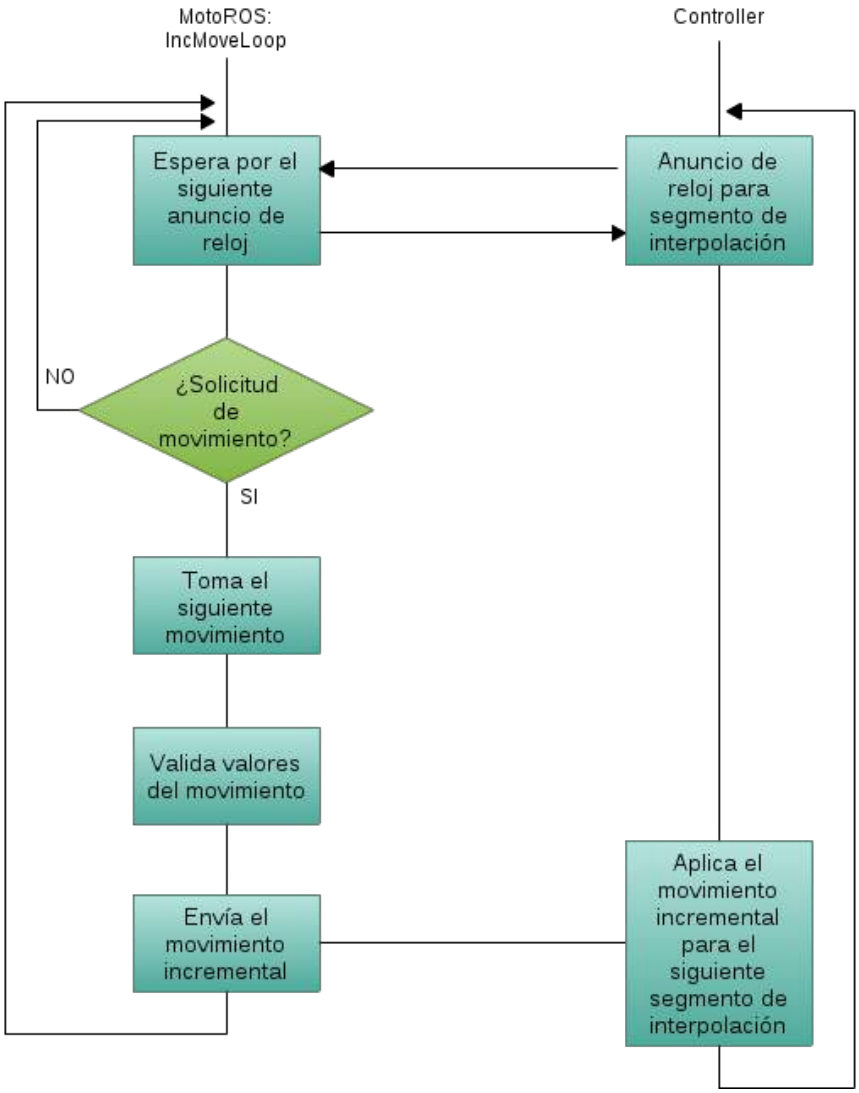


Figura 2-15.: Ejecución de la tarea *IncMoveLoop*. Adaptado de [Motoman, 2017]

2.3.5. Rosbridge

Rosbridge funciona como una capa de abstracción de software (*middleware*) que facilita el manejo y programación de aplicaciones distribuidas, proporcionando un acceso simple a mensajes y servicios en ROS a partir de sockets, mediante el uso de objetos serializados usando JSON (ver Figura **2-16**), los cuales son definidos a partir de una sintaxis básica de Javascript [Crick et al., 2017]. De esta manera, *rosbridge* permite que complejos cálculos computacionales sean distribuidos en múltiples máquinas, así como la creación de interfaces de usuario remotas mediante su capa de aplicación en Javascript (*rosjs*), para interacción a

través de los navegadores web modernos [Toris et al., 2014].

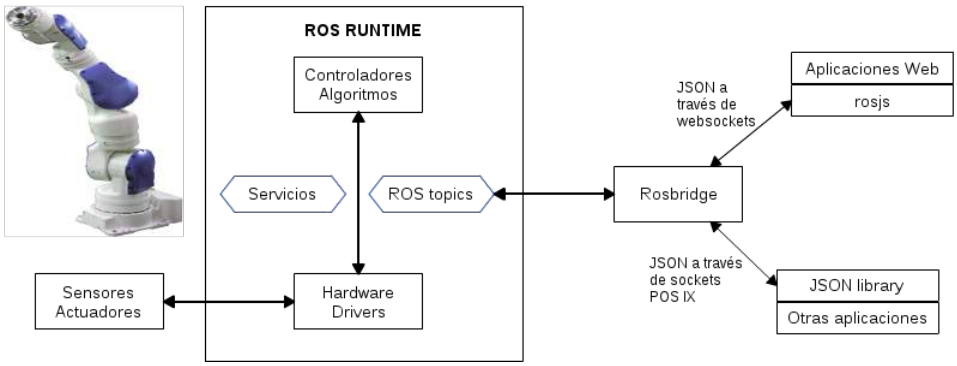


Figura 2-16.: Serialización de servicios y *topics* en ROS [Crick et al., 2017]

Puesto que JSON es bastante simple la serialización puede ser construida incluso manualmente. Sin embargo, existen diferentes librerías que permiten llevar a cabo este tarea de forma más fácil y menos propensa a errores [Crick et al., 2017]. A continuación se presenta el ejemplo de codificación de un mensaje de ROS en JSON para publicar un mensaje de velocidad en el *topic /cmd_vel* [Toris et al., 2015]:

```
{ "op": "publish",
  "topic": "/cmd_vel",
  "type": "geometry_msgs/Twist",
  "msg": {"linear":{"x":1.0,"y":0,"z":0},
          "angular":{"x":0,"y":0,"z":0}
}
```

Diversas investigaciones han sido realizadas a través de *rosbridge*. [Toris et al., 2015] presenta el proyecto *Robot Web Tools* (RWT) cuyo objetivo es integrar plataformas robóticas como ROS con tecnologías web modernas para facilitar la creación de aplicaciones e interfaces hombre-máquina en la nube. Específicamente aborda el desarrollo de herramientas para la transmisión de mensajes complejos de ROS utilizando menor ancho de banda. Entre estos se incluyen video y nube de puntos.

Por otra parte, una aproximación al uso de dispositivos móviles para permitir la conexión con software robótico es descrita en [Codd-Downey and Jenkin, 2015]. Para ello, ha desarrollado una aplicación inspirada en herramientas de visualización de ROS con la posibilidad de enviar funciones de comando a través de interfaces gráficas y de la captura de información inercial (mediante acelerómetro y giroscopio [Speers et al., 2013]).

Con el creciente uso de esta tecnología, se torna crucial el tema de seguridad en los sistemas. Debido a lo anterior, [Toris et al., 2014] propone el uso de *web tokens* para garantizar

autenticación de clientes remotos que se conectan a ROS.

Adicionalmente, el diseño e implementación de una herramienta de realidad aumentada (AR) se aborda en [Makris et al., 2016], en un entorno industrial colaborativo. Este sistema asiste al operador durante un ensamble, proporcionando control del proceso, instrucciones inmersivas e incluyendo visualización de los componentes requeridos y la trayectoria definida para el robot.

2.3.6. Visualization Tool Kit

VTK es una herramienta multi-plataforma de código abierto para el procesamiento, visualización y análisis de datos científicos. Su código ha sido reescrito para aprovechar las funcionalidades de las tarjetas gráficas modernas, alcanzando un incremento significativo en las capacidades de procesamiento y manteniendo la mayoría de sus interfaces de programación [Hanwell et al., 2015].

Esta librería proporciona un rápido acceso a diferentes algoritmos de renderizado, así como a componentes para visualización e interacción con los resultados. Lo anterior ha constituido una base para el desarrollo de diversas aplicaciones en los campos de imágenes médicas, química, computación gráfica, dinámica de fluidos y análisis de elementos finitos [O'Leary et al., 2017].

[Zhu et al., 2015] propone una nueva tecnología en el modelamiento virtual de entornos tridimensionales mediante el uso de VTK para teleoperación de robots. Mientras que en la investigación realizada por [Tan et al., 2016] se introduce el uso de técnicas de diseño y visualización en la reconstrucción de imágenes médicas.

2.4. Robótica colaborativa

La planeación de trayectorias incluyendo múltiples robots es un tema de investigación de gran importancia al incluir aspectos de evasión de colisiones y coordinación de los movimientos. Adicionalmente, la interacción entre robots para tareas específicas incrementa la productividad y versatilidad de aplicaciones complejas. Lo anterior conlleva a un continuo aumento en el uso de celdas de robótica colaborativa [Wu et al., 2016].

[Larsen et al., 2015] propone una aproximación hacia procesos de ensamble en la producción de fuselaje para aviones utilizando dos manipuladores industriales. Para ello, plantea estrategias de parametrización de la trayectoria y la definición del material como elemento

maestro, para las posiciones de cada robot.

De la misma manera [Wagner et al., 2016] describe la realización de un proceso de escaneo utilizando dos robots. La idea básica consiste en la planeación de los movimientos considerando que uno de los robots desplaza el sensor, mientras que el otro manipula la pieza correspondiente.

En la literatura acerca de aplicaciones de robótica colaborativa resalta actualmente la interacción entre manipuladores industriales que trabajan de forma conjunta, o grupos de AGV's coordinados para efectuar una tarea. Sin embargo, no se evidencian trabajos de investigación relevantes en los cuales se incluyan robots industriales y móviles simultáneamente en aplicaciones cooperativas. Específicamente, [Ferrati et al., 2016] aborda la manipulación de objetos mediante robots heterogéneos (industrial y móvil) para su uso en fábricas inteligentes. Sin embargo, su enfoque es la optimización de trayectorias para cada robot con áreas de trabajo definidas para la transición del producto.

3. Materiales y Métodos

Este trabajo es llevado a cabo en el Laboratorio Sala CAM - LabFabEx de la Universidad Nacional de Colombia, el cual tiene a su disposición diversas máquinas de nivel industrial y académico, así como algunas herramientas de software de gran utilidad en el proyecto.

3.1. Sistema robótico

Para el desarrollo de esta investigación se hace uso de un sistema robótico compuesto por el robot Yaskawa Motoman MH6, incluyendo dos ejes externos: guía lineal y posicionador rotacional. La Figura **3-1** presenta los elementos que conforman el sistema mencionado. Adicionalmente, en el Anexo A se presenta una descripción de las articulaciones y ejes cartesianos del sistema.

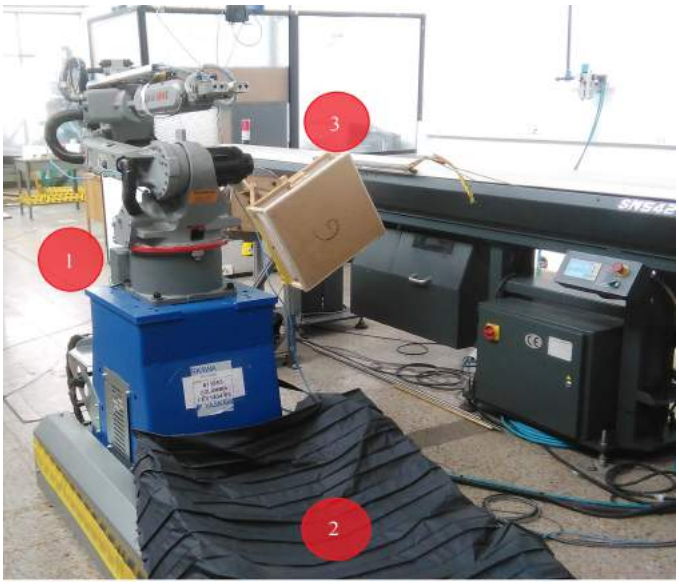


Figura 3-1.: Sistema robótico: 1) Robot, 2) Guía lineal, 3) Posicionador rotacional

Debido al trabajo de investigación continuado de tres años, se dispone de una red de comunicación para realizar teleoperación de las diferentes máquinas a través de Internet. Específicamente en lo correspondiente al controlador (DX100) del robot se ha configurado un esquema de red que se presenta en la Figura **3-2**. En éste se incluye un servidor web que proporciona el acceso de forma remota a la plataforma del laboratorio, y que transmite a

su vez la información enviada por el usuario hacia un PC dedicado al robot que finalmente comanda la operación de la máquina. Lo anterior facilita la inclusión de la plataforma obtenida mediante ROS para comunicarse con el controlador del robot a través de Ethernet o de una conexión inalámbrica en el laboratorio.

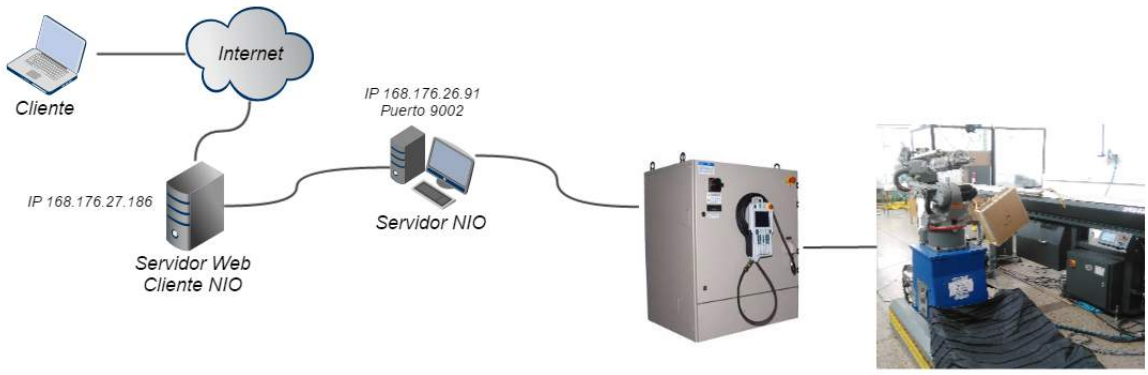


Figura 3-2.: Esquema de comunicación inicial del sistema

3.1.1. Robot Yaskawa Motoman MH6

El robot Motoman MH6 es un manipulador serial [Klimchik et al., 2016] de tipo industrial que cuenta con seis articulaciones rotacionales (6 DOF). Éste ha sido dispuesto para aplicaciones de propósito general, y sus especificaciones son registradas en la Tabla **3-1**.

Tabla 3-1.: Especificaciones del robot Motoman

Carga máxima		6 kg
Alcance vertical		2486 mm
Alcance horizontal		1422 mm
Repetibilidad		± 0.08 mm
Rango máximo de movimiento	Eje - S	$\pm 170^o$
	Eje - L	$+155^o / -90^o$
	Eje - U	$+250^o / -175^o$
	Eje - R	$\pm 180^o$
	Eje - B	$+225^o / -45^o$
	Eje - T	$\pm 360^o$
Máxima velocidad	Eje - S	$220^o/s$
	Eje - L	$200^o/s$
	Eje - U	$220^o/s$
	Eje - R	$410^o/s$
	Eje - B	$410^o/s$
	Eje - T	$610^o/s$

El rango de movimiento para cada una de las articulaciones del robot define su espacio de trabajo alcanzable de acuerdo con la Figura **3-3**.

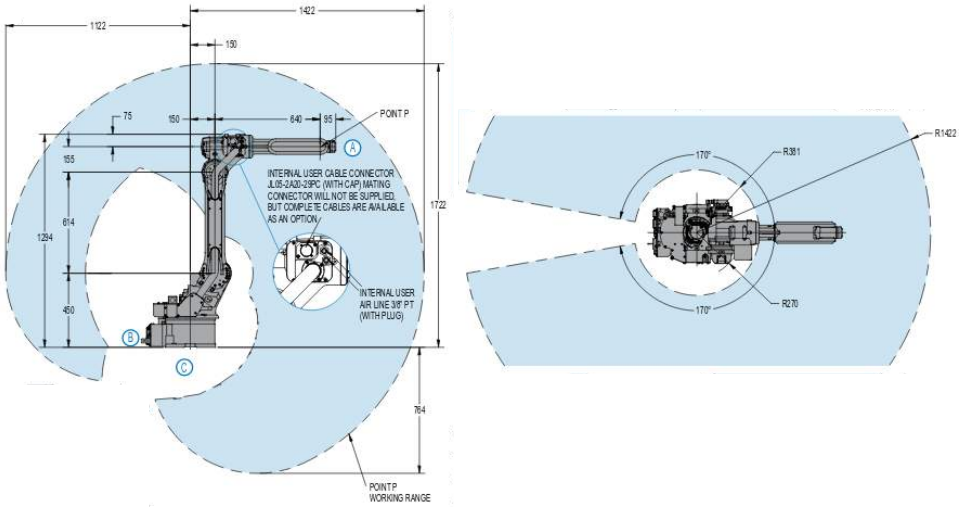


Figura 3-3.: Espacio de trabajo alcanzable del robot [Motoman, 2009]

3.1.2. Guía lineal TSL1000

La guía lineal TSL1000 de Yaskawa es un dispositivo de fijación al suelo, que permite desplazar linealmente el robot hasta una longitud de tres metros a lo largo del laboratorio, extendiendo de esta manera su alcance. En la Tabla **3-2** se describen las especificaciones técnicas correspondientes.

Tabla 3-2.: Especificaciones de la guía lineal.

Carga máxima	1000 kg
Desplazamiento	3000 mm
Velocidad máxima	0.72 m/s
Repetibilidad	± 0.05 mm

3.1.3. Posicionador rotacional

El posicionador rotacional es un dispositivo que permite girar la pieza de trabajo para facilitar el acceso del robot a toda su geometría. Este elemento ha sido diseñado y fabricado como

parte de algunos proyectos académicos de investigación realizados previamente, y en los cuales se han establecido las especificaciones funcionales requeridas, tal como se encuentran relacionadas en la Tabla **3-3**.

Tabla 3-3.: Especificaciones del posicionador rotacional.

Rango de movimiento	$\pm 360^o$
Velocidad máxima	1.2 rad/s
Carga axial máxima	2500 N
Carga radial máxima	1500 N

El conjunto conformado por el robot y los dos ejes externos específicos constituyen un sistema robótico único, altamente flexible y versátil, con el potencial para desarrollar aplicaciones de gran complejidad.

3.2. Modelo de comunicación PC - Controlador

Como herramienta de software para el desarrollo de este trabajo se hace uso de ROS junto con ROS-I, para llevar a cabo la comunicación entre PC y Controlador DX100 a través de Ethernet. Esto facilita que un nodo de ROS pueda comandar los movimientos del sistema robótico a la vez que recibe información acerca de la posición actual de cada articulación y las señales de estado correspondientes.

Debido a que el sistema robótico utilizado presenta una configuración que no ha sido abordada previamente a través de ROS, el objetivo de alcanzar una comunicación satisfactoria entre éste y el controlador (cuando se incluyen los ejes externos), conlleva a una etapa exhaustiva para establecer solución a las incompatibilidades surgidas en la interacción del modelo virtual y el real. Específicamente, ya que no se dispone de ningún tipo de documentación que permita abordar el problema de comunicación, la depuración de cada falla evidenciada se hace a partir de exploración experimental hasta obtener el resultado deseado.

Finalmente se hace necesario efectuar modificaciones en el código fuente tanto de la aplicación MotoROS creada por Yaskawa y la cual se ejecuta de fondo en el controlador, así como del driver de Motoman para ROS desarrollado por ROS-I.

En la Figura **3-4** se ilustra el procedimiento realizado en esta investigación para obtener un modelo de comunicación funcional.

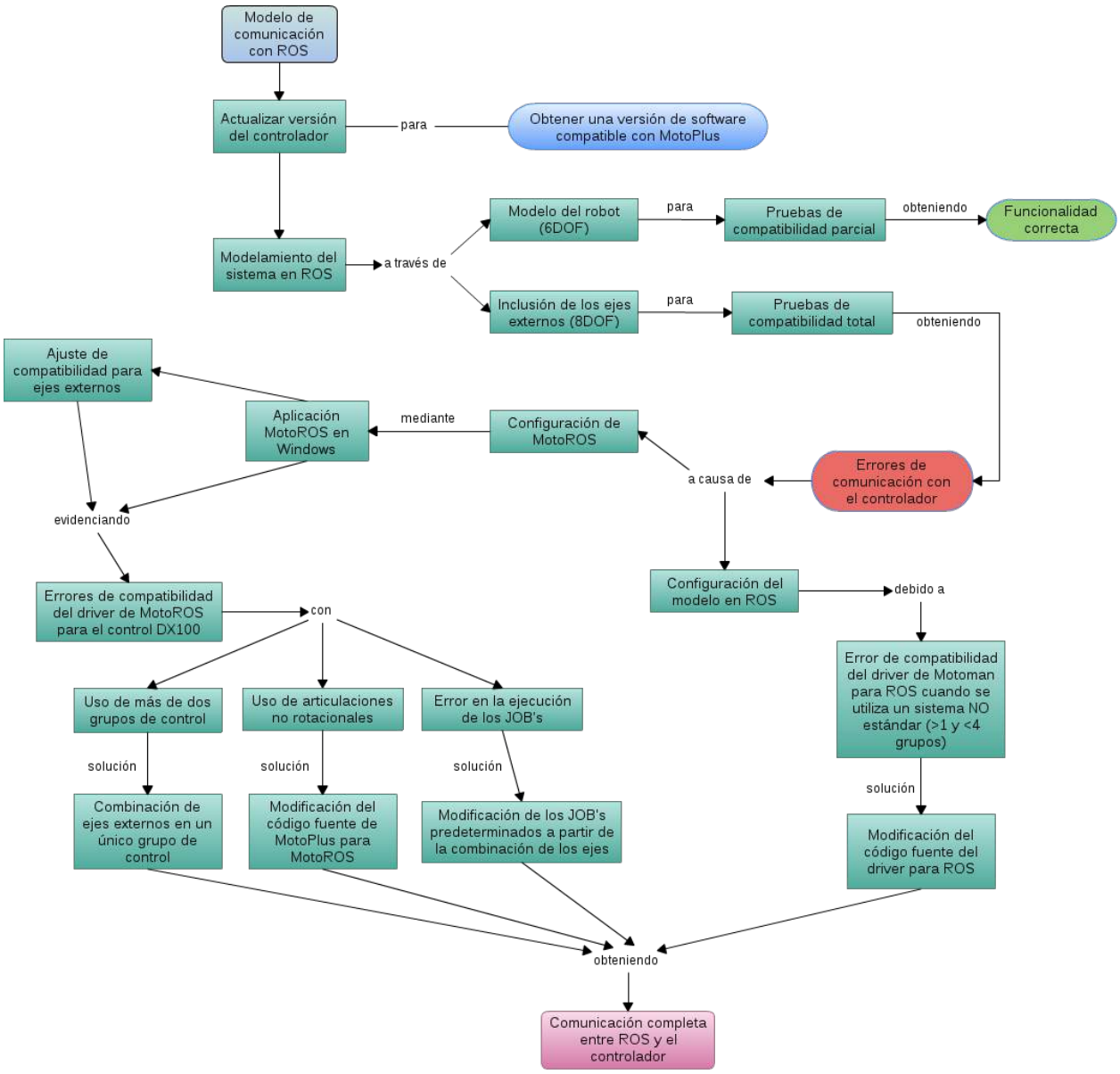


Figura 3-4.: Metodología para el desarrollo del modelo de comunicación

3.2.1. Actualización de la versión del controlador

MotoROS es una aplicación creada mediante el software de desarrollo de Motoman, MotoPlus SDK. Por lo tanto para poder instalarla en el controlador DX100 del robot, se requiere que éste sea compatible con archivos de MotoPlus. Para ello la versión de sistema del equipo debe ser **DS3.91.00-14** o superior.

Debido a que inicialmente la versión del controlador corresponde a **DS1.58.00A-00**, se hace necesario llevar a cabo una actualización hacia la versión **DS3.97.00A-14** que es compatible

con MotoPlus, aunque ésta no se encuentra habilitada por defecto. El Anexo B presenta la actualización en la versión del software y la habilitación de esta funcionalidad.

3.2.2. Modelamiento del sistema robótico en ROS

Existe una amplia librería de robots que pueden utilizarse para la creación de diferentes aplicaciones en ROS. No obstante, los componentes de este sistema en particular no se encuentran disponibles.

Se plantea inicialmente realizar el modelo únicamente con los seis ejes del brazo, obteniendo un entorno virtual VMD que permita la interacción con dichos ejes. Para ello, es definido el modelo URDF [Lentin, 2015], el cual contiene la información geométrica que describe el robot. De igual manera se incluyen modelos 3D para efectos visuales y de colisión: el primero es usado solamente para presentación en la interfaz gráfica; y debido a que no interviene en la validación de colisiones, éste puede ser altamente detallado. Por su parte, el segundo conjunto de modelos no deben incluir gran nivel de detalle, pues a medida que su complejidad incrementa requiere mayor tiempo de procesamiento para validar posibles colisiones, éste debe ser sencillo pero incluir toda la geometría del robot. Para este caso en particular se han utilizado los mismos modelos en la componente visual y en la de colisiones.

Posteriormente, es necesario configurar los archivos definidos para el controlador virtual, de forma que éste pueda establecer una apropiada comunicación con el controlador del robot. Para probar esta representación inicial, es necesario deshabilitar los ejes externos en la configuración del equipo pues se genera una incompatibilidad entre la información procesada por el PC (6 ejes) y aquella en el controlador del robot (8 ejes). De esta manera es posible comandar satisfactoriamente movimientos desde la aplicación de ROS directamente hacia el robot y realimentar la posición del modelo virtual.

Debido a las pruebas experimentales de movimiento realizadas y aunque los rangos para cada articulación han sido definidos de acuerdo a la Tabla **3-1**, se evidencia que los ejes U y B pueden presentar interferencia mecánica con accesorios ubicados en el robot, por lo cual los correspondientes límites son entonces redefinidos de acuerdo a la Tabla **3-4**.

Una vez se ha implementado correctamente la interfaz para el brazo robótico, se procede a incluir los dos ejes externos que completan el sistema.

Para la integración correspondiente al primer eje externo, se define una cadena cinemática abierta que comprende desde el marco de referencia asociado a la base de la guía lineal, hasta el TCP del robot. Lo anterior facilita el uso del plugin *Track-IK* [Beeson and Ames, 2015]

Tabla 3-4.: Límites de movimientos para el robot virtual

Rango máximo de movimiento	Eje - S	$\pm 170^o$
	Eje - L	$+155^o / -90^o$
	Eje - U	$+130^o / -175^o$
	Eje - R	$\pm 180^o$
	Eje - B	$+187^o / -30^o$
	Eje - T	$\pm 360^o$

para estos siete ejes. Sin embargo, en el caso del posicionador no es posible implementar esta librería debido a que no contiene ninguna cadena cinemática, puesto que se encuentra conformado únicamente por un eje.

El modelo virtual del sistema robótico se presenta en la Figura **3-5**. Para este caso, con el interés de obtener coincidencia entre la ubicación virtual y la posición física de los elementos dispuestos en el laboratorio, la guía lineal junto con el robot (definidos como el grupo *arm_on_rail*) son fijados al origen de la representación, y el posicionador rotacional (grupo *station*) se ubica espacialmente utilizando el propio robot como instrumento de medición. Adicionalmente, son validadas las direcciones de giro positiva y negativa para cada uno de los ocho ejes.

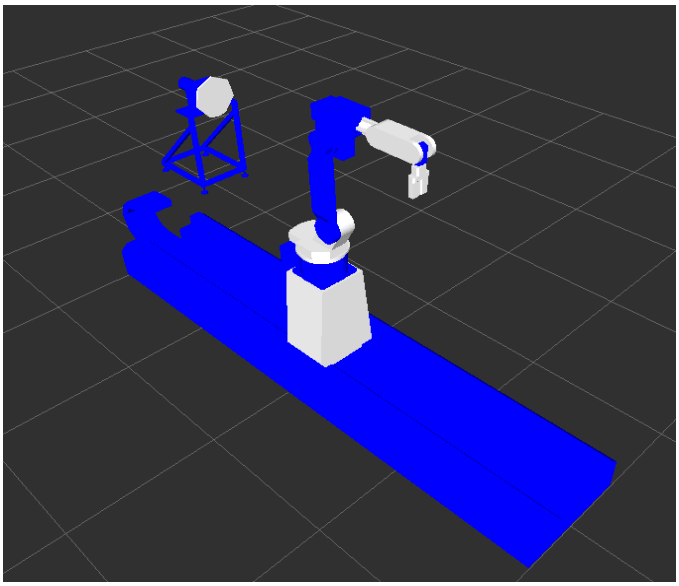


Figura 3-5.: Modelo virtual del sistema robótico

Al intentar establecer comunicación entre el modelo obtenido y el controlador del robot, se experimentan errores de compatibilidad que rechazan los intentos de conexión. Debido a que

no se encuentran reportes de situaciones similares, es contactado directamente personal de ROS-I y posteriormente *Yaskawa America*.

Por parte de *Yaskawa Motoman* es proporcionada una aplicación para Windows que permite emular el comportamiento de un cliente ROS que intenta conectarse a los puertos de estado y de movimiento en el controlador de acuerdo con la Figura **2-11**.

3.2.3. Aplicación cliente de MotoROS en Windows

La aplicación cliente es utilizada con el fin de validar el funcionamiento de MotoROS ejecutándose en el controlador del robot. La interfaz del emulador se presenta en la Figura **3-6**. Ésta permite definir la dirección IP del equipo al cual se desea conectar, así como el modelo del robot correspondiente. Cabe señalar que el modelo particular del robot real (MH6) no se encuentra disponible, pero a partir de la configuración de un robot similar (MH5), se crea una opción compatible para este caso. Adicionalmente, incluye la posibilidad de conexión independiente a cada uno de los dos puertos involucrados en la comunicación.

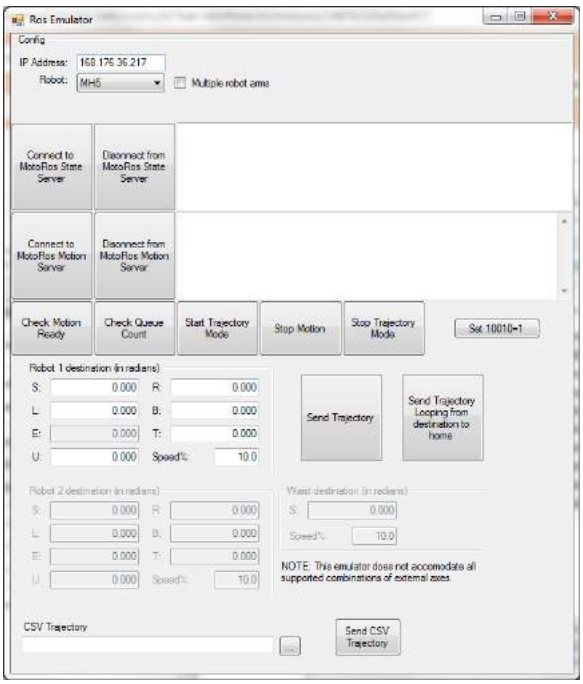


Figura 3-6.: Emulador de ROS para Windows

Las pruebas efectuadas permiten (al igual que con el modelo directo de ROS) establecer comunicación con el controlador cuando son deshabilitados los ejes externos, pero presenta errores al utilizar el sistema completo (8DOF). Lo anterior indica que la aplicación de MotoROS no permite la conexión al controlador, por lo cual ésta debe ser depurada.

3.2.4. MotoROS: Control de los ejes externos

Mediante pruebas realizadas con el apoyo de *Yaskawa America* se evidencia que el controlador DX100 presenta algunas deficiencias en su interacción con la aplicación MotoROS. Específicamente el modelo DX100 permite incluir hasta dos grupos de control, donde estos grupos hacen referencia a cada elemento configurado. Debido a que el sistema dispone de tres elementos (Robot: R1, Guía lineal: B1 y Posicionador: S1), su definición en el controlador debe ser modificada para superar esta restricción.

A partir de lo anterior; los dos ejes externos son combinados en un único grupo, donde ahora S1 es un dispositivo de dos ejes (guía lineal + posicionador). Debido a esta reconfiguración, el controlador considera que el posicionador ha sido ubicado sobre la guía lineal y se desplaza con ella, mientras que el robot se encuentra fijo. Sin embargo, esta situación no presenta inconveniente al utilizarse con ROS (o de cualquier forma en la que el cálculo del movimiento no sea efectuado en el controlador), ya que la cinemática inversa es obtenida de forma independiente y posteriormente transmitida la posición para cada articulación.

Combinar los ejes externos permite que el emulador establezca conexión con los puertos de estado y movimiento apropiadamente. No obstante, se evidencia ahora que no es posible obtener la posición actual de la guía lineal. A partir de una verificación en el comportamiento de la tarea *state server* (Figura **2-12**) resalta el hecho que MotoROS ha sido concebido para utilizarse en conjuntos conformados únicamente por articulaciones rotacionales. La existencia de la guía lineal involucra una incompatibilidad con la aplicación. Debido a esta situación es necesario efectuar una modificación en el código fuente de MotoROS, a fin de considerar el uso de articulaciones no rotacionales.

Finalmente, la conexión a los puertos de comunicación es realizada exitosamente y la lectura de la posición para cada articulación es correcta. Sin embargo, al intentar comandar un movimiento del sistema robótico, el controlador activa los servomotores pero la trayectoria no es ejecutada ni por el robot ni por los ejes externos. La solicitud retorna al emulador un error debido a que las señales de estado del robot indican que éste no se encuentra listo para efectuar un movimiento.

Por protocolo de seguridad de Motoman, la activación de estas señales se obtiene mediante la ejecución (en modo automático) de una rutina almacenada en el controlador, la cual debe especificar el grupo que desea comandar (R1 o S1). Por esta razón es creado un programa para cada grupo (como se muestra en la Figura **3-7**), habilitando cada elemento para moverse e informando esta condición a MotoROS mediante la instrucción *SKILLSND*.

En la Figura **3-8**, la activación de la salida $OUT\#891$ indica una inicialización correcta

```
JOB CONTENT: MASTER
J:ROS_R1                                S:0000
CONTROL GROUP: R1                       TOOL: **
0000 NOP
0001 DOUT OT#(890) OFF //ROS_DONE
0002 DOUT OT#(889) OFF //ROS_READY
0003 TIMER T=0.05
0004 SKILLSND "ROS-START"
0005 TSYNC 1
0006 DOUT OT#(889) ON //ROS_READY
0007 WAIT OT#(890)=ON //ROS_DONE
0008 DOUT OT#(889) OFF //ROS_READY
0009 DOUT OT#(890) OFF //ROS_DONE
0010 SKILLSND "ROS-STOP"
0011 END
```

```
JOB CONTENT: MASTER
J:ROS_S1                                S:0000
CONTROL GROUP: S1                       TOOL: **
0000 NOP
0001 DOUT OT#(889) OFF //ROS_READY
0002 DOUT OT#(890) OFF //ROS_DONE
0003 TIMER T=0.05
0004 SKILLSND SL6 "ROS-START"
0005 TSYNC 1
0006 WAIT OT#(890)=ON //ROS_DONE
0007 SKILLSND SL6 "ROS-STOP"
0008 END
```

Figura 3-7.: Programas para R1 y S1

de MotoROS a partir de la ejecución de la tarea principal (*Main task* de la Figura **2-10**) y el inicio de la tarea *Connection Server* (Figura **2-11**) para monitorear la conexión a los puertos. Una vez la comunicación con los dos servidores es correcta, las señales $OUT\#893$ y $OUT\#894$ son activadas para comandar movimientos al robot y realizar lectura de estados respectivamente.

Figura 3-8.: Señales de status del controlador

Una vez el sistema robótico interactúa correctamente con el emulador, la aplicación obtenida de MotoROS es funcional, por lo cual resta configurar el lado de ROS que se ejecuta en el PC.

3.2.5. Error de comunicación entre ROS y el controlador del robot

La conexión desde el PC corriendo ROS hacia el controlador del robot se establece inicialmente a través del puerto de estado (50241); lo cual activa la ejecución de la tarea *State Server*. A partir del monitoreo de la actividad en la red mediante el uso de *Wireshark* se

obtiene que esta etapa funciona correctamente.

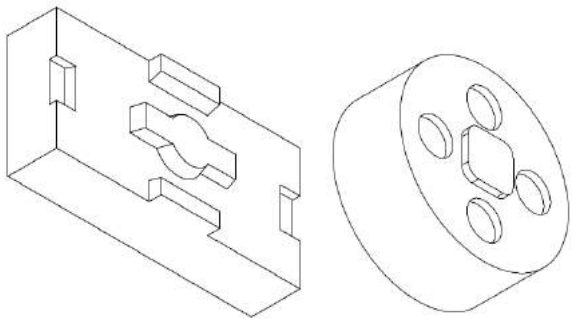
Sin embargo, para poder comandar un movimiento del robot, el PC debe establecer una conexión TCP al puerto 50240 y de esta manera iniciar la tarea *Motion Server*. Entonces la captura de *Wireshark* confirma que el PC no intenta conectarse en ningún momento a este puerto y se genera un error que termina la comunicación. Lo anterior sugiere que hay algunas modificaciones que deben realizarse en los paquetes creados para el robot, así como los archivos de configuración.

Inicialmente se efectúan ajustes a los paquetes del robot en ROS, para lograr que estos funcionen correctamente en un modo demo y posteriormente es validada la interacción con el driver. En constraste con la condición anterior, para este caso se evidencia que dicho driver no permite establecer comunicación con el controlador cuando se encuentran definidos más de un grupo de control y menos de cuatro. Es necesario entonces ajustar el código fuente para poder conectarse a un equipo con una configuración diferente.

Finalmente, la comunicación es satisfactoria a partir de un nodo de ROS, con lo cual el modelo virtual desarrollado se encuentra completo. A partir de éste, puede desarrollarse la aplicación robótica de interés utilizando ROS.

3.3. Exploración del mecanizado a través de programación estructurada

A través de algunos ejercicios básicos se busca explorar de manera preliminar el sistema robótico de ocho articulaciones, abordando el desarrollo de aplicaciones de mecanizado multiejes mediante una programación estructurada. En la Figura **3-9** se presentan los modelos de dos piezas para ser mecanizadas, las cuales incluyen primitivas básicas de diseño.



(a) Pieza 1 (b) Pieza 2

Figura 3-9.: Modelo de piezas a ser mecanizadas

Mediante el software de simulación de Motoman (MotoSim EG-VRC) se crean los programas parametrizados, que con el uso de variables de posición del robot permiten definir continuamente las primitivas de movimiento (arcos o paralelogramos). En cada una de las piezas se establece una condición de trayectoria para que la herramienta se ubique perpendicular a la superficie durante la ejecución de todo el movimiento del robot.

La Figura **3-10** describe el diagrama de flujo para llevar a cabo el mecanizado de las piezas, mediante estrategia de remoción de material de volúmenes elementales.

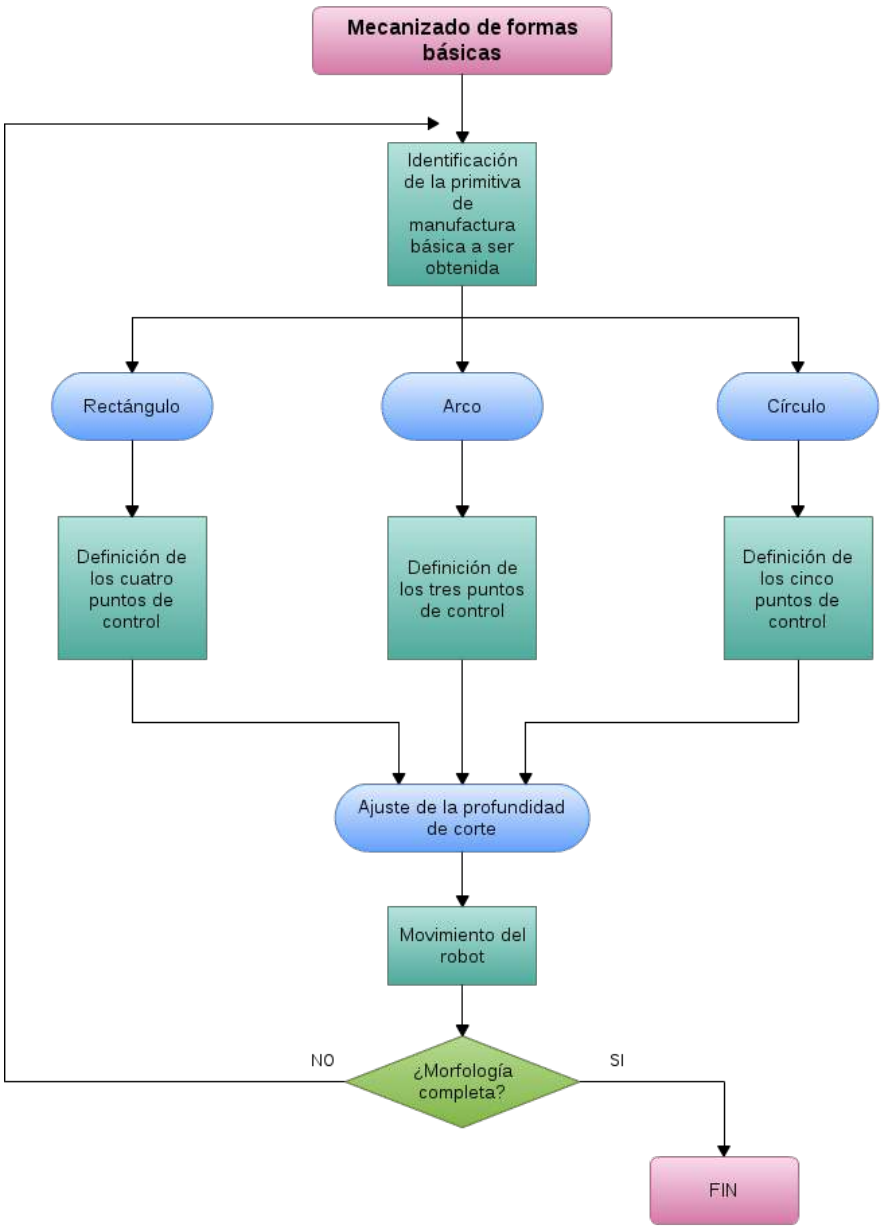


Figura 3-10.: Esquema para el mecanizado de formas básicas

3.3.1. Estrategia experimental de manufactura para la pieza 1

La estrategia de mecanizado para la Pieza 1 es bosquejada en la Figura **3-11**, a partir de remoción de material basado en volúmenes elementales (MRSEV, *Material Removal Shape Element Volume*), mediante el uso de patrones geométricos aplicados en áreas específicas de la pieza, como en **3-11**a, **3-11**b, **3-11**c y **3-11**d. Adicionalmente, para la sección circular se define un patrón compuesto por segmentos de arco según **3-11**e.

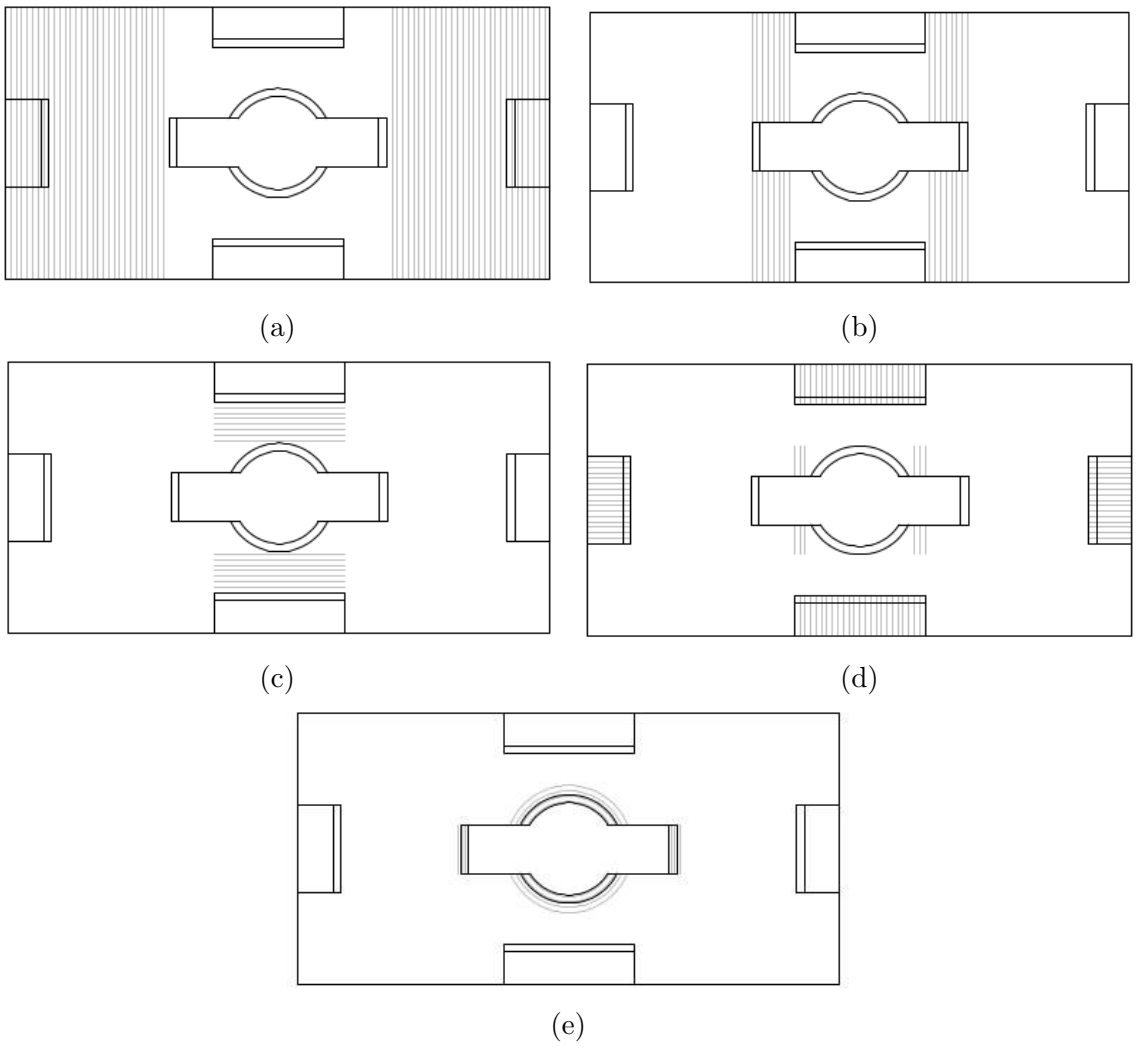


Figura 3-11.: Modelo de piezas a ser mecanizadas

Cabe señalar que todas las formas geométricas básicas en la pieza contienen paredes inclinadas que deben ser calculadas de acuerdo al paso en cada sección, lo cual conlleva a la aparición de escalones en la superficie de la pieza.

3.3.2. Estrategia experimental de manufactura para la pieza 2

Para la Pieza 2 se tiene una estrategia de corte ilustrada en la Figura **3-12**. En este caso se hace uso de un patrón circular completo **3-12**a y **3-12**d; al igual que se incluyen secciones rectangulares **3-12**b y segmentos de arco **3-12**c para describir la geometría de esta pieza.

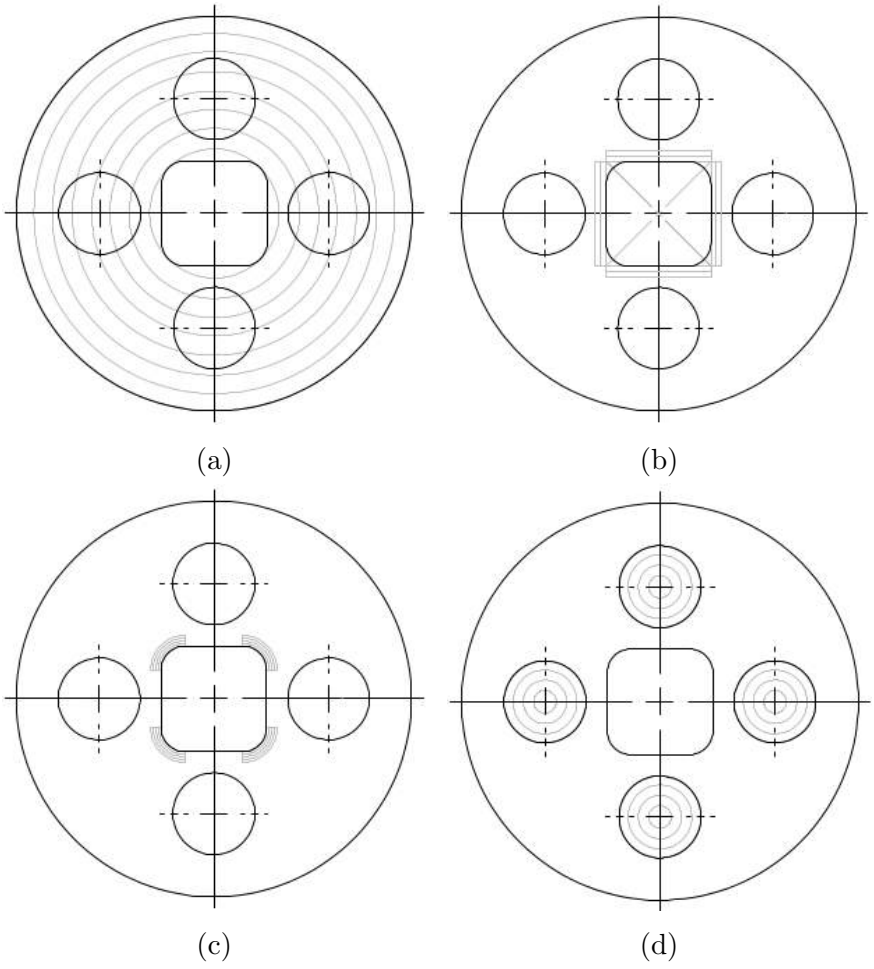
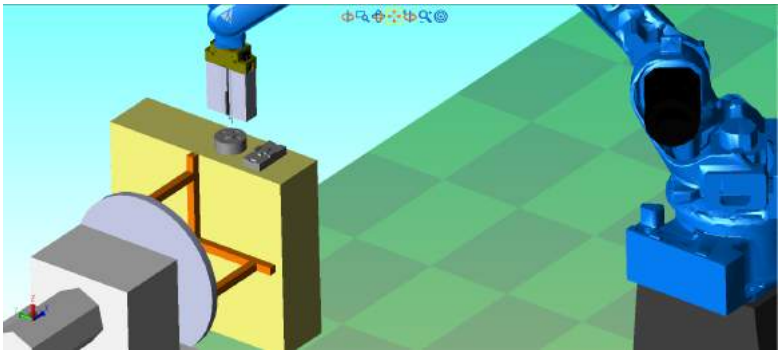


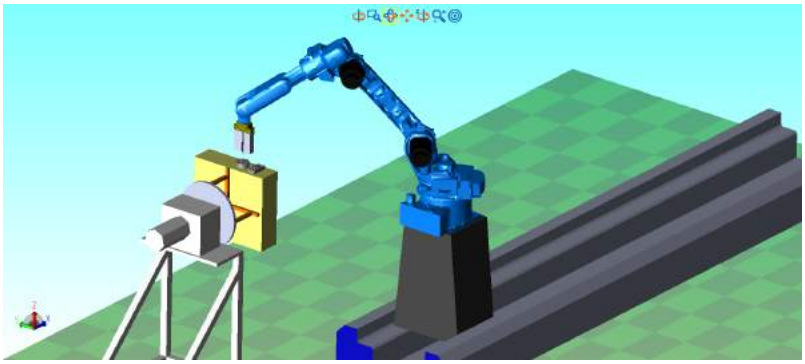
Figura 3-12.: Modelo de piezas a ser mecanizadas

3.3.3. Montaje experimental para las piezas 1 y 2

Para las dos piezas experimentales, el material de inicio correspondiente es fijado a la estructura soportada con el posicionador rotacional, lo cual genera un cambio en su posición angular debido el movimiento del octavo eje. En la Figura **3-13** se presenta la simulación del montaje de las piezas a través de MotoSim.



(a)



(b)

Figura 3-13.: Ubicación de las piezas en entorno de simulación MotoSim EG-VRC

Se ha seleccionado un material blando para ser mecanizado, de forma que las fuerzas generadas sean relativamente bajas. Lo anterior facilita que tanto el robot como la sujeción sean sometidos a muy bajos esfuerzos. El montaje es como se muestra en la Figura **3-14**.

Figura 3-14.: Montaje de las piezas en el posicionador rotacional

En el proceso de corte de las piezas se ha contemplado la interpolación de los ocho ejes que conforman el sistema robótico. Sin embargo, debido a que las piezas generaban diferentes exigencias en el posicionamiento de la herramienta (para garantizar por una parte que ésta siempre se encuentre normal a la superficie), al igual que la aparición de áreas quemadas en la pieza, las trayectorias son ejecutadas con diferentes velocidades.

3.4. Generación de trayectorias de mecanizado

Las trayectorias de mecanizado se obtienen mediante dos estrategias, de acuerdo con la Figura **3-15**. La primera corresponde a superficies libres definidas a través un archivo CAD, el cual contiene la pieza deseada. La segunda hace referencia a la definición manual de la trayectoria a partir de su descripción matemática ya sea utilizando una ecuación paramétrica o con la construcción geométrica de la superficie.

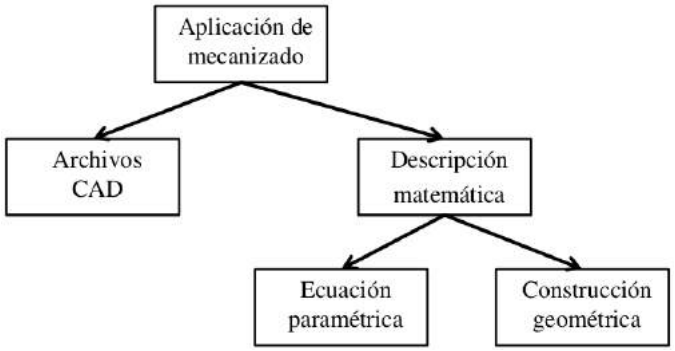


Figura 3-15.: Esquema para la generación de trayectorias de mecanizado

3.4.1. Estrategia para generación de trayectorias en superficies libres

En este caso específico se hace uso de la librería VTK, particularmente para la computación gráfica de la pieza. Debido a que los diferentes procesos aplicados conllevan una carga computacional alta, se plantea la conveniencia de seleccionar un formato relativamente ligero para los archivos CAD. Por lo anterior se sugiere que estos sean exportados en formatos PLY o STL.

La Figura **3-16** presenta una descripción básica de la estrategia establecida para la generación de trayectorias a partir de superficies con formas libres. Este proceso se subdivide en una etapa denominada Modelamiento CAD seguida por una de Modelamiento CAM

La etapa CAD inicia con un incremento en la densidad de la superficie y posteriormente se aplica un efecto de dilatación.

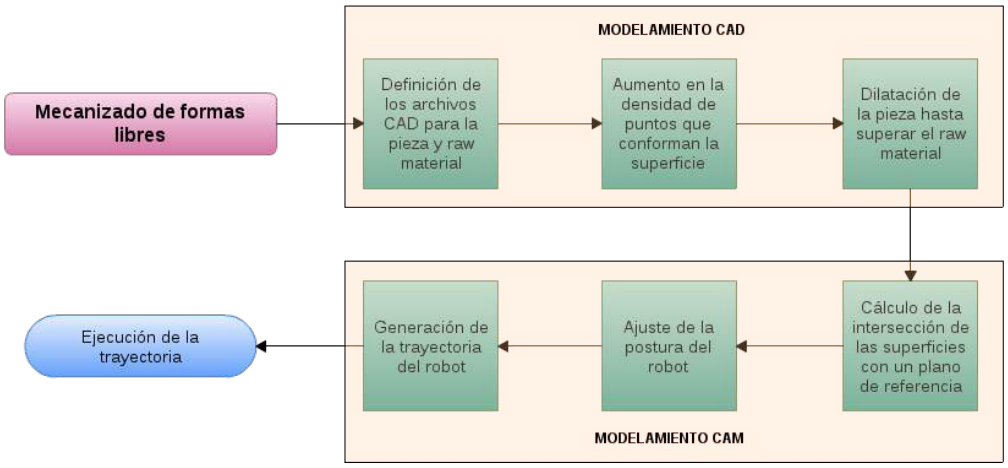


Figura 3-16.: Estrategia para la generación de trayectorias de mecanizado en formas libres

- **Incremento en la densidad de puntos de la superficie**
  Experimentalmente se evidencia que si la pieza a mecanizar presenta superficies con mayor curvatura, la guía lineal debe moverse a una alta velocidad, e incluso en algunas cirscunstancias exceder su máxima capacidad. Lo anterior conlleva a la aparición de una alarma en el robot debido a comandos excesivos de movimiento. A través del análisis de una captura de *Wireshark* cuya información es registrada en la Tabla **3-5**, los movimientos enviados justo antes de la aparición de la falla sugieren que ROS-I comanda una diferencia posicional de $0{,}1803m$ en un intervalo de $0{,}2s$. Esto requiere una velocidad de $0{,}902m/s$, lo cual supera la máxima velocidad de la guía lineal ($0{,}7188m/s$). Se evidencia de hecho que la ejecución de casi todos los desplazamientos se realiza a bajas velocidades, y son solamente unos pequeños puntos de la trayectoria aquellos que representan gran exigencia para la guía lineal.

Tabla 3-5.: Posiciones comandadas por ROS-I en la aparición de falla debido a exceso de velocidad.

Tiempo [s]	**Posición [m]**	**Velocidad comandada [m/s]**
128,0	−0,0464	−0,00195
128,2	−0,1689	−0,61233
128,4	−0,3492	−0,90166
128,6	−0,3491	−0,00065
128,8	−0,3490	−0,00065

En la concepción de ROS-I al menos en lo correspondiente a los drivers de Motoman, no se dispone directamente sobre el equipo de un control de velocidad que permita evitar

la aparición de este problema. Por lo anterior, se plantea el incremento en la densidad de puntos de las superficies de inicio, para que el perfil de velocidad del controlador no pueda desarrollarse completamente y de esta manera obtener una reducción general en la velocidad alcanzada durante la ejecución de la trayectoria.

Particularmente en este caso se implementa un filtro para aplicar una subdivisión lineal a la superficie de entrada. De esta manera, cada triángulo que conforma la superficie es dividido iterativamente en cuatro nuevos triángulos a la salida, como en la Figura **3-17**.

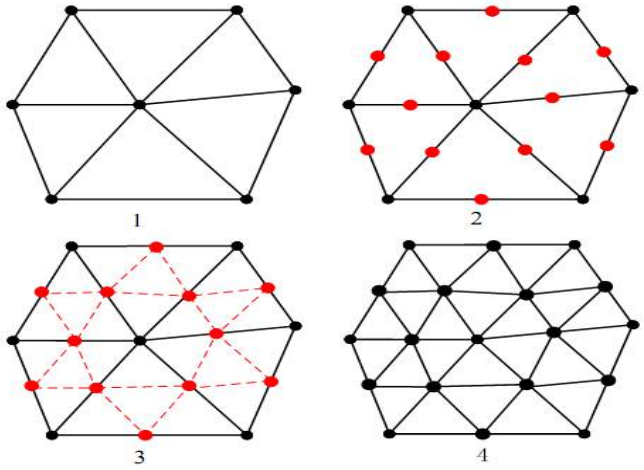


Figura 3-17.: Incremento en la densidad de puntos de la superficie

- **Dilatación de la superficie**
  El principio de modelamiento en la dilatación tridimensional consiste en desplazar una esfera de determinado radio a lo largo de toda la superficie, de forma que el centro de dicha esfera define la nueva superficie, como se aprecia en la Figura **3-18**.

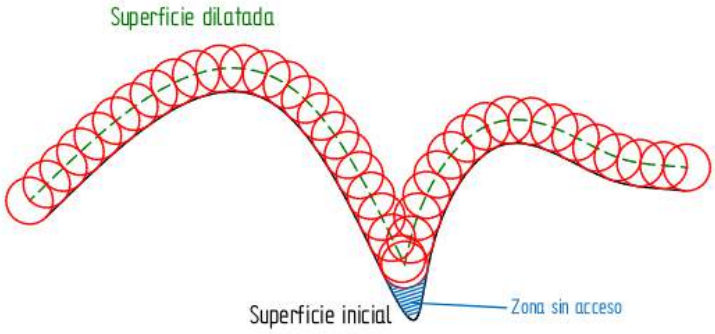
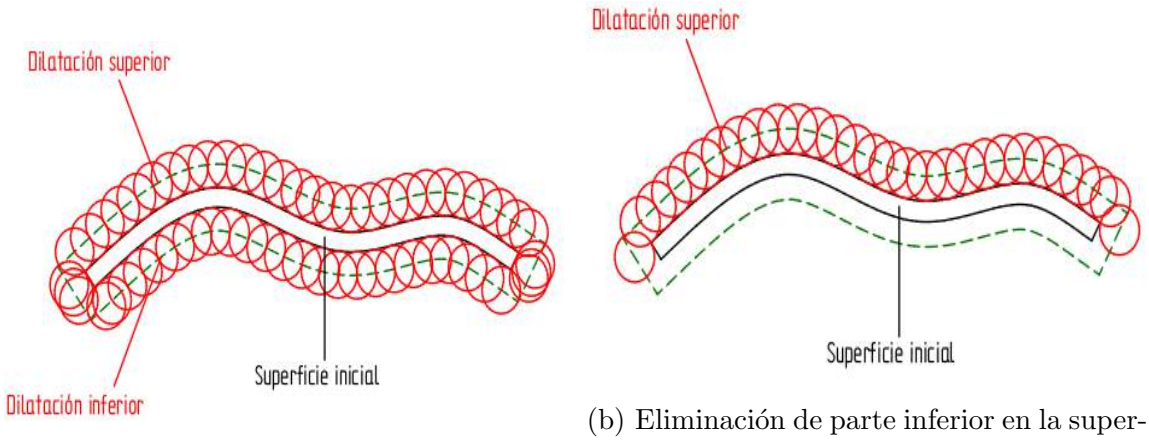


Figura 3-18.: Principio de dilatación de una superficie

El objetivo de aplicar este procedimiento es generar las pasadas de corte en caso tal que sea necesario realizar desbaste del material inicial. Adicionalmente, proporciona la descripción requerida para los movimientos de retracción y reposicionamiento de la herramienta durante el mecanizado.

Debido a que la dilatación es un proceso aplicado tanto en el área superior como inferior de la superficie, se hace necesario eliminar la parte inferior de la pieza dilatada, con el fin de garantizar trayectorias accesibles y libres de colisiones. La Figura **3-19**a ilustra el resultado de la dilatación aplicada a una superficie, mientras que la Figura **3-19**b presenta la superficie dilatada luego de eliminar su parte inferior.



(a) Superficie dilatada

(b) Eliminación de parte inferior en la superficie dilatada

Figura 3-19.: Dilatación de la superficie

Se aplica metódicamente una secuencia iterativa de dilatación de la superficie sobre la pieza deseada, hasta que ésta cubre completamente la pieza inicial del mecanizado. Se tiene entonces que, el punto de partida es como se muestra en la Figura **3-20**a, donde la superficie de inicio se encuentra por encima de la superficie deseada. Tras dilataciones continuas como en las Figuras **3-20**b y **3-20**c se obtienen superficies que crecen de forma progresiva y que eventualmente cubrirán la superficie definida como inicial para el proceso o *raw material*, de acuerdo a lo presentado en la Figura **3-20**d. En el instante en que la superficie dilatada no genera ningún mecanizado en la pieza, el proceso iterativo se detiene y dicha superficie es además ignorada. Finalmente, debe aplicarse el mecanizado a cada una de las superficies generadas en orden opuesto, como en la Figura **3-20**e.

Cada una de las superficies obtenidas debe ser procesada de forma independiente para calcular la trayectoria que define el mecanizado de cada capa.

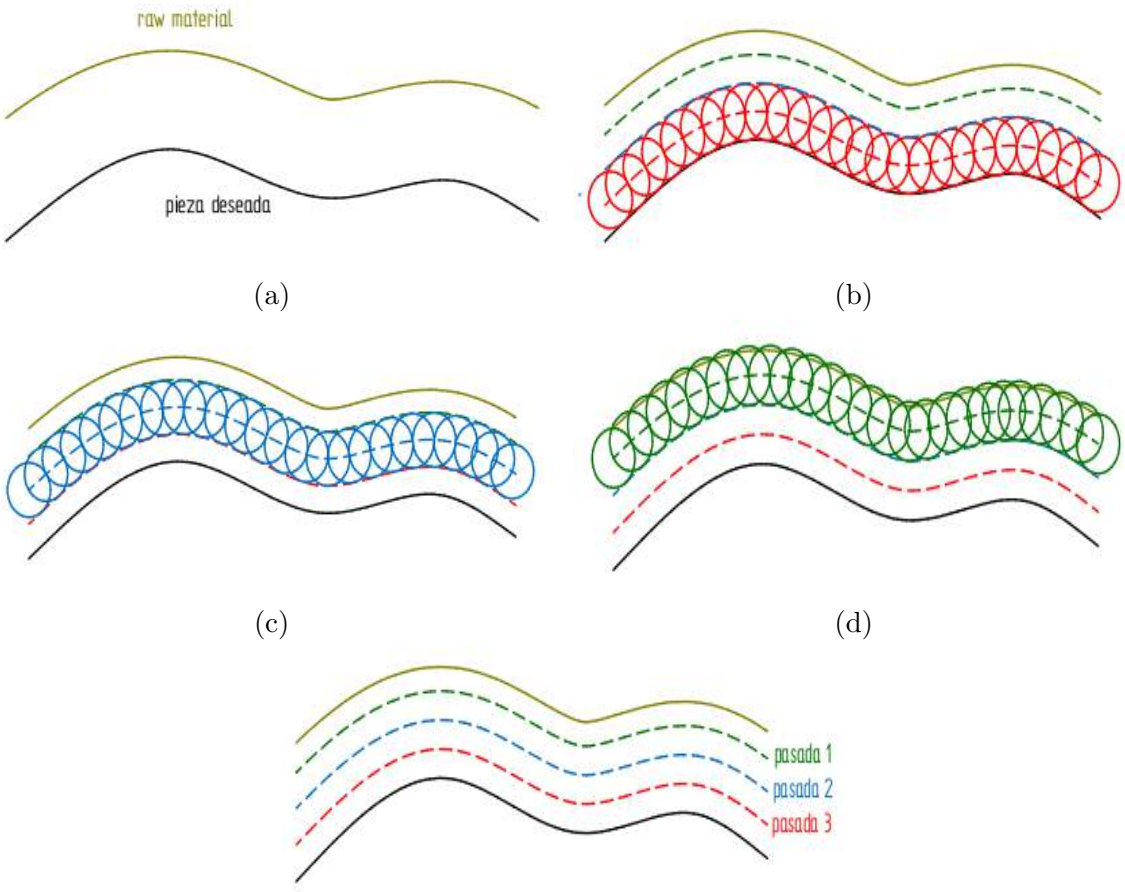
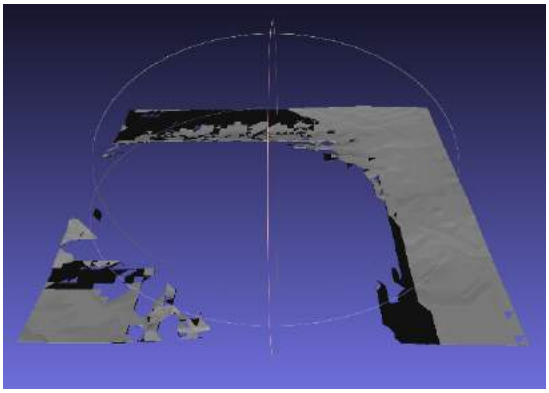


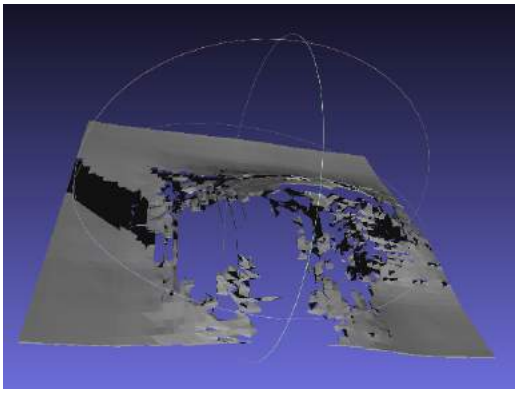
Figura 3-20.: Dilatación iterativa de la superficie deseada

Dependiendo de las formas geométricas presentes en la pieza, el uso de esferas con mayor diámetro puede conllevar a una disminución en los detalles de la superficie obtenida tras la dilatación, debido a que esferas más grandes no pueden alcanzar completamente espacios reducidos. Es de esperar entonces la conveniencia de utilizar esferas pequeñas, lo cual además facilita obtener menores profundidades de corte en el mecanizado. No obstante, mediante pruebas se evidencia que a medida que el tamaño de la esfera se reduce, aparecen progresivamente agujeros en la superficie dilatada, particularmente en aquellas zonas de mayor curvatura y la generación de trayectorias no funciona correctamente. En la Figura **3-21** se aprecia el efecto de diferentes radios de esfera usados para obtener dilatación de las superficies.

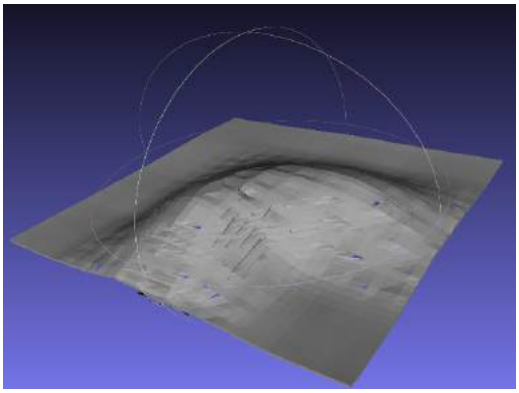
- **Superficies regladas**
  La etapa CAM inicia con un concepto similar al de las superficies regladas. En este caso



(a) Dilatación con radio 1mm



(b) Dilatación con radio 2mm



(c) Dilatación con radio 4mm

Figura 3-21.: Resultados de la dilatación de la superficie usando diferentes radios de esfera

la superficie se obtiene a través de un plano transversal que se desplaza a lo largo de la pieza, con un avance de acuerdo al radio de la herramienta utilizada. Durante cada descripción de este plano se obtiene la línea de intersección con la superficie, definiendo de esta manera el conjunto de líneas que conforman la trayectoria del robot durante el proceso de mecanizado. Adicionalmente, el posicionamiento del robot entre cada línea de mecanizado se realiza con un movimiento en vacío, a lo largo de la correspondiente superficie dilatada. A partir de lo anterior, se tiene que no hay contacto con la pieza durante el segundo conjunto de líneas, y que son utilizadas únicamente como secuencias de reposicionamiento de la herramienta para cada corte.

La Figura **3-22** presenta las líneas de intersección obtenidas entre la superficie deseada y cada plano de referencia.

- **Postura del robot a lo largo de la superficie**
  Cada una de las líneas calculadas como intersección de la superfice con un plano de referencia definen la trayectoria del robot. De forma general, a partir de la descripción

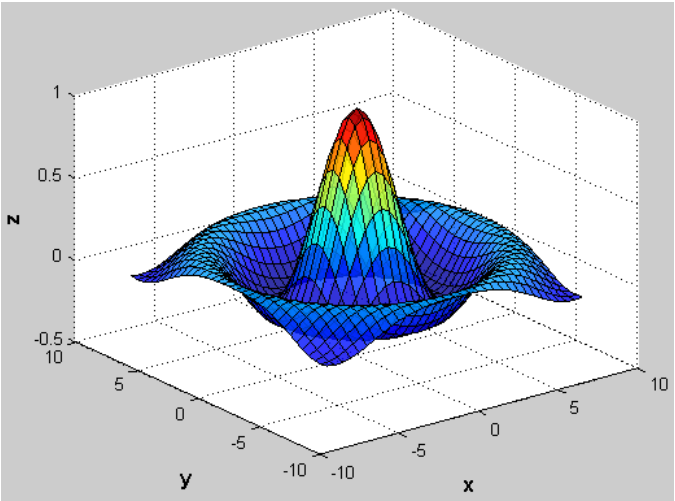


Figura 3-22.: Intersección de la superficie con un plano de referencia

de cada línea como una secuencia de puntos, se obtiene directamente la información de la posición tridimensional para cada segmento de la trayectoria. Adicionalmente, para establecer la orientación de la herramienta se calcula la normal a cada punto, y se alinea el eje Z del TCP tal que sea coincidente con la normal obtenida. Por otra parte, el eje X es ubicado de acuerdo a un vector definido entre el punto actual y el siguiente en la línea, para garantizar la misma orientación en toda la trayectoria. Finalmente, el vector Y es calculado como el producto cruz $Z \times X$. La Figura **3-23** representa la orientación de la herramienta a lo largo de la superficie.

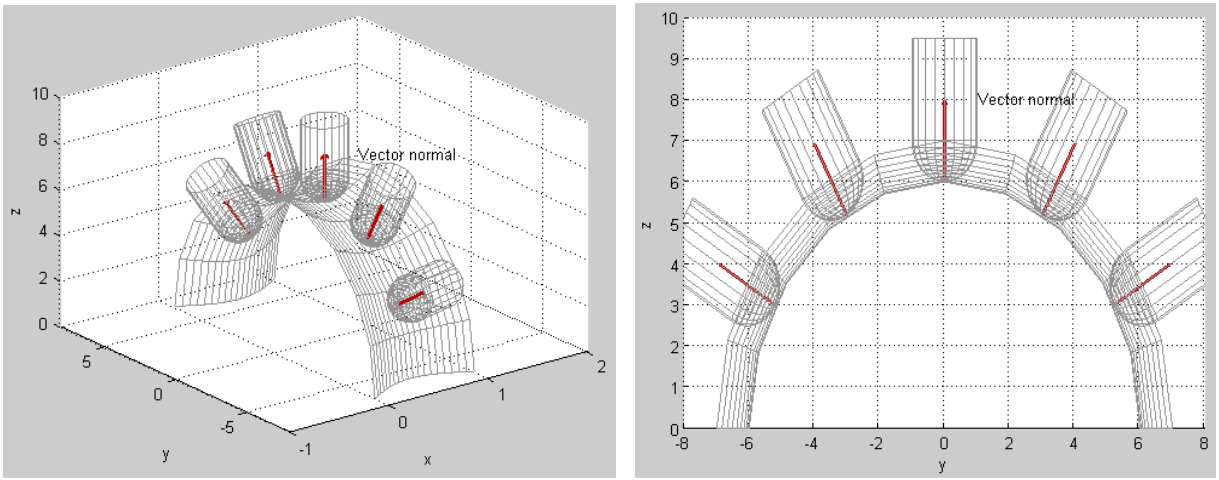


Figura 3-23.: Variación de la orientación de la herramienta con respecto al vector normal en cada punto

3.4.2. Descripción matemática de la superficie

Cuando la trayectoria deseada puede ser definida mediante una ecuación paramétrica, es posible incluir la descripción matemática directamente en un nodo de ROS para generar la información tridimensional para cada punto. Adicionalmente, la orientación de la herramienta es definida con un quaternion donde los correspondientes valores pueden varias dependiendo de la superficie o establecerse fijos a partir de la pose actual del robot cuando el nodo es ejecutado.

Una superficie sencilla utilizada en este trabajo es la helicoidal definida a partir de las ecuaciones 3-1, 3-2 y 3-3.

$$x = k \cdot t \tag{3-1}$$

$$y = r \cdot sin(\omega t) \tag{3-2}$$

$$z = r \cdot cos(\omega t) \tag{3-3}$$

El correspondiente algoritmo en C++ es presentado en la Figura **3-24**, donde la posición inicial del robot es definida como cero de pieza, tal que la trayectoria resultante es calculada con respecto a éste. Lo anterior sugiere que valores positivos en la componente z conllevan a movimientos sin mecanizado.

```
int generateHelicoidPath(){
 // parameters
 double r = 0.01, k = 0.0002, w = 1;
 // center is start position
 double centerX, centerY, centerZ;

 for(int i = 0; i < t.size(); i++){
  double z_aux = r * cos(w * t[i]);
  double y_aux = r * sin(w * t[i]);
  double x_aux = k * t[i];

  current_pose.position.x = centerX + x_aux;
  current_pose.position.y = centerY + y_aux;
  current_pose.position.z = centerZ + z_aux;

  output_pose.push_back(current_pose);
 }
return 1;
}
```

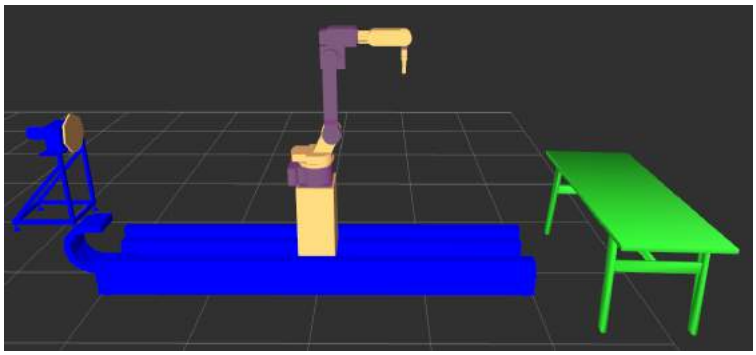
Figura 3-24.: Código ejemplo para trayectoria

3.5. Montaje experimental

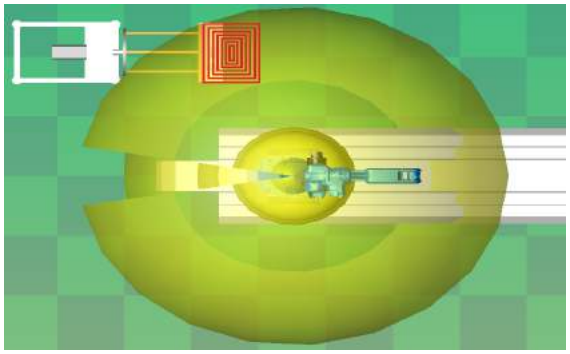
3.5.1. Layout del laboratorio

Las pruebas experimentales de mecanizado se llevan a cabo a través de enfoques de 7 DOF y 8 DOF, donde el primer caso es alcanzado únicamente la interpolación del robot y la guía lineal. En este escenario una mesa es frecuentemente usada para soportar el material a mecanizar, tal que de acuerdo con [Klimchik et al., 2017], [Caro et al., 2014] y [Lin et al., 2017], corresponde a la zona del espacio de trabajo del robot con mejor comportamiento dinámico y de rigidez (Ver Figura **3-25**).

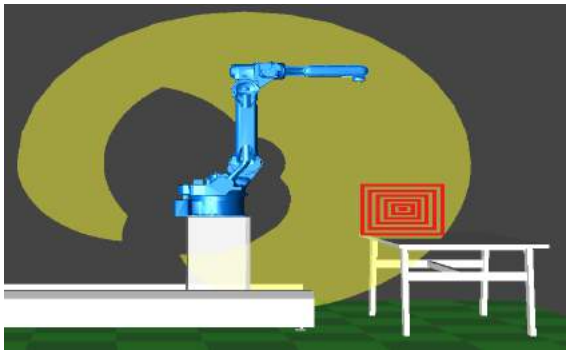
Para la exploración del mecanizado con 8 DOF el material es fijado en la estructura del posicionador rotacional, lo cual desplaza el área de trabajo hacia éste (ver Figura **3-25**b).



(a) Layout en mecanizado



(b) Espacio de trabajo en posicionador



(c) Espacio de trabajo en mesa

Figura 3-25.: Layout del laboratorio

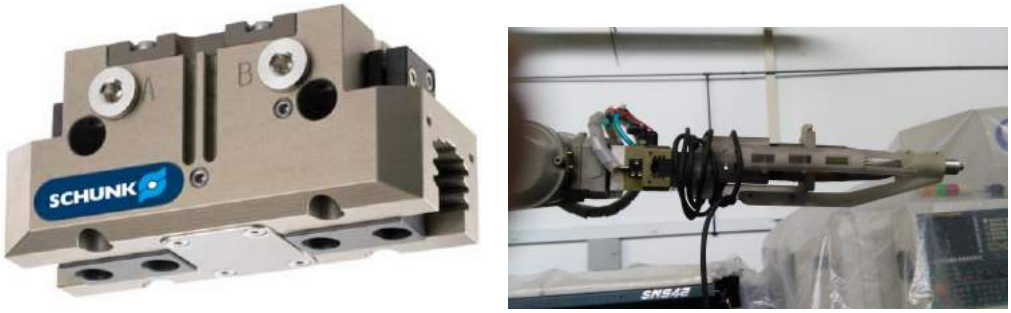
3.5.2. Fijación de la herramienta al robot

Como husillo para el mecanizado se utiliza un *mototool Proxxon* LBS/E. N^o 28 485, cuya información técnica es especificada en la Tabla **3-6**.

Tabla 3-6.: Especificaciones técnicas del *mototool*

Tensión	230 VAC
Potencia	100 W
Revoluciones	$5000 - 22000$ rpm
Longitud	300 mm
Peso	630 gr
Intensidad de ruido (máx)	70 dB(A)
Vibración en el puño (máx)	2,5 m/s^2

Con el objetivo de fijar la herramienta motorizada a la muñeca del robot, y con el apoyo de un grupo de estudiantes auxiliares de ingeniería de manufactura, se ha diseñado y fabricado un soporte para la herramienta a través de tecnología de impresión 3D con el equipo *PolyJet Eden 260V* disponible en el laboratorio. La Figura **3-26**a presenta la estructura del gripper que se encuentra montado en la muñeca del robot, y sobre el cual se fija el soporte fabricado. A su vez, la Figura **3-26**b presenta el montaje final del soporte y la herramienta en el robot.



(a) Gripper *PGN+100-2* (b) Soporte del *mototool*

Figura 3-26.: Montaje del soporte fabricado para el *mototool*

3.5.3. Alistamiento del robot Motoman

Para realizar pruebas a través de ROS con el robot Motoman del laboratorio es necesario realizar un cambio en la configuración de los ejes externos, a través de un archivo CMOS.BIN que contiene toda la información del robot: configuración, parámetros, *ladder*, variables, etc. Este archivo es cargado al controlador a través de una memoria USB insertada en el *teach pendant* e iniciando el equipo en un modo de Mantenimiento. A partir de lo anterior, el robot es ahora compatible con el modelo virtual definido en ROS, incluyendo el envío de estados entre PC-Controlador.

Adicionalmente, dependiendo de la herramienta definida para cada operación, se requiere configurar el TCP de forma que los cálculos de las interpolaciones sean realizados con res-

pecto al extremo de la herramienta. Para ello, existe un procedimiento manual que consiste en registrar cinco diferentes posiciones del robot mientras la herramienta se encuentra en contacto con un punto de referencia, o ingresando directamente los datos a partir de las dimensiones respectivas. Igualmente, deben registrarse los datos definitivos en los archivos de configuración del modelo del robot en ROS, para garantizar coincidencia entre el entorno real y el virtual.

3.6. Aproximación a aplicaciones de robótica colaborativa

Con el objetivo de abordar aplicaciones de robótica colaborativa, se plantea realizar un ejercicio basado en el seguimiento de trayectorias entre el robot Motoman y un vehículo AGV diseñado y construido en el laboratorio LabFabEx. En este caso, cada uno de los robots dispone de un PC encargado de planear la trayectoria correspondiente y transmitirla para su ejecución.

El reto consiste entonces en coordinar las trayectorias generadas, para lo cual se configura un esquema de red como el presentado en la Figura **3-27**. En ésta se aprecia que los PC's para cada máquina se encuentran conectados a una red inalámbrica común, a la vez que se comunican a través de Ethernet con los controladores respectivos.

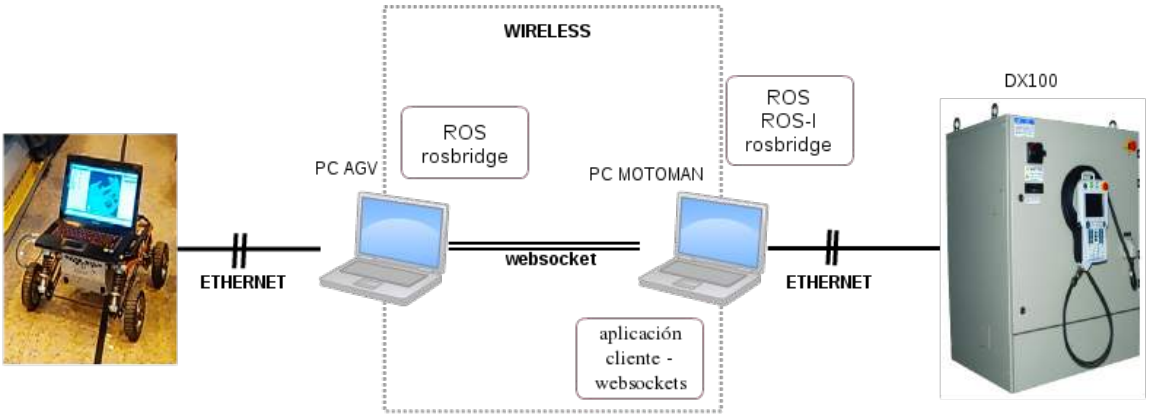


Figura 3-27.: Esquema de comunicación entre Motoman y AGV

El robot AGV se desplaza a lo largo del laboratorio de forma independiente, debido a su algoritmo de navegación implementado mediante ROS. A partir de su sistema de posicionamiento en el mapa del laboratorio, el controlador publica continuamente la pose estimada para el AGV. El PC asociado al robot Motoman debe acceder a esta información para comandar el movimiento correspondiente.

De acuerdo con [Crick et al., 2017], cada PC ejecuta una aplicación *rosbridge* mediante la cual es definido un *websocket* en el puerto 9090. Adicionalmente, el PC del Motoman imple-

menta una aplicación cliente que se conecta a los dos puertos, estableciendo un canal para leer la información publicada en un *topic* del AGV y publicarla a su vez en un *topic* para el Motoman. A partir de lo anterior, se obtiene realimentación de la posición estimada del vehículo en el laboratorio.

Debido a que el punto de referencia para cada robot no es coincidente, debe efectuarse una calibración de las posiciones correspondientes, mediante lo cual se establece una compensación aproximada definida en las ecuaciones 3-4 y 3-5.

$$X_{Motoman} = -1{,}0068 X_{AGV} - 9{,}4015 \tag{3-4}$$

$$Y_{Motoman} = -0{,}46667 Y_{AGV} - 0{,}7422 \tag{3-5}$$

A partir de lo anterior pueden finalmente realizarse pruebas de interacción entre los equipos.

4. Resultados y Discusión

El modelo virtual y de comunicación desarrollado para el sistema robótico de 8 DOF permite comandar movimientos desde un PC ejecutando ROS hacia el controlador DX100 y recibir continuamente realimentación de posición para cada articulación con fines de visualización, como se aprecia en la Figura **4-1**. Adicionalmente, debido a la facilidad de implementación de la librería OMPL en ROS, se considera el planeador de trayectorias RRT-Connect [Kuffner and LaValle, 2000] para las pruebas realizadas durante esta investigación, ya que experimentalmente ha proporcionado mejores resultados en contraste con otros algoritmos como KPIECE y PRM, pues genera trayectorias con menos movimientos redundantes.

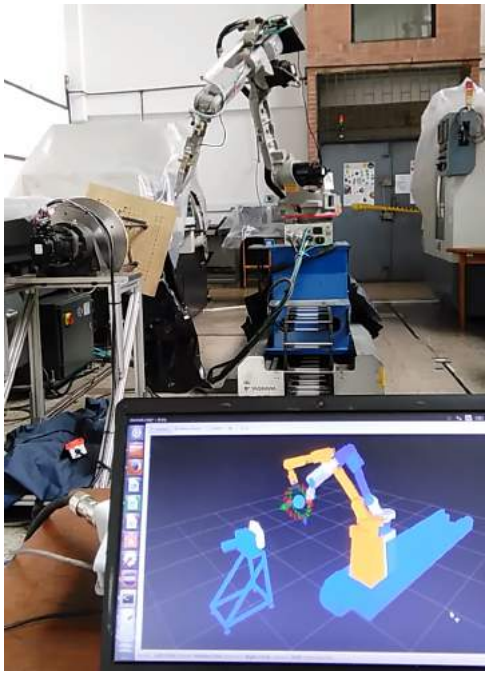


Figura 4-1.: Comunicación entre modelo virtual y real

Las trayectorias generadas deben contener información de posición para cada una de las ocho articulaciones. No obstante; debido a la construcción del modelo virtual, es posible llevar a cabo fácilmente aplicaciones que requieran únicamente el movimiento de algunos componentes del sistema.

4.1. Planeación de trayectorias con obstáculos

Con el objetivo de validar la planeación de trayectorias en presencia de obstáculos se propone un escenario experimental como el representado en la Figura **4-2**, de forma que son comandados movimientos entre dos puntos objetivos en el espacio alcanzable del robot, mientras que la trayetoria resultante considera la evasión de colisiones con el entorno.

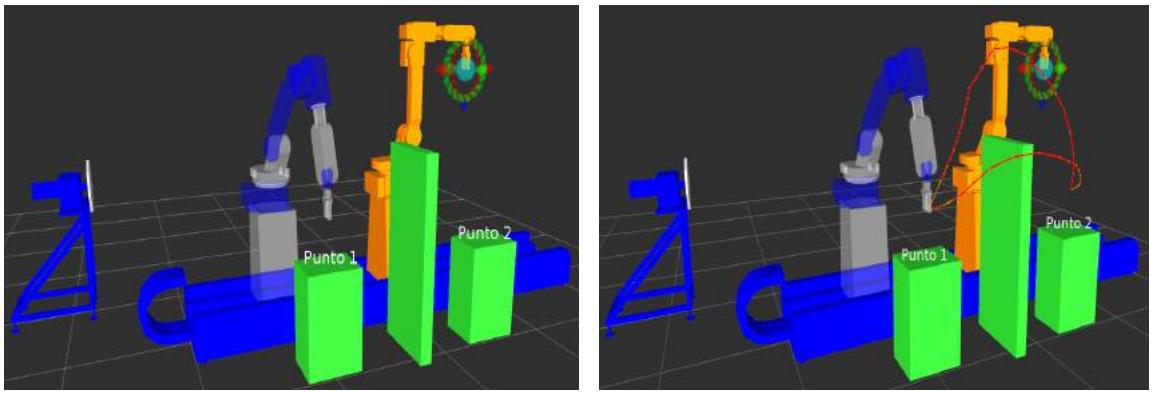
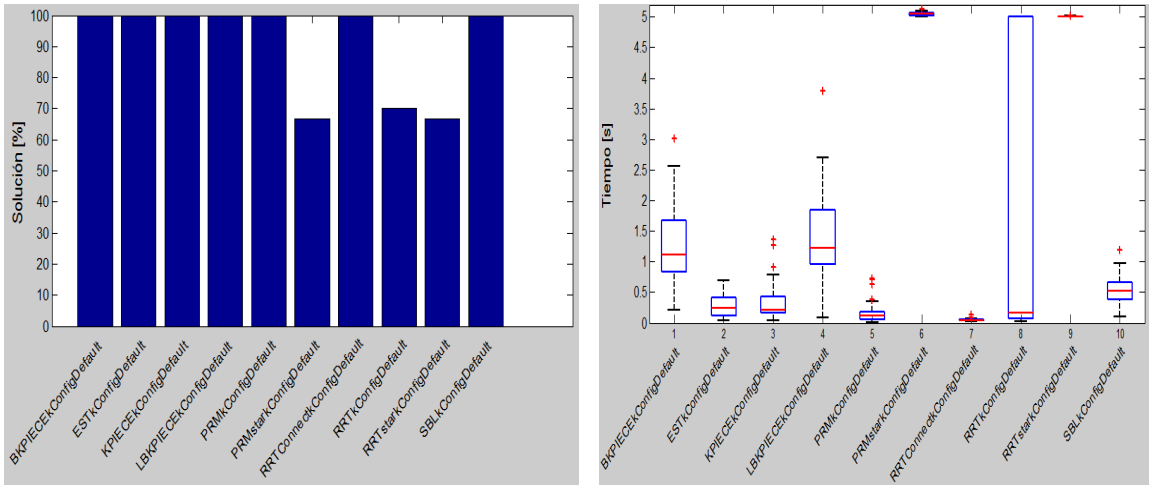


Figura 4-2.: Escenario básico para generación de trayectorias con obstáculos

A partir de lo anterior se lleva a cabo un análisis comparativo entre los algoritmos disponibles en la librería OMPL teniendo en cuenta el porcentaje de soluciones encontradas y el tiempo requerido para ello. En el primer caso, a partir de la Figura **4-3**a se aprecia que tres algoritmos alcanzan solución alrededor del 70 % de las pruebas. Adicionalmente, de acuerdo con la Figura **4-3**b la mayoría de ellos presentan tiempos de respuesta elevados y significativa dispersión de los datos.



(a) (b)

Figura 4-3.: Solución de trayectorias alcanzables en el escenario propuesto

Sin embargo, el algoritmo *RRTConnectkConfigDefault* proporciona una rápida respuesta con baja dispersión de datos. Por lo anterior, éste es utilizado en las pruebas siguientes.

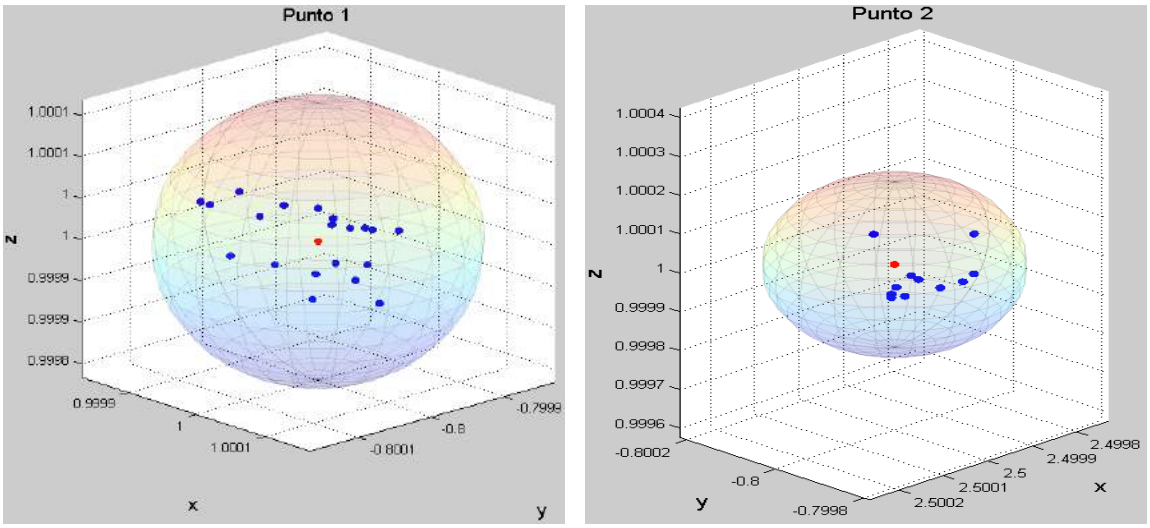
4.1.1. Prueba con algoritmo RRTConnectkConfigDefault

Haciendo uso de este algoritmo se realiza la misma prueba descrita anteriormente pero son registrados ahora los errores de posicionamiento alcanzados para cada punto, los cuales se presentan en la Tabla **4-1**.

Tabla 4-1.: Medición de precisión en posicionamiento del robot

Estimador	**Punto 1** [mm]	**Punto 2** [mm]	**Aleatorio** [mm]
RMSE_X	$7{,}4378e-02$	$8{,}3720e-02$	$7{,}7716e-02$
RMSE_Y	$4{,}9445e-02$	$4{,}2982e-02$	$4{,}7323e-02$
RMSE_Z	$5{,}1705e-02$	$5{,}5504e-02$	$5{,}3042e-02$

Las mediciones sugieren que las posiciones alcanzadas por el robot se encuentran comprendidas en una esfera de radio $0{,}142mm$ para el punto 1 (Figura **4-4**a) y $0{,}149mm$ para el punto 2 (Figura **4-4**b).



(a) (b)

Figura 4-4.: Precisión en posicionamiento del robot

4.2. Prueba experimental para el mecanizado de superficies planas

Para las pruebas experimentales se dispone de una fresa cilíndrica plana de dos filos con un diámetro de 1/8", las trayectorias de corte son definidas de forma paralela al plano YZ del robot con pasos de $1mm$, lo cual corresponde aproximadamente a la tercera parte del diámetro de la herramienta. Debido a que el movimiento de mecanizado se lleva a cabo solamente en una dirección, entre cada una de las secciones de corte obtenidas se generan desplazamientos en zig-zag para efectuar el posicionamiento de la herramienta.

La generación de trayectorias es inicialmente validada a través del mecanizado de una superficie plana controlando los 6 ejes del robot junto con la guía lineal (7 DOF).

Por la geometría propia de la superficie, la herramienta no cambia su orientación durante el proceso pero se mantiene normal a cada punto en la trayectoria de mecanizado, la cual es presentada en la Figura **4-5**; en ésta se evidencian los movimientos durante el proceso de corte, así como las retracciones para reposicionamiento.

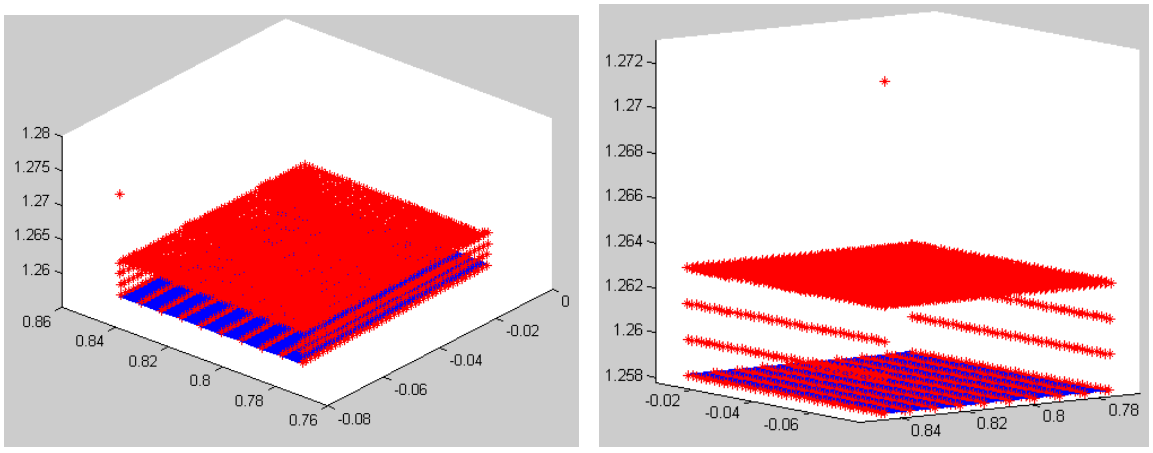


Figura 4-5.: Trayectoria generada para la superficie plana

Para esta prueba preliminar de maquinado se utiliza un material plástico con forma prismática, que por su consistencia altamente pastosa se hace difícil de mecanizar. No obstante, permite analizar el comportamiento en operación tanto del sistema robótico como del soporte fabricado y la herramienta disponible (*mototool* y fresa). La Figura **4-6** incluye la superficie obtenida luego del proceso de mecanizado, evidenciando bastante adherencia de viruta al material y en consecuencia la pieza resultante no presenta una buena calidad superficial.

En contraste, resalta que no hay partición de la viruta desprendida, obteniendo formas

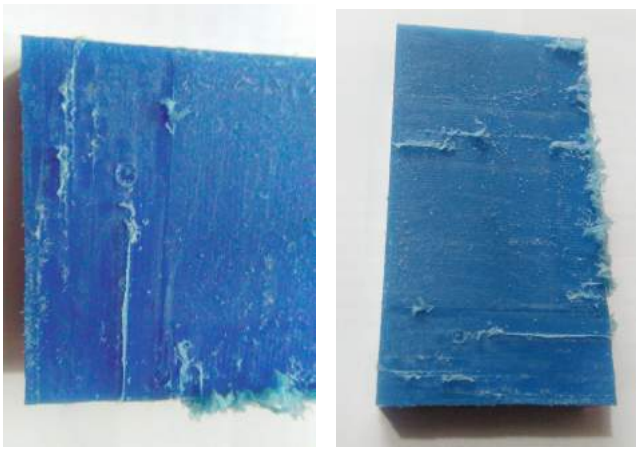
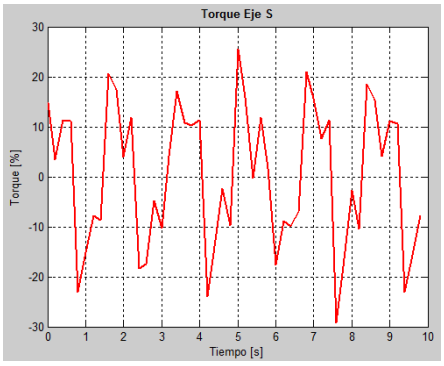


Figura 4-6.: Pieza mecanizada en superficie plana

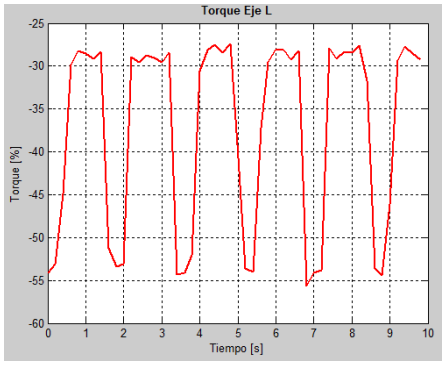
continuas y alargadas, como se ilustra en la Figura **4-7**.

Figura 4-7.: Viruta obtenida en el proceso de mecanizado de pieza plástica

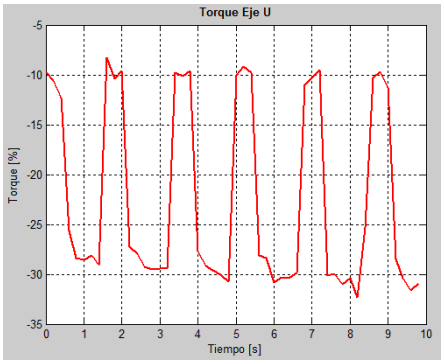
Por otra parte, mediante una herramienta virtual de osciloscopio disponible en los robots Motoman se lleva a cabo la medición instantánea de los torques a los cuales se encuentra sometido cada eje durante el proceso de mecanizado y validar de esta manera si se genera algún esfuerzo excesivo sobre el equipo. La información es obtenida en una escala porcentual dependiendo de la capacidad de carga para cada eje, por lo tanto el Anexo C especifica la equivalencia de los torques máximos en el robot. La medición de los torques ocurridos durante el mecanizado de la superficie plana evidencian un comportamiento con tendencia periódica en los ejes del sistema robótico. Cabe indicar que en condiciones normales de operación, es frecuente que por períodos instantáneos de tiempo los torques alcancen valores entre $150\,\% - 200\,\%$ de su capacidad máxima, dependiendo de la aplicación en particular. No obstante, los valores alcanzados para este caso son relativamente bajos, lo cual sugiere un reducido esfuerzo para el robot. En la Figura **4-8** se presentan las gráficas correspondientes a los torques obtenidos.



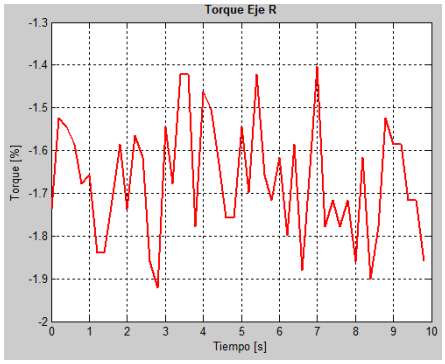
(a) Torque Eje S



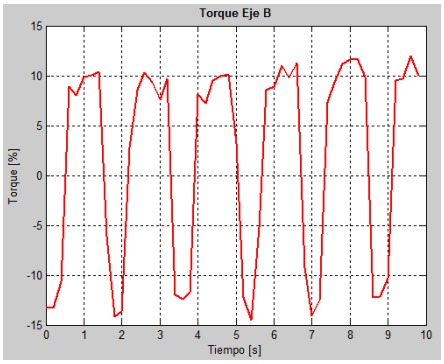
(b) Torque Eje L



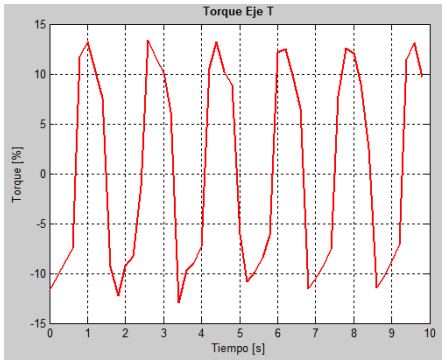
(c) Torque Eje U



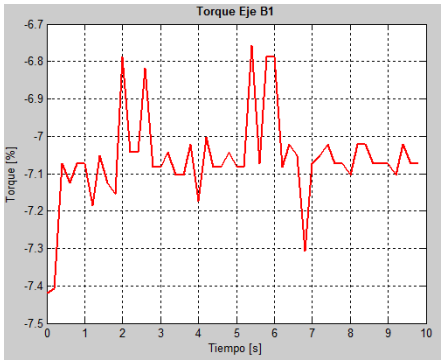
(d) Torque Eje R



(e) Torque Eje B



(f) Torque Eje T



(g) Torque Eje B1

Figura 4-8.: Comportamiento de los torques obtenidos durante el mecanizado de superficies planas

Particularmente los ejes S, L y U reflejan los valores de torque más altos (por debajo del 55 %) debido a que estos sostienen la masa del equipo y lo mueven a lo largo de la trayectoria. Los ejes R, B y T que constituyen la muñeca del robot y en consecuencia se ven principalmente involucrados en su postura, presentan valores pequeños (menores al 15 %) pero específicamente el eje R describe un comportamiento oscilatorio con magnitud inferior al 2 %. Finalmente; pese a que la guía lineal (B1) desplaza la masa del robot, el torque desarrollado por este eje es bastante bajo ($< 8\,\%$) pero contiene igualmente un efecto oscilatorio.

4.3. Prueba experimental para el mecanizado de superficies libres con orientación fija de la herramienta

Para abordar las pruebas de mecanizado de superficies con formas libres se plantea inicialmente la generación de una trayectoria en la cual no se producen cambios en la orientación de la herramienta durante el proceso de corte. Esta superficie es definida a partir de un modelo CAD como el mostrado en la Figura **4-9**.

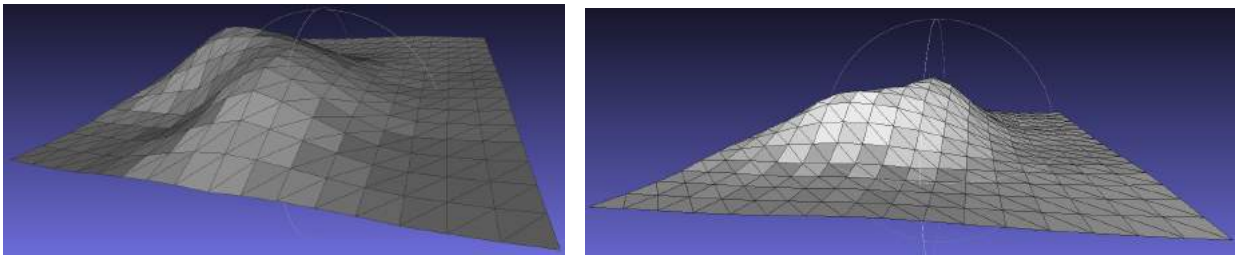
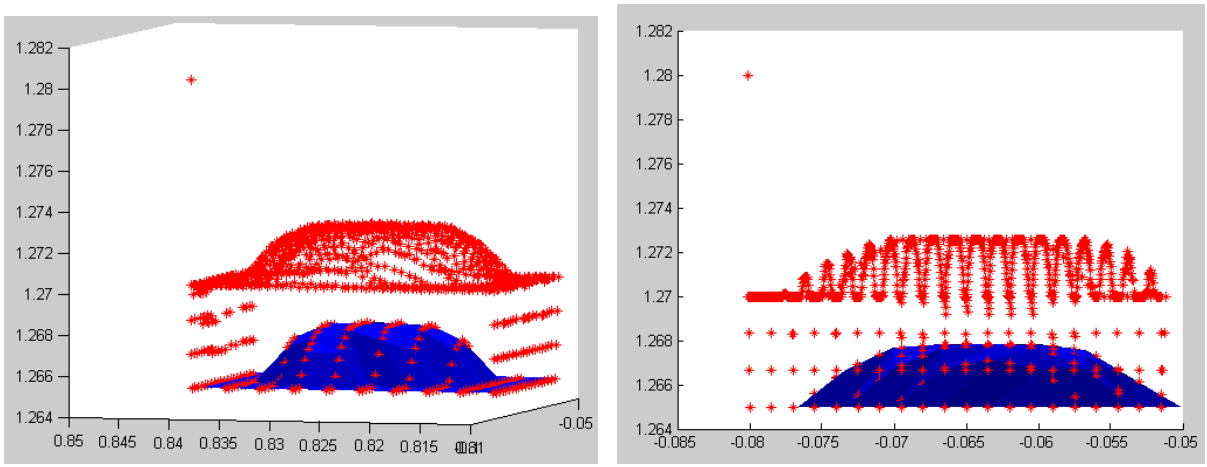


Figura 4-9.: Modelo CAD de la superficie con forma libre

Esta superficie CAD es generada de manera arbitraria a partir de la deformación de un plano, tal que al aplicar sobre ésta el algoritmo para generación de la trayectoria de mecanizado se obtienen las posiciones para el robot de acuerdo a la Figura **4-10**. Es posible apreciar que los movimientos en vacío efectuados por la herramienta para reposicionarse en cada pasada de corte se llevan a cabo a través de una geometría con forma similar a la pieza, que corresponde a la superficie dilatada. La misma condición ocurre en la prueba anterior; pero debido a tratarse de una geometría plana, el efecto no es tan evidente.

El mecanizado de esta superficie se lleva a cabo en el posicionador, pero no es efectuada ninguna rotación sobre este dispositivo. La pieza resultante se aprecia en la Figura **4-11**, en



(a) (b)

Figura 4-10.: Trayectoria generada para la superficie libre

ella se evidencian las marcas producidas por la herramienta en cada pasada de corte aunque la calidad percibida visualmente es bastante buena. De cualquier manera, constituye una aproximación interesante al proceso de mecanizado multi-eje con el robot.

Figura 4-11.: Mecanizado de superficie libre sin cambio de orientación de la herramienta

4.3.1. Prueba de la trayectoria con ajuste en orientación de la herramienta

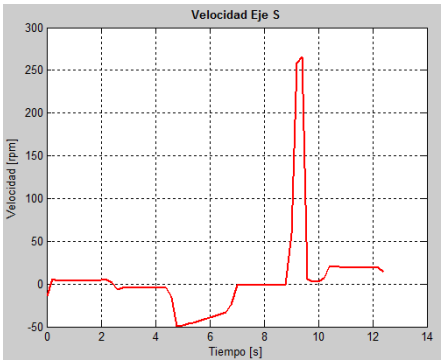
Sobre la misma superficie de forma libre se plantea que la herramienta sea orientada de acuerdo a la normal correspondiente a cada punto de la trayectoria. No obstante, a partir de esta prueba se evidencia una alta exigencia en velocidad a la que se encuentran sometidos los ejes del sistema robótico cuando debido a la curvatura de la pieza se requieren cambios fuertes en la postura del robot de manera casi instantánea.

La Figura **4-12** ilustra la velocidad desarrollada por el motor de cada eje durante la ejecución de la trayectoria. En general, la velocidad experimentada es relativamente baja, aunque existen instantes en los cuales el ajuste en la postura del robot para mantener alineada la herramienta con la superficie conlleva a que la velocidad se incremente fuertemente presentando picos pronunciados en cada una de las gráficas.

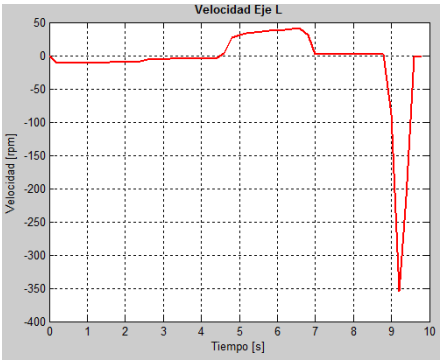
Pese a que el orden de magnitud en los picos de velocidad para casi todos los ejes del robot es menor a 350 rpm, resalta que el eje R alcanza valores significativamente mayores (900 rpm) de acuerdo con la Figura **4-12**d. Sin embargo, la guía lineal (Figura **4-12**g) se ve particularmente afectada por la exigencia instantánea de velocidad (500 rpm), generando la aparición de alarmas en el robot durante su ejecución. Cabe señalar que en la medida en la cual las superficies tienden a ser más planas hay menor aparición de estos picos de velocidad.

Igualmente, los torques obtenidos durante la ejecución de la trayectoria son presentados en la Figura **4-13**. En contraste con la prueba inicial, las gráficas describen un comportamiento más suavizado, con magnitudes similares y sin periodicidad, especialmente para los ejes S (Figura **4-13**a), L (Figura **4-13**b) y U (Figura **4-13**c). Por otra parte; debido al ajuste en la orientación de la herramienta, los torques para R (Figura **4-13**d) y B1 (Figura **4-13**g), se incrementan significativamente alcanzando valores de 40 % en el primero y 10 % en el segundo, mientras que los ejes B (Figura **4-13**e) y T (Figura **4-13**f) se mantienen en rangos similares a los anteriores, por lo cual en general los valores obtenidos se encuentran en rangos adecuados de operación. Se tiene entonces que la mayor exigencia se debe a los cambios de velocidad requeridos para el proceso, y no en los esfuerzos generados sobre el robot.

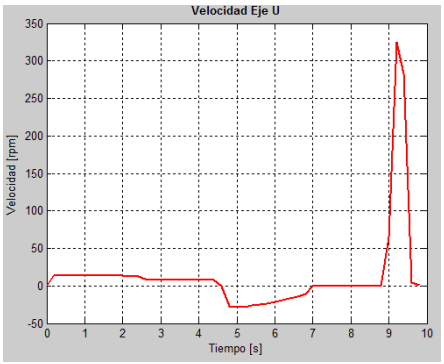
Para evitar la aparición de alarmas, las trayectorias generadas son definidas entonces a través de una interpolación lineal con resolución de 0,5mm para describir el desplazamiento de la herramienta en la dirección del avance (previamente se utilizó 0,8mm). Los ajustes en esta resolución favorecen la obtención de un movimiento más suavizado, especialmente para la guía lineal.



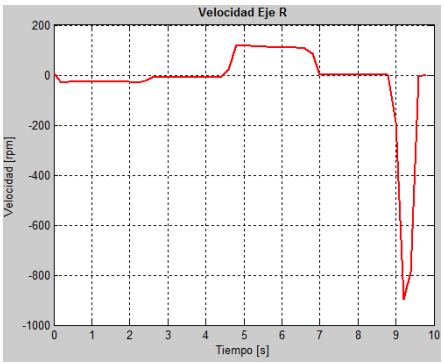
(a) Velocidad Eje S



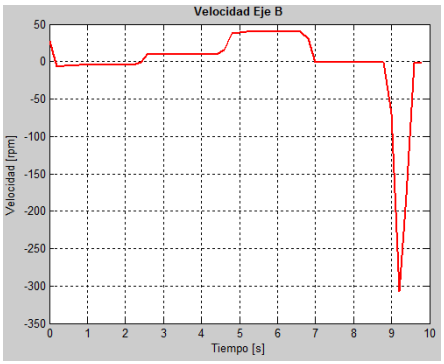
(b) Velocidad Eje L



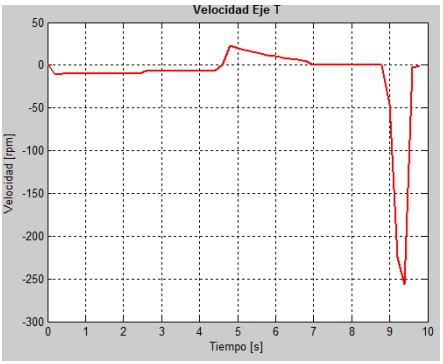
(c) Velocidad Eje U



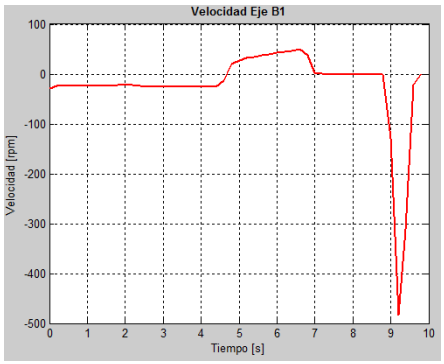
(d) Velocidad Eje R



(e) Velocidad Eje B

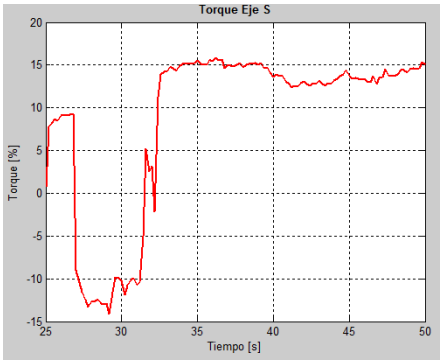


(f) Velocidad Eje T

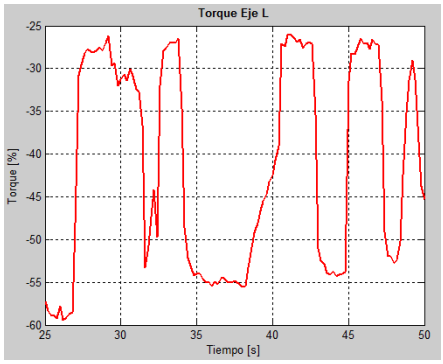


(g) Velocidad Eje B1

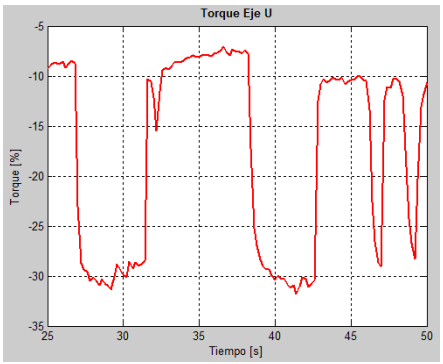
Figura 4-12.: Velocidades obtenidas durante la trayectoria para superficies libres con orientación de la herramienta



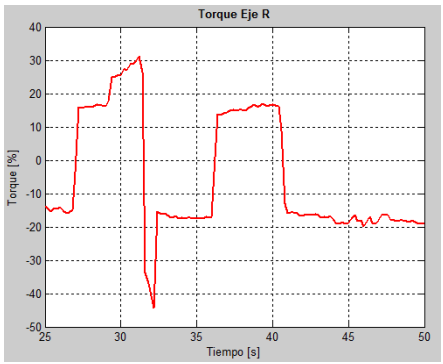
(a) Torque Eje S



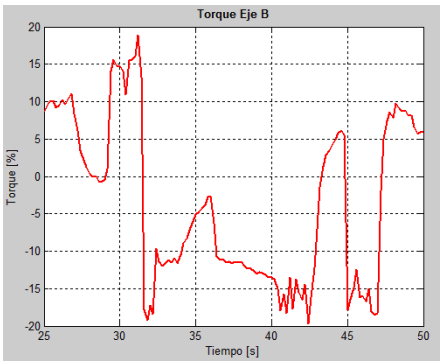
(b) Torque Eje L



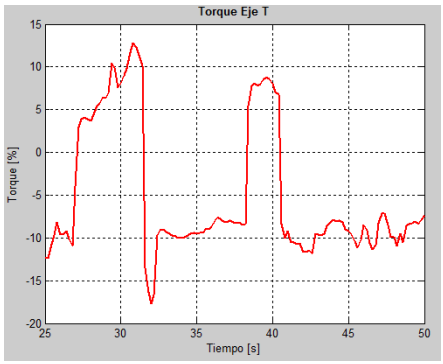
(c) Torque Eje U



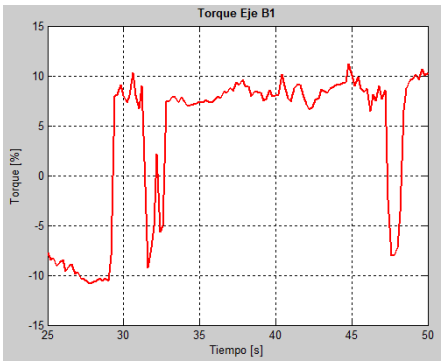
(d) Torque Eje R



(e) Torque Eje B



(f) Torque Eje T



(g) Torque Eje B1

Figura 4-13.: Torques obtenidos durante la trayectoria para superficies libres con orientación de la herramienta

Una vez la trayectoria puede ser ejecutada sin la aparición de alarmas, se evidencia que a pesar de realizar previamente la calibración de la herramienta, las instrucciones de rotación confieren un giro alrededor de un punto que no coincide con el extremo de la fresa, obteniendo un ajuste excéntrico (como el visualizado en la Figura **4-14**), que genera errores de mecanizado en la pieza. A partir de lo anterior se realiza la calibración de la herramienta a través del software *MotoCalV EG* como se ilustra en el Anexo D, el cual proporciona datos más precisos que permiten reducir el efecto de excentricidad. Sin embargo, por tratarse de superficies arbitrarias no es posible estimar con exactitud si la pieza resultante corresponde al CAD modelado, o si ésta se ve afectada por la calibración. Por ello, se hace uso de una superficie esférica.

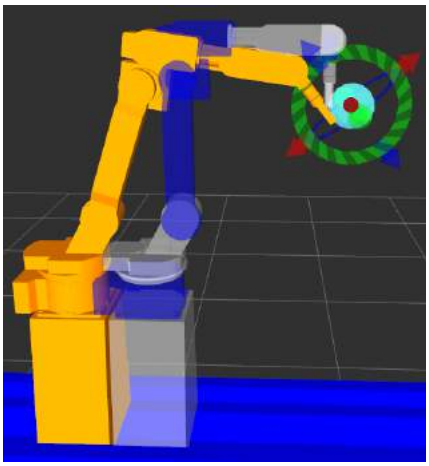


Figura 4-14.: Movimiento excéntrico alrededor del TCP

4.4. Prueba experimental para el mecanizado de una superficie esférica como forma libre con ajuste en la orientación de la herramienta

Se sugiere el uso de una superficie esférica para ser mecanizada considerando cambio de orientación en la herramienta e incluyendo 7 DOF. Lo anterior con el interés de validar el comportamiento del sistema cuando la pieza corresponde a una forma regular. Sin embargo, la ejecución de la trayectoria genera igualmente algunos movimientos fuertes que conllevan a que la herramienta se incruste en la pieza.

Se plantea una prueba para obtener solución de cinemática inversa mediante una optimización de las velocidades de articulación de acuerdo con [Flacco and Luca, 2014], a partir de la ecuación 4-1, donde Δq corresponde al incremento angular de cada articulación y Δx

describe el desplazamiento cartesiano deseado para el TCP del robot y J el Jacobiano del robot.

$$\Delta q = J^T \left(JJ^T\right)^{-1} \Delta x = J^{\#} \Delta x \tag{4-1}$$

La matriz $J^{\#} = J^T \left(JJ^T\right)^{-1}$ corresponde a la pseudo inversa *Moore-Penrose* de la matriz Jacobiana, dado que $JJ^{\#} = I$.

Experimentalmente se alcanza un movimiento suavizado para cada articulación del robot, pero se obtiene una diferencia de posicionamiento aproximada de $1mm$ con respecto a la posición objetivo, especialmente cuando se involucra mayor desplazamiento de la guía lineal. Debido a que esta estrategia presenta menor precisión que el planteamiento inicial, es descartada para la aplicación.

Con el fin de obtener definitivamente un movimiento suavizado a partir de la generación de las trayectorias, se establece un nuevo incremento en la resolución del desplazamiento para la herramienta durante la ejecución del mecanizado hasta $0{,}1mm$. Esta modificación proporciona resultados satisfactorios pues favorece el desarrollo de unos movimientos suaves en los ejes del conjunto robótico, con mayor control de la trayectoria y alcanzando precisión de posicionamiento en centésimas de milímetro.

La Figura **4-15** presenta la superficie esférica a ser mecanizada e incluye la correspondiente trayectoria generada para el proceso. En este caso se evidencia a diferencia de las pruebas anteriores que los movimentos definidos para la herramienta no son aplicados en dirección vertical con relación a la pieza, sino que se inclinan de acuerdo a la normal calculada para cada punto en la superficie. Se aprecia igualmente que los movimientos de retracción para posicionar la heramienta se llevan a cabo en la misma dirección.

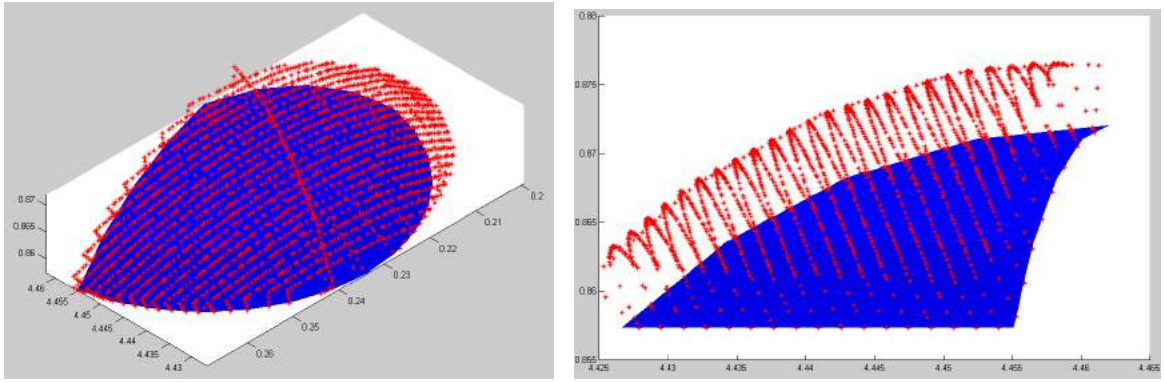


Figura 4-15.: Trayectoria generada para la superficie esférica

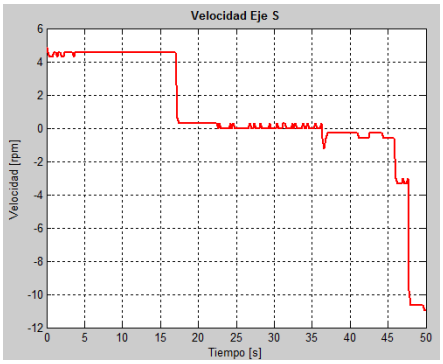
Es importante señalar que dependiendo del tamaño de cada superficie de mecanizado, la densidad de puntos obtenidos para la trayectoria varía, al igual que sucede con el tiempo requerido para planear el movimiento del robot hacia cada uno de los puntos de acuerdo a la convergencia del algoritmo de planeación. En general, se aprecia que el tiempo estimado para efectuar este cálculo cuando no se requiere esquivar ningún obstáculo, oscila alrededor de 4 centésimas de segundo.

El mecanizado de la superficie esférica ajustando la orientación de la herramienta favorece la obtención de un mejor acabado en la pieza, teniendo en cuenta que se hace uso de una herramienta de punta plana. La Figura **4-16** presenta la pieza obtenida en el proceso.

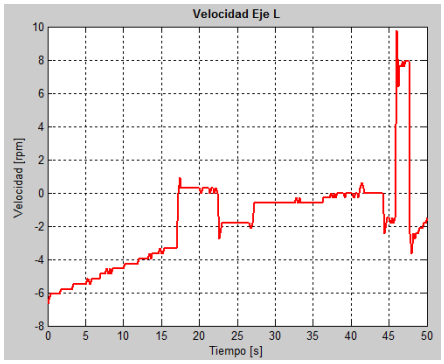
Figura 4-16.: Mecanizado de superficie esférica

La velocidad desarrollada por cada uno de los ejes resalta una notable reducción en comparación con las pruebas preliminares. La Figura **4-17** incluye las gráficas de velocidad para este caso. Específicamente la velocidad en los motores de los ejes S (Figura **4-17**a), L (Figura **4-17**b), U (Figura **4-17**c) y B1 (Figura **4-17**g) se encuentra por debajo de 15 *rpm*. Adicionalmente, el eje R (Figura **4-17**d) experimenta la disminución más drástica hasta menos de 6 *rpm*, al igual que el eje T (Figura **4-17**f). La velocidad más alta para este caso es alcanzada por la articulación B (Figura **4-17**e) con menos de 25 *rpm*. A partir de lo anterior, el movimiento alcanzado durante el proceso es bastante suavizado; facilitando de esta manera un mejor control de la trayectoria del robot, sin la aparición de alarmas y finalmente una mejor calidad en el mecanizado.

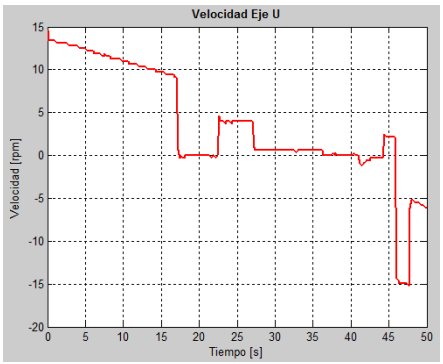
Por otra parte, en este caso el comportamiento de los torques no varía significativamente, pues en general se mantienen en los mismos rangos de magnitud a los registrados previamente, como se aprecia en la Figura **4-18**. En contraste, el eje R (Figura **4-18**d) es el único que evidencia un cambio importante, registrando valores máximos aproximados del 20 %.



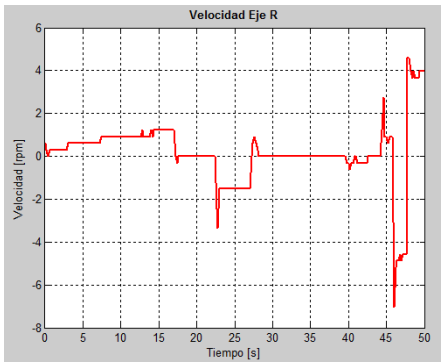
(a) Velocidad Eje S



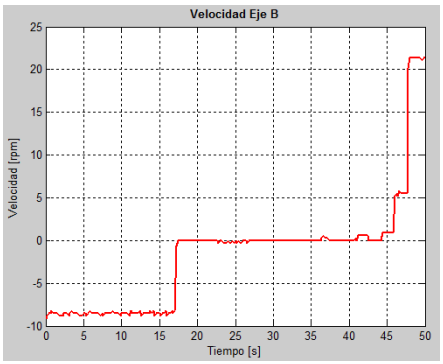
(b) Velocidad Eje L



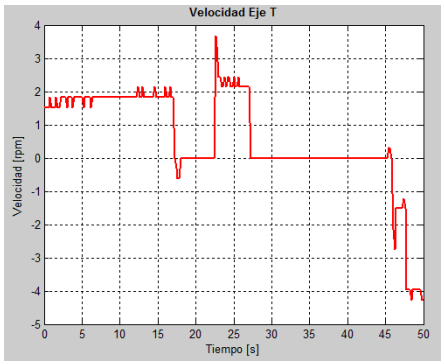
(c) Velocidad Eje U



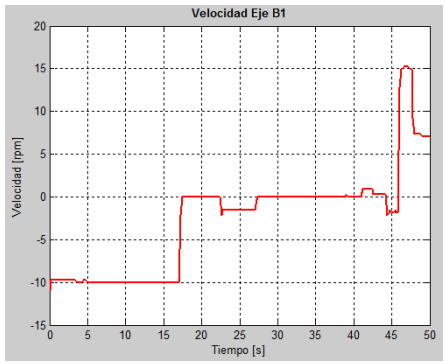
(d) Velocidad Eje R



(e) Velocidad Eje B

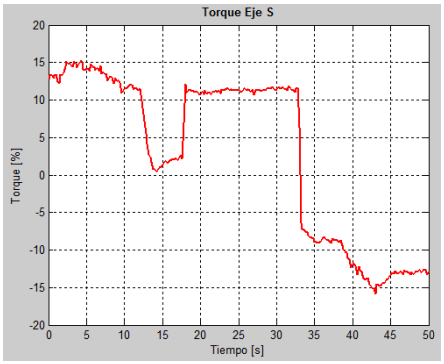


(f) Velocidad Eje T

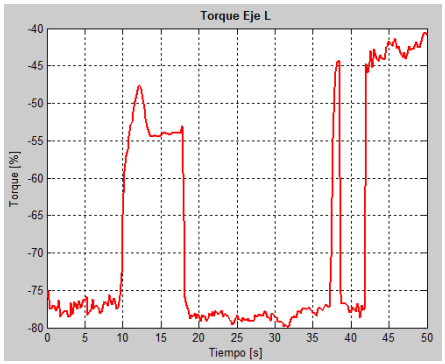


(g) Velocidad Eje B1

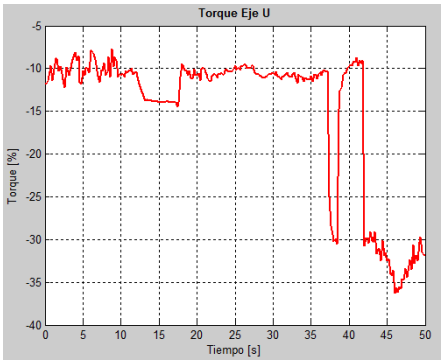
Figura 4-17.: Velocidades obtenidas durante la trayectoria esférica



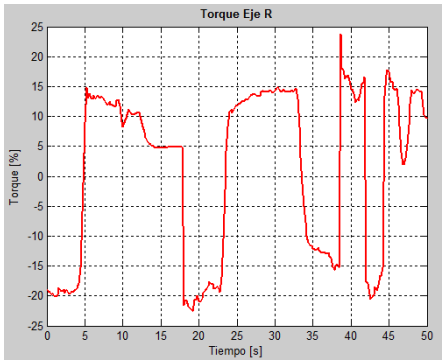
(a) Torque Eje S



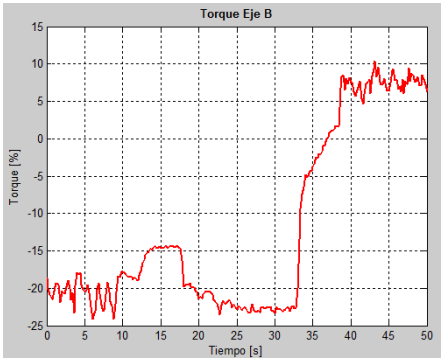
(b) Torque Eje L



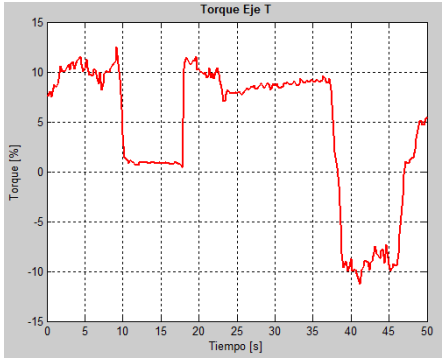
(c) Torque Eje U



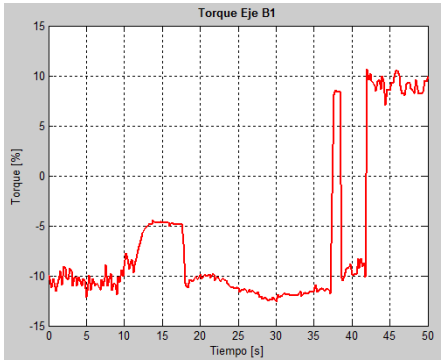
(d) Torque Eje R



(e) Torque Eje B



(f) Torque Eje T



(g) Torque Eje B1

Figura 4-18.: Torques obtenidos durante la trayectoria esférica

Considerando las zonas sugeridas en [Lin et al., 2017] y [Klimchik et al., 2017] para ubicación óptima de las piezas de trabajo en los procesos de mecanizado robotizado, es importante señalar que el uso del posicionador rotacional impide satisfacer este criterio, debido a la distribución física de los elementos en el laboratorio. Pues al llevarse a cabo aplicaciones que incluyen este eje externo, implica que el robot debe trabajar en la zona izquierda-posterior de su espacio alcanzable. Igualmente dependiendo de la ubicación definida para la pieza, el movimiento de la guía lineal puede generar que la zona de trabajo varíe durante la ejecución.

4.5. Prueba experimental de mecanizado mediante el uso de los 8 ejes

Finalmente se incluye el posicionador para llevar a cabo la ejecución de trayectorias utilizando los ocho ejes del sistema robótico. En contraste con las pruebas realizadas hasta el momento, las trayectorias generadas han sido tratadas en un dominio cartesiano; es decir, estableciendo un movimiento definido mediante la pose (posición y orientación) para el TCP del robot (*arm_on_rail*) en cada instante de la trayectoria, con respecto a un marco de referencia fijo en el entorno virtual.

Este método para el comando de movimientos es válido cuando el grupo correspondiente se encuentra en su conjunto conformado por una cadena cinemática. Sin embargo, la inclusión del posicionador conlleva a que el sistema robótico no pueda ser considerado de este modo. Por dicha razón, su uso directo conlleva a la aparición de errores tras el envío de cualquier trayectoria.

Debido a lo anterior, se sugiere la implementación de una etapa adicional encargada de calcular la cinemática inversa para cada punto de la trayectoria una vez ha sido generada. Esto permite definir y comandar de forma independiente la posición de cada una de las articulaciones que conforman el grupo de control, sin importar su descripción.

4.5.1. Movimiento sincronizado de los ejes

La prueba inicial que se lleva a cabo en la integración de los 8 ejes es la ejecución de un movimiento sincronizado. En general, este tipo de movimiento se encuentra definido mediante una estructura *maestro - esclavo*, tal que un cambio en la posición del maestro provoca que el esclavo se ajuste para mantener constante la pose relativa entre ellos y obtener así un efecto de seguimiento, de manera que para este caso en particular el maestro corresponde al posicionador, mientras que el esclavo al conjunto de la guía lineal y el robot.

La pose correspondiente para el conjunto del robot y la guía lineal dependiendo de la rotación

aplicada al posicionador se obtiene a partir del cálculo cinemático de la transformación relativa entre el TCP del robot con respecto al marco de referencia asociado al posicionador y de este último con respecto a la guía lineal, de acuerdo con la ecuación 4-2.

$$ {}^{4}_{1}T = {}^{2}_{1}T \, {}^{3}_{2}T \, {}^{4}_{3}T \tag{4-2} $$

Siendo ${}^{4}_{1}T$ la matriz de transformación para la pose ajustada del TCP (*arm_link_tool0*), ${}^{2}_{1}T$ la transformación de un marco de referencia fijo en el posicionador (*station_link*) con respecto a la base de la guía lineal (*base_rail_link*), ${}^{3}_{2}T$ describe la rotación aplicada al octavo eje (*link_s1*) con respecto a su marco de referencia fijo y finalmente ${}^{4}_{3}T$ corresponde a la pose relativa entre el TCP y el marco de referencia rotacional, como se presenta en la Figura **4-19**.

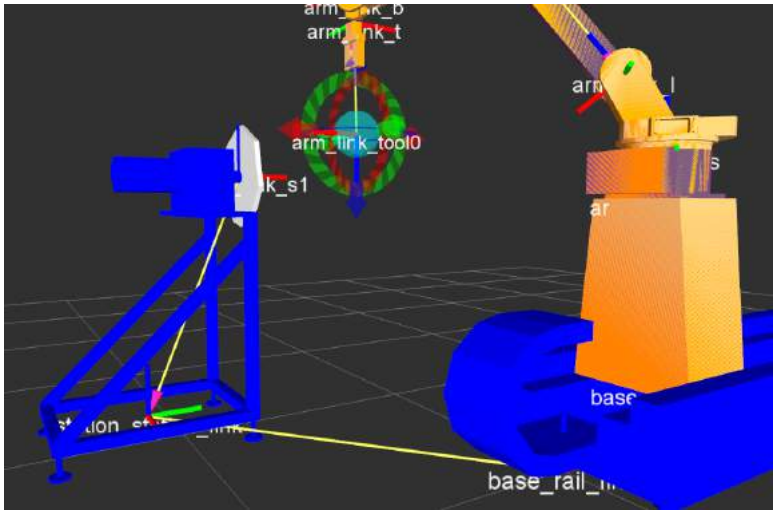


Figura 4-19.: Marcos de referencia del posicionador y robot

Teniendo en cuenta que para ${}^{2}_{1}T$ la información es constante y dependiente de la ubicación física del posicionador con respecto a la guía lineal, entonces la expresión resultante de la ecuación 4-2 para este caso en particular se presenta en la ecuación 4-3.

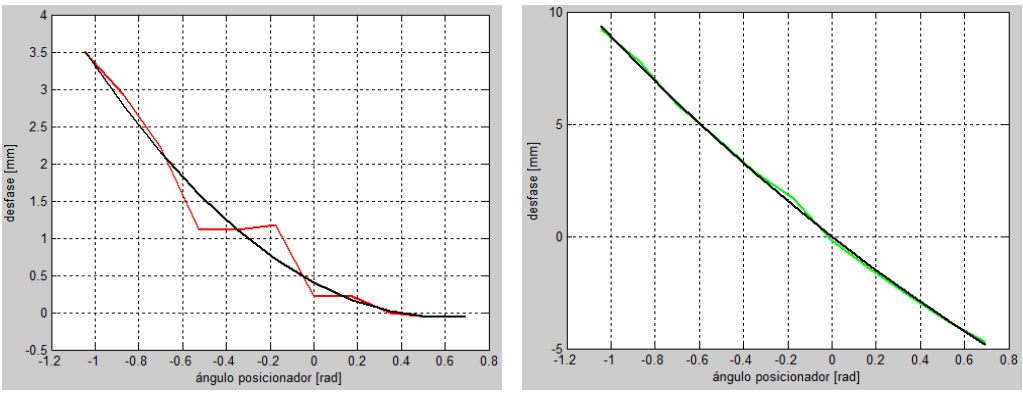
$$ {}^{4}_{1}T = \begin{bmatrix} 1 & 0 & 0 & -0{,}7553 \\ 0 & cos\theta & -sin\theta & 0{,}9886 \\ 0 & sin\theta & cos\theta & 1{,}0066 \\ 0 & 0 & 0 & 1 \end{bmatrix} {}^{4}_{3}T \tag{4-3} $$

Con θ correspondiente al ángulo aplicado al posicionador y ${}^{4}_{3}T$ calculado a partir de la trayectoria generada para el TCP mediante el uso de la librería *tf* de ROS [Foote, 2013].

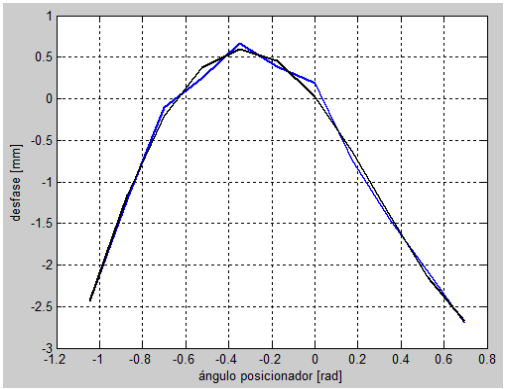
Finalmente de la matriz de transformación ${}^{4}_{1}T$ se extraen las componentes de posición y orientación para el cálculo de la cinemática inversa y obtener la nueva posición en articulaciones para el sistema robótico.

4.5.2. Error de alineación

A partir del movimiento sincronizado en todos los ejes, se evidencia que al comandar el desplazamiento del TCP para obtener seguimiento de un punto de referencia definido en la pieza que ha sido fijada al posicionador, es generado un desfase que se incrementa progresivamente a medida que varía el ángulo de giro para el eje. Lo anterior sugiere una desalineación entre el eje de rotación del posicionador con respecto a la guía lineal, situación surgida a partir del propio anclaje de los dispositivos físicos. Debido a que el error de alineación involucra un efecto no deseado en el comportamiento del sistema, se lleva a cabo una medición experimental del desfase generado en un rango de giro del posicionador, utilizando la realimentación de posición del propio robot. A partir de lo anterior se obtiene una descripción del error en cada dirección XYZ, como se aprecia en la Figura **4-20**. La información presentada sugiere que el mayor desfase se encuentra a lo largo del eje Y.



(a) Error en la dirección X (b) Error en la dirección Y



(c) Error en la dirección Z

Figura 4-20.: Desfase de posicionamiento del TCP debido a la rotación aplicada al posicionador

De acuerdo al error en cada uno de los ejes XYZ, la curva respectiva es aproximada mediante una función polinomial definida de acuerdo a las ecuaciones 4-4, 4-5 y 4-6.

$$error\_X = 1{,}3334s_1^2 - 1{,}5678s_1 + 0{,}4018 \quad (4\text{-}4)$$

$$error\_Y = 1{,}158s_1^2 - 7{,}7285s_1 - 0{,}0096 \quad (4\text{-}5)$$

$$error\_Z = 1{,}4086s_1^4 + 3{,}0528s_1^3 - 3{,}6942s_1^2 - 3{,}2571s_1 + 0{,}0195 \quad (4\text{-}6)$$

Estas compensaciones son aplicadas a cada punto calculado de la trayectoria y posteriormente procesadas de acuerdo a lo descrito en la sección anterior. En la Tabla **4-2** se registra un estimador del error de alineación antes y después de la compensación.

Tabla 4-2.: Compensación del error de alineación

Estimador	**Condición inicial** [mm]	**Modelo compensado** [mm]
RMSE_X	1,6410	0,2133
RMSE_Y	4,8219	0,1481
RMSE_Z	1,4262	0,0908

4.5.3. Mecanizado helicoidal

Inicialmente es calculada la trayectoria para una superficie con forma helicoidal de acuerdo a las ecuaciones 3-1, 3-2 y 3-3. Los movimientos son comandados de forma independiente para cada eje del sistema robótico, considerando un valor constante para el ángulo del posicionador, con el fin de obtener una idea del mecanizado sin considerar el error de alineación.

La trayectoria generada describe un movimiento continuo con paso de $1mm$ en la dirección x, la cual conlleva al mecanizado de una sección cilíndrica en la pieza. La Figura **4-21** muestra la trayectoria helicoidal, considerando un ángulo igual a 0 rad para el posicionador.

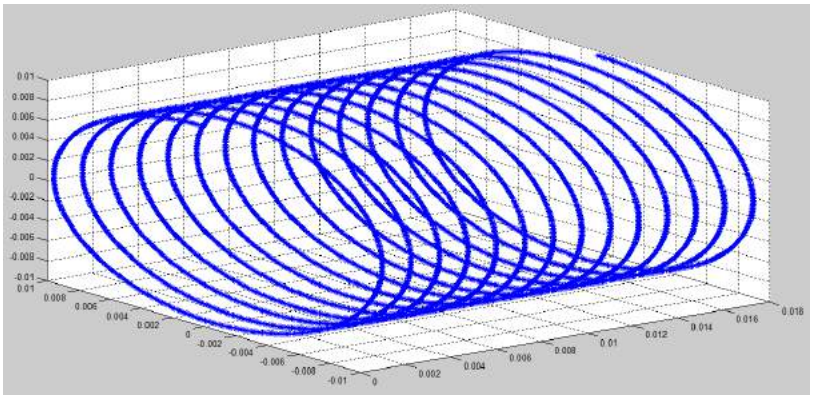


Figura 4-21.: Trayectoria de mecanizado helicoidal

Debido a que la trayectoria se define centrada en la dirección Z con respecto a la cara de la pieza, el resultado es una superficie cóncava de acuerdo a la Figura **4-22**, resaltando una buena calidad obtenida para el mecanizado.

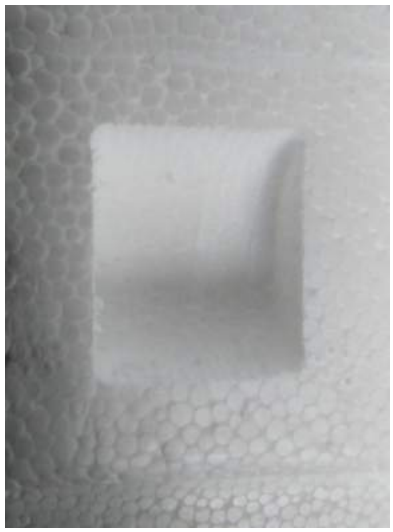


Figura 4-22.: Mecanizado helicoidal de la pieza

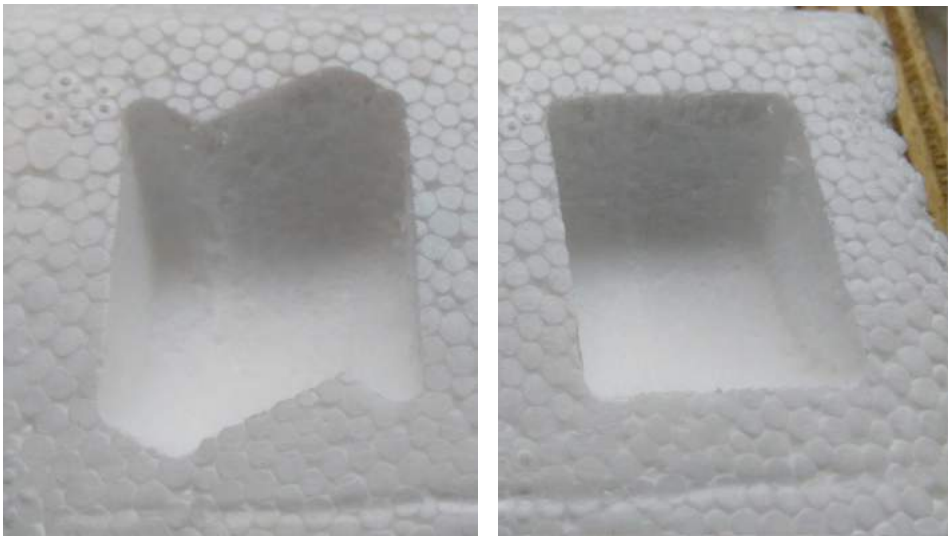
4.5.4. Compensación del error

A partir de la trayectoria helicoidal anterior, se lleva a cabo un nuevo mecanizado aplicando movimientos de rotación al posicionador en el rango $\pm 0{,}35\ rad$; pues debido a la disposición física del robot, en esta zona se puede alcanzar cualquier orientación para la herramienta sin presentar inconvenientes debido a límites de recorrido. Esta prueba se efectúa sin aplicar ninguna compensación al error de seguimiento por efecto del posicionador, con el interés de obtener un comparativo con el caso ideal en el cual no hay rotación en el octavo eje.

La pieza obtenida presenta una desviación pronunciada en la dirección Y, a la vez que incluye una serie de altibajos en la superficie mecanizada.

Finalmente, son aplicadas las aproximaciones polinomiales definidas en las ecuaciones 4-4, 4-5 y 4-6 para compensar el error de posicionamiento generado por la rotación, y validar que de esta manera la superficie mecanizada se asemeja a la condición inicial.

En la Figura **4-23**a se presenta el mecanizado resultante para el caso en el cual no se lleva a cabo compensación del error, mientras que la Figura **4-23**b ilustra el efecto de la compensación correspondiente. Para este último caso se evidencia que el sistema se comporta de acuerdo a lo esperado, dando lugar a un mecanizado similar al ideal.



(a) Sin compesación (b) Con compensación

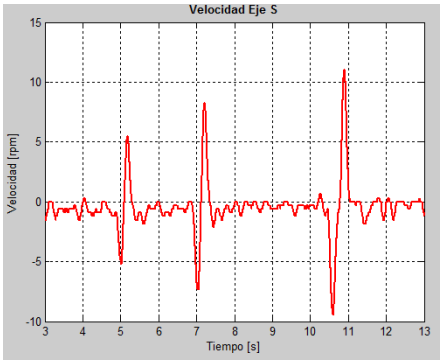
Figura 4-23.: Mecanizado resultante al incluir rotación en el posicionador

A partir de lo anterior, se tiene que el ajuste realizado genera unos resultados satisfactorios. Aunque de ser necesario podría efectuarse una aproximación más detallada especialmente en el eje Z.

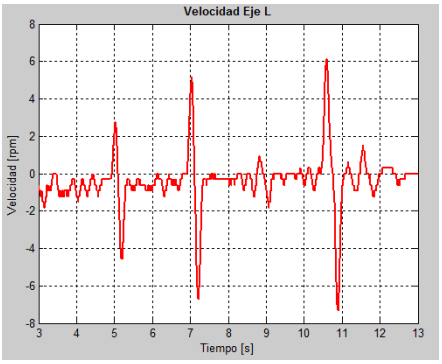
En este prueba se evidencia un comportamiento suavizado en las velocidades (Figura **4-24**) y torques (Figura **4-25**) experimentados por cada eje. Para el primer caso las velocidades más altas son alcanzadas por los ejes S (Figura **4-24**a) y B1 (Figura **4-24**g), debido a que junto con L (Figura **4-24**b) y U (Figura **4-24**c) ajustan la posición del TCP, mientras que las menores velocidades se obtienen para R (Figura **4-24**d), B (Figura **4-24**e) y S1 (Figura **4-24**h) con menos de 2 rpm.

Acorde con lo anterior, los torques que presentan mayor fluctuación corresponden al S (Figura **4-25**a) y B1 (Figura **4-25**g), mientras que para el posicionador (Figura **4-25**h) el valor registrado refleja solamente el sostenimiento de su propia estructura mecánica 13 %.

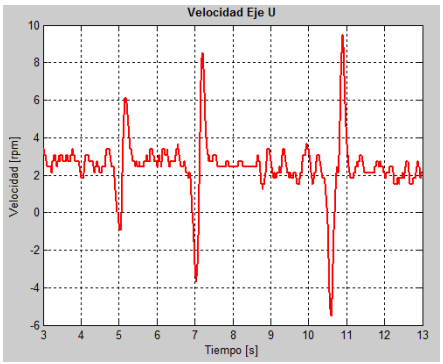
Considerando las pruebas realizadas, es importante señalar que dependiendo de la aplicación en particular, el sistema robótico permite la versatilidad de abordar diferentes soluciones, incluyendo un mecanizado solamente con el robot, incluyendo ya sea la guía lineal o el posicionador, o haciendo uso de los ocho ejes.



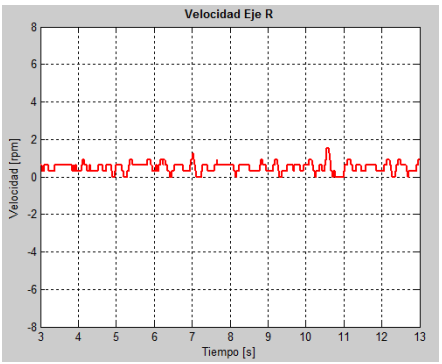
(a) Velocidad Eje S



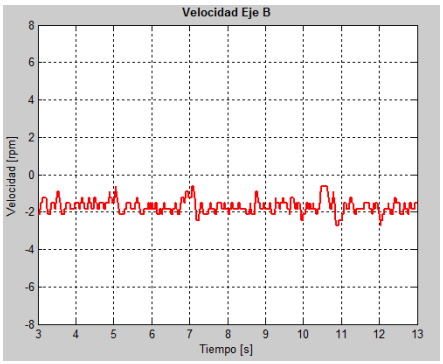
(b) Velocidad Eje L



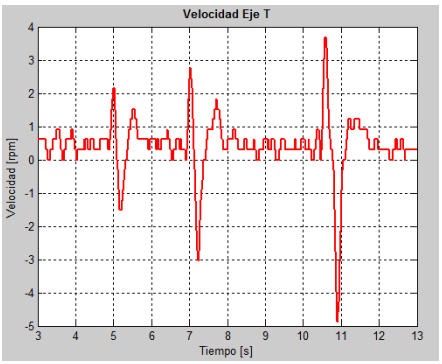
(c) Velocidad Eje U



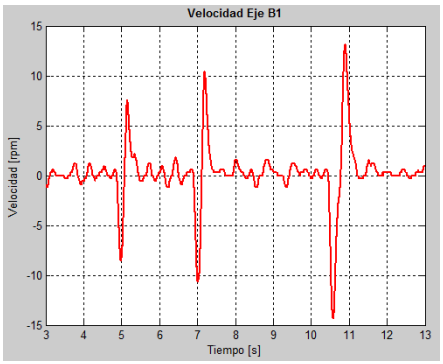
(d) Velocidad Eje R



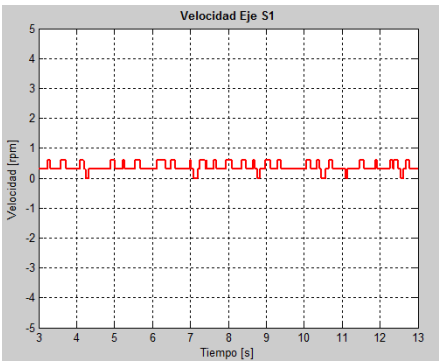
(e) Velocidad Eje B



(f) Velocidad Eje T

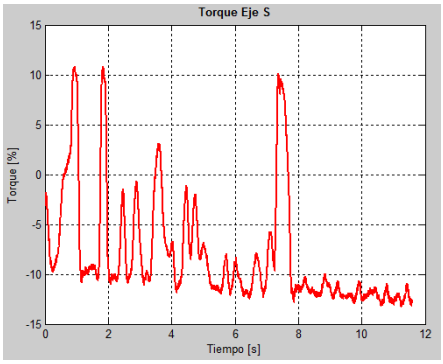


(g) Velocidad Eje B1

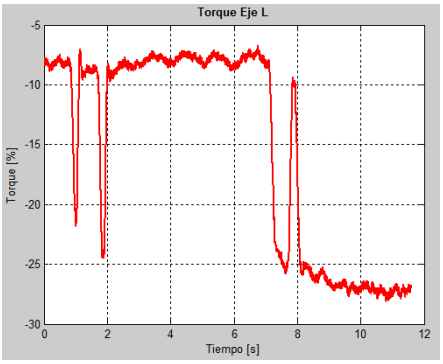


(h) Velocidad Eje S1

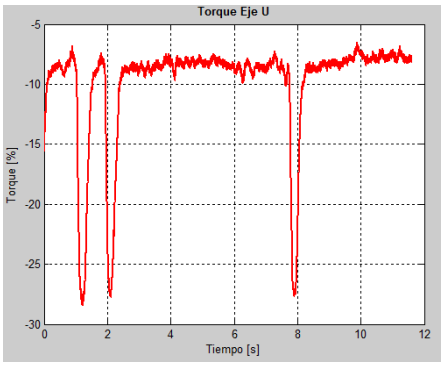
Figura 4-24.: Velocidades obtenidas durante el mecanizado helicoidal



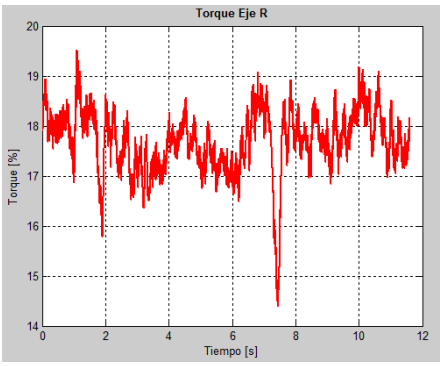
(a) Torque Eje S



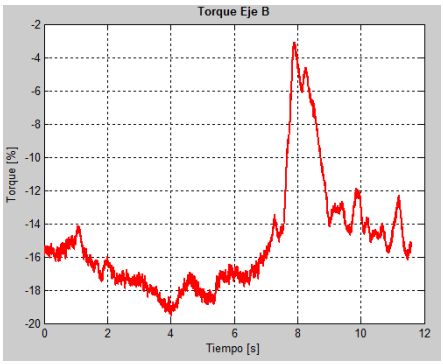
(b) Torque Eje L



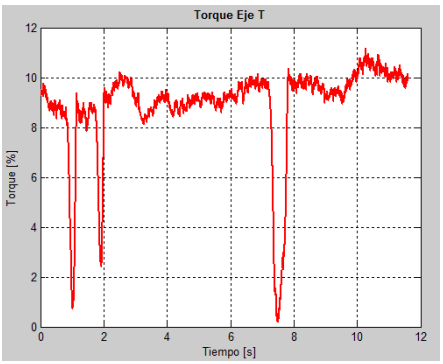
(c) Torque Eje U



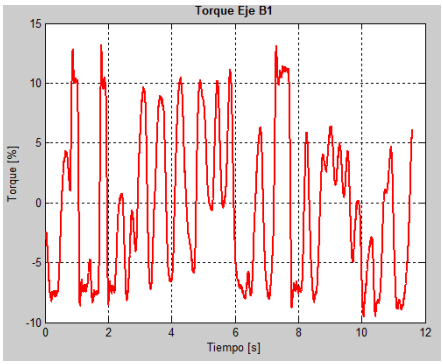
(d) Torque Eje R



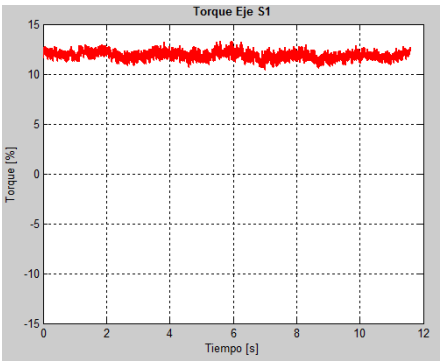
(e) Torque Eje B



(f) Torque Eje T



(g) Torque Eje B1



(h) Torque Eje S1

Figura 4-25.: Torques obtenidos durante el mecanizado helicoidal

4.5.5. Superficie convexa

En contraste con la superficie obtenida en la sección anterior, se realiza una modificación en la generación de la trayectoria, con el fin de obtener una superficie convexa para el mecanizado. La trayectoria obtenida es como se presenta en la Figura **4-26**. En este caso el movimiento también es continuo, pero el desplazamiento de la herramienta se lleva a cabo solamente sobre la sección superior de la trayectoria helicoidal completa.

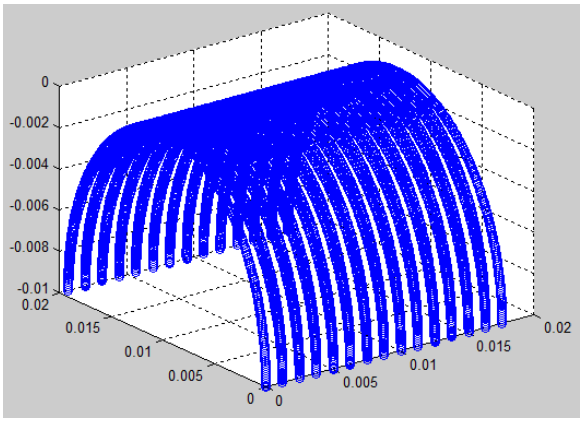


Figura 4-26.: Trayectoria de mecanizado helicoidal convexa

La pieza obtenida corresponde a la presentada en la Figura **4-27**. Cabe señalar que para esta condición el mecanizado resultante evidencia una menor calidad que en la forma cóncava. Esta situación puede ser un efecto producido debido a que la trayectoria generada no tiene en consideración la compensación del radio de la herramienta.

Figura 4-27.: Pieza mecanizada helicoidal convexa

4.6. Prueba experimental para el mecanizado de planos oblicuos

A partir de una entidad constructiva básica y correspondientes transformaciones de posición y orientación se lleva a cabo la generación de trayectorias para planos oblicuos. En este caso particular se hace uso de formas triangulares que definen la superficie deseada, como se aprecia en la Figura **4-28**

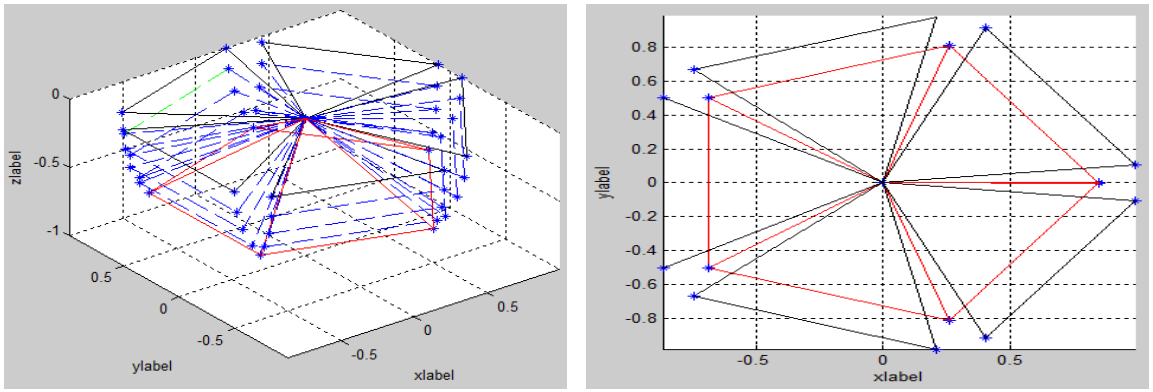
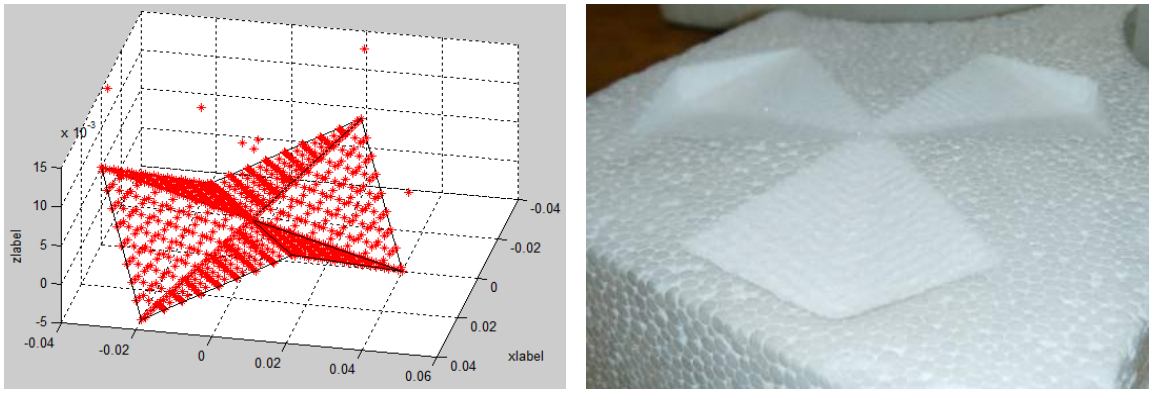


Figura 4-28.: Generación de planos oblicuos

La trayectoria resultante se presenta en la Figura **4-29**a, mientras que la superficie mecanizada se aprecia en la Figura **4-29**b.

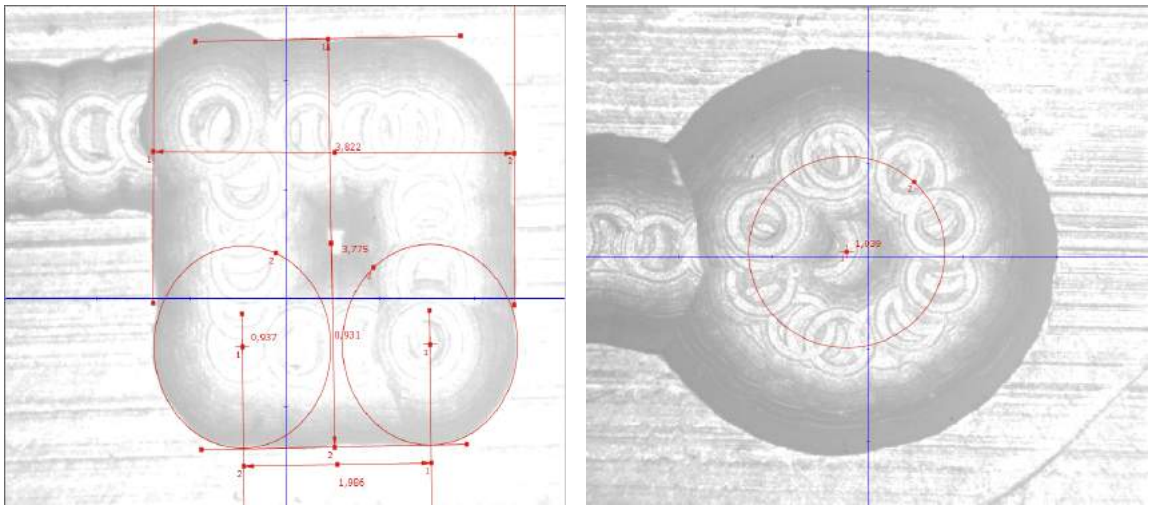


(a) (b)

Figura 4-29.: Mecanizado de planos oblicuos

4.7. Medición de precisión en mecanizado

Finalmente, con el objetivo de estimar la precisión alcanzada en el proceso de mecanizado con el sistema robótico de 8 DOF se plantea la fabricación de piezas de prueba para medición con cámara de precisión disponible en la máquina *Zoller Smile 400*. Inicialmente se obtienen formas cuadradas (Figura **4-30**a) y circulares (Figura **4-30**b) en el plano XY



(a) Cuadrado $l = 2mm$ (b) Círculo $r = 1mm$

Figura 4-30.: Mecanizado de piezas de prueba en plano XY

Por otra parte, se genera un mecanizado escalonado en el plano YZ para efectuar mediciones de precisión en la dirección Z. La pieza resultante presenta un escalón de altura $1mm$, como se aprecia en la Figura **4-31**.

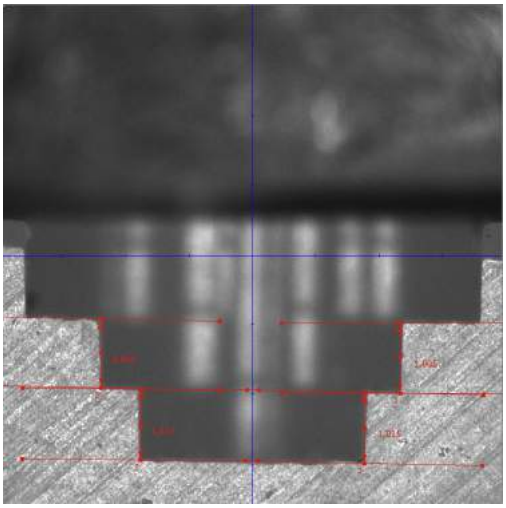


Figura 4-31.: Mecanizado de piezas de prueba en dirección Z

Las mediciones obtenidas son registradas en la Tabla **4-3**, resaltando una precisión en el orden de $23\mu m$.

Tabla 4-3.: Mediciones de precisión en mecanizado

Estimador	**Cuadrado** [mm]	**Círculo** [mm]	**Escalón** [mm]
Media	0,0012	0,0055	0,0004
Mediana	0,0115	0,01	0,003
Desviación estándar	0,0235	0,0162	0,0141
RMSE	0,0232	0,0165	0,0140

4.8. Prueba experimental de interacción entre Motoman y AGV

Para la prueba de interacción entre el robot Motoman y el AGV, este último se desplaza en el laboratorio a partir de su algoritmo de navegación con sensor láser, mientras publica su pose estimada. A partir de esta información, el PC del Motoman evalúa si se encuentra dentro de su espacio de trabajo alcanzable, para posteriormente planear una trayectoria y finalmente comandar el respectivo movimiento. En los casos en los cuales el AGV se encuentra fuera de alcance, no es posible calcular la cinemática inversa para la posición objetivo y en consecuencia el Motoman permanece inmóvil hasta que el vehículo se hace alcanzable.

En contraste con las pruebas de mecanizado, para este caso debe efectuarse continuamente una planeación de trayectoria del robot Motoman hacia la ubicación en cada instante del AGV. Adicionalmente, debido a que una vez ha sido definida la posición objetivo para el manipulador y comandado el correspondiente movimiento, no es posible enviar nuevas instrucciones hasta que se ha alcanzado dicho objetivo, pues esta condición conlleva a la aparición de errores.

Debido a lo anterior; cada vez que el robot Motoman se desplaza hacia la posición recibida, el movimiento constante del AGV implica que éste ha cambiado de ubicación, por lo cual el Motoman debe llevar a cabo una nueva planeación y ejecución de trayectoria, como en la Figura **4-32**.
Esta situación genera un efecto de seguimiento intermitente, que se hace inevitable debido al comportamiento dinámico que presenta el objetivo. A partir de lo anterior, se incluyen dos estrategias que permiten reducir un poco la condición descrita:

1. Para el instante en el cual se define la posición objetivo, la ubicación del AGV (Punto C) es comparada con respecto a la posición actual del TCP (Punto A). Si espacialmente

Figura 4-32.: Prueba experimental Motoman - AGV

estos se encuentran alejados más de $3cms$ (rango de ubicación del AGV), el objetivo es definido como una posición intermedia (Punto B) de acuerdo con la Figura **4-33**. Esto facilita por una parte, que el robot alcance el punto definido más rápido y pueda desplazarse hacia la nueva ubicación del AGV a través de segmentos de camino más corto.

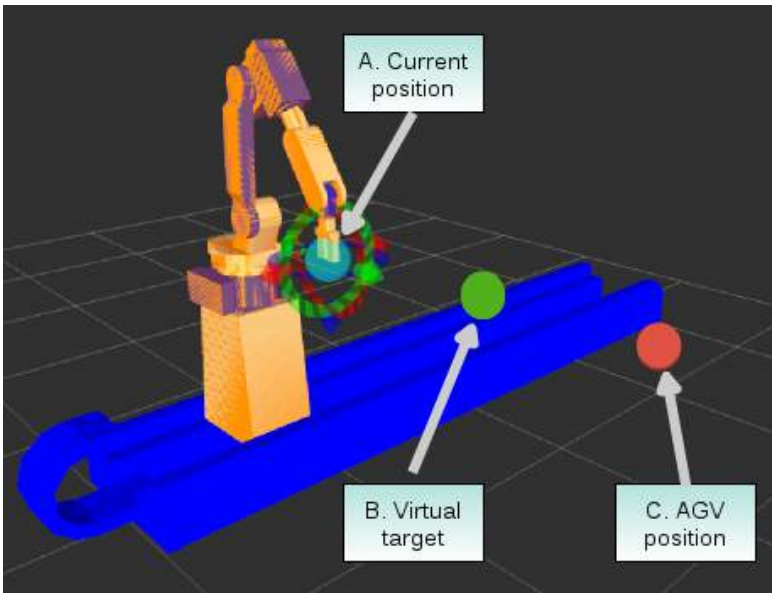


Figura 4-33.: Aproximación virtual Motoman - AGV

2. Mientras el robot Motoman se encuentra en movimiento hacia el objetivo virtual (de A a B), una planeación de trayectoria es realizada considerando esta posición y la

ubicación del AGV en cada instante (de B a C), pero sin comandar su ejecución. De esta manera, cuando el Motoman ha alcanzado la posición correspondiente, una nueva trayectoria ya ha sido calculada y se encuentra lista para transmitirse al controlador.

Las estrategias descritas favorecen un movimiento menos intermitente, pero definitivamente el efecto de desaceleración al acercarse al objetivo y la posterior aceleración en la ejecución de la nueva trayectoria, generan visualmente una pausa en el desplazamiento del robot Motoman. A través del movimiento instantáneo comandado hacia el robot se obtienen los perfiles de velocidad desarrollados para cada articulación y se presentan en la Figura **4-34**. Se evidencian condiciones de velocidad inicial y final igual a cero en los puntos objetivo.

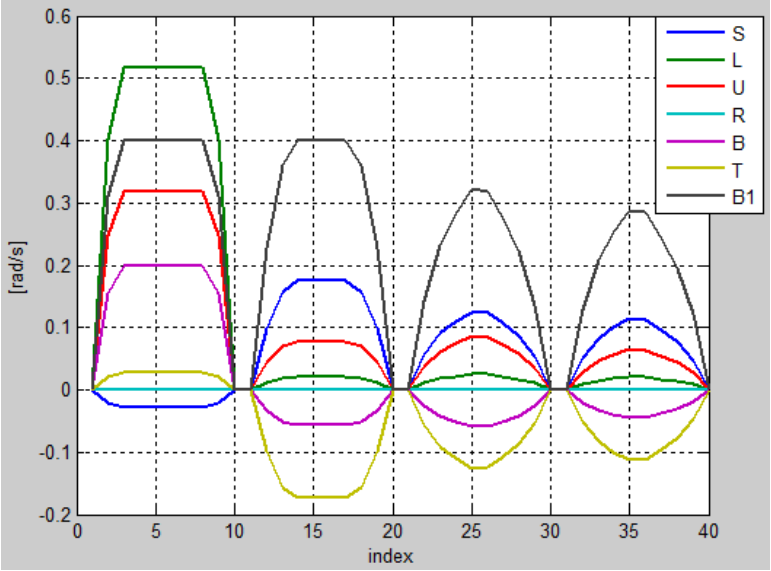


Figura 4-34.: Perfiles de velocidad del robot Motoman

Considerando el conjunto de pruebas experimentales descritas en este capítulo, se evidencia que éstas no pueden ser llevadas a cabo mediante el uso de programación convencional debido a la complejidad de las tareas y la inclusión de condiciones de comunicación con otros robots.

5. Conclusiones y Trabajo futuro

5.1. Conclusiones

La ejecución de movimientos sincronizados para los ocho ejes con los que cuenta el sistema robótico es una aplicación única desarrollada para el laboratorio LabFabEx, de forma que incluso el propio controlador de Motoman no es capaz de efectuar una coordinación estricta de estos elementos. De hecho, la sincronización alcanzada por defecto incluye únicamente el robot y el posicionador rotacional, ignorando cualquier intervención de la guía lineal. Para efectos funcionales, se obtiene que su inclusión favorece el ajuste de la posición del robot en el intento de alcanzar su objetivo y de esta manera evitar ocasionalmente la aparición de límites de software en sus ejes. Por otra parte, no se encuentra registro en la literatura acerca de aplicaciones robóticas desarrolladas con un conjunto de este tipo, por lo cual se abre la puerta a proyectos de investigación novedosos.

El uso de los 8 DOF del sistema para aplicaciones de mecanizado permite comandar el proceso definiendo prácticamente cualquier orientación para la herramienta en un rango de trabajo extendido. Particularmente, la estrategia de posicionar el eje de la herramienta de forma normal a la superficie en cada instante durante la trayectoria de maquinado proporciona una buena calidad en la superficie resultante incluso utilizando una herramienta plana. Adicionalmente, esta investigación junto con la versatilidad de los múltiples ejes, facilita el mecanizado de geometrías altamente complejas y de mayor dimensión de forma casi automática, a la vez que permite que un mismo problema sea abordado de diferentes maneras; tales como, un mecanizado utilizando 7 ejes (robot + guía lineal), o la inclusión del octavo eje para fines de interpolación o de indexación. Lo anterior puede sugerir un campo de trabajo muy interesante en la exploración de la fabricación de prototipos usando técnicas de remoción de material.

Pese a que el robot Motoman MH6 no es un modelo concebido para llevar a cabo procesos de mecanizado, las mediciones obtenidas para los torques durante las pruebas experimentales de este trabajo reflejan la acción de un esfuerzo reducido sobre las articulaciones del sistema, lo cual sugiere la posibilidad de explorar el uso de materiales que incluyan mejores propiedades de maquinabilidad sin afectar la integridad del equipo. Específicamente las mayores dificultades se encuentran dirigidas hacia un control de velocidad, particularmente para el movimiento de la guía lineal, pues éste genera los movimientos más fuertes para el TCP

debido al manejo de la inercia propia y del robot.

Los procedimientos convencionales de programación de robots (basados en métodos de enseñanza) son altamente eficientes en procesos industriales rígidos, los cuales se encuentran caracterizados por condiciones físicas invariantes que garantizan escenarios de continua repetibilidad. Cabe resaltar igualmente que la correspondiente programación se torna extensa en la medida en que la complejidad de la tarea incrementa. Por lo anterior, no constituye una estrategia eficiente cuando se trata de aplicaciones que requieren mayor versatilidad. En contraste, la herramienta desarrollada en esta investigación facilita el hecho de adaptarse a cambios de pieza de manera casi automática, a partir de modelos CAD así como de modelos matemáticos o geométricos. A la vez que le permite al sistema robótico la opción de comunicarse con agentes externos, en un enfoque hacia aplicaciones de robótica cooperativa.

La planeación de una trayectoria alcanzable y libre de colisiones para el robot es efectuada en un promedio de 0,04 *seg*. Para el caso de las aplicaciones de mecanizado abordadas; la alta densidad de puntos que definen la superficie y el propio proceso, hacen inapropiado aplicar la evasión de colisiones para cada segmento de la trayectoria generada. Lo anterior debido a que en ejecución se evidencia una discontinuidad del movimiento, presentada debido al retraso asociado a los cálculos cinemáticos, de planeación y de envío de la posición (cartesiana o por articulaciones) correspondiente para el robot. Esto sugiere que la planeación de trayectorias evadiendo posibles colisiones; para este caso, aplica hasta el instante en el que la pieza a mecanizar empieza a ser procesada. Una vez se da inicio a esta etapa, la trayectoria es calculada en su totalidad y enviada al controlador para su respectiva ejecución.

En otro tipo de aplicaciones como aquellas de interacción con dispositivos externos que se desplazan de manera autónoma en el laboratorio LabFabEx, es requerido que la trayectoria sea calculada en cada instante, debido a que se incluye una realimentación continua de la posición de referencia que le permita ajustar el objetivo y planear una nueva trayectoria hacia éste (lo que conlleva adicionalmente a que la evasión de colisiones sea considerada en cada planeación). Esta situación genera un inevitable retraso entre los movimientos del robot, a partir de lo cual deben abordarse diversas estrategias que permitan reducir su efecto. Específicamente en este trabajo se estructura experimentalmente la interacción del robot Motoman con un AGV abordando el desarrollo de aplicaciones robóticas colaborativas de forma dinámica, a partir del comportamiento obtenido por el vehículo y su correspondiente algoritmo de navegación.

A través de esta investigación se ha obtenido un modelo virtual para el sistema robótico implementado mediante ROS-I, el cual establece una base para el desarrollo de diversas aplicaciones a partir de la inclusión de herramientas de hardware (sensores, actuadores, robots) mediante el uso de ROS.

Las mediciones obtenidas a partir de las probetas mecanizadas sugieren que las piezas generadas presentan una condición de precisión dimensional en el orden de 0,03mm.

El robot Motoman es capaz de seguir de manera precisa al AGV. Sin embargo, este último presenta una incertidumbre de ubicación de 3cm y un algoritmo de control en construcción, por lo cual se induce un error en el proceso colaborativo.

5.2. Trabajo futuro

Es importante desarrollar un control de velocidad para el robot Motoman a través de ROS-I para facilitar el envío inmediato de las trayectorias de mecanizado así como la interacción con los AGV's, lo cual potencializa la plataforma de planeación de trayectorias en vuelo.

Mediante la herramienta de osciloscopio disponible en el controlador del robot se han llevado a cabo mediciones de los torques alcanzados por cada eje en las pruebas experimentales de mecanizado. Se plantea sin embargo la conveniencia de incluir instrumentos externos de medición que permitan evidenciar el efecto del proceso sobre el dispositivo fabricado para la herramienta, con el fin de conocer el comportamiento de todos los elementos involucrados. Adicionalmente, se recomienda extender el trabajo realizado en mecanizado incluyendo control de torque.

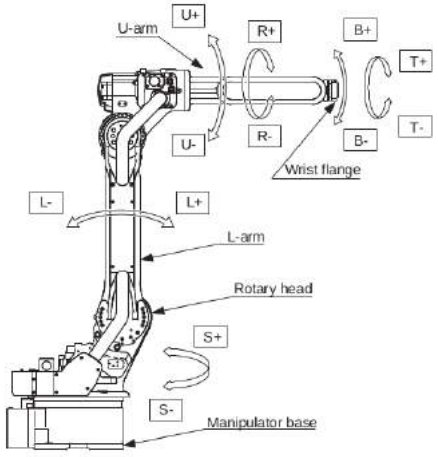
La generación de trayectorias para las aplicaciones de mecanizado se efectúa a partir de las intersecciones de la pieza con planos paralelos a YZ. Se sugiere evaluar la posibilidad de optimizar este método, de forma que los planos de corte sean definidos dependiendo de la geometría de la pieza, con el fin de reducir la aparición de movimientos excesivos en el robot o tiempo de mecanizado, entre otros.

A partir de la interacción entre el robot Motoman y un AGV, la cual ha sido desarrollada mediante una comunicación usando *websockets* plantea la realización de pruebas básicas para comandar el robot a través de una aplicación web. Lo anterior sugiere la posibilidad de integrar los resultados de este trabajo con las plataformas web del laboratorio, enfocado hacia las fábricas inteligentes.

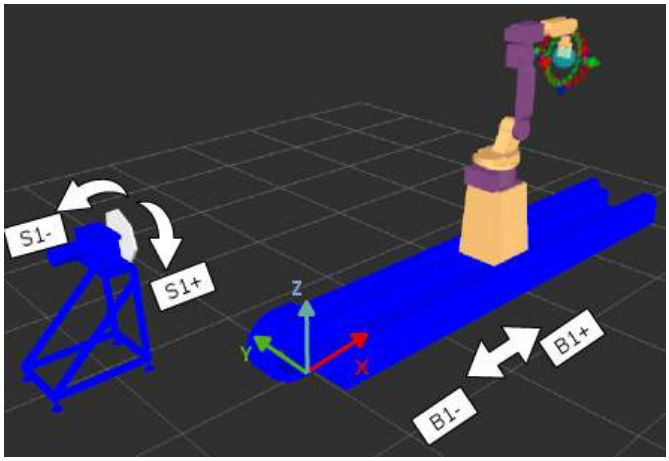
Extensión de la plataforma desarrollada hacia aplicaciones de impresión 3D.

A. Anexo: Representación de articulaciones y ejes cartesianos del sistema robótico

Se ilustran cada una de las articulaciones que conforman el sistema robótico. Éstas son definidas para el robot (S, L, U, R, B, T), guía lineal (B1) y posicionador rotacional (S1) e incluye la representación de los ejes cartesianos XYZ.



(a) Ejes del robot [Motoman, 2009]



(b) Ejes externos

Figura A-1.: Articulaciones del sistema robótico y ejes cartesianos

B. Anexo: Actualización de la versión de software del controlador y habilitación de la función MotoPlus

Para realizar la actualización de la versión del software es necesario configurar un dispositivo de almacenamiento externo (USB o Compact flash) que incluya el nuevo sistema operativo para el controlador y encenderlo mientras se presiona una secuencia de teclas en el *teach pendant*. La información en el dispositivo de almacenamiento es leída automáticamente y el equipo solicita confirmación de la operación. La Figura **B-1** muestra la pantalla de actualización, y resta simplemente esperar que el proceso se complete satisfactoriamente.

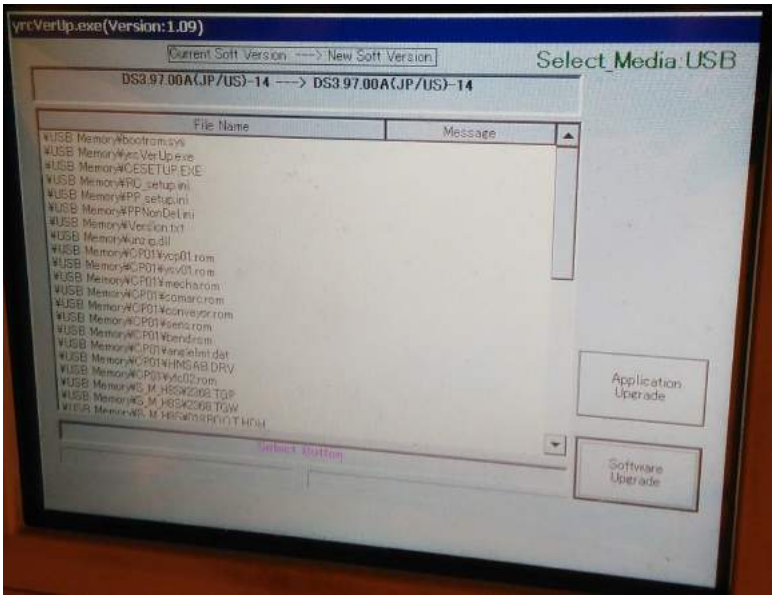


Figura B-1.: Actualización de la versión del controlador

Por otra parte, para la habilitación de MotoPlus debe iniciarse el controlador en modo mantenimiento y seleccionar la opción respectiva, como se aprecia en la Figura **B-2**.

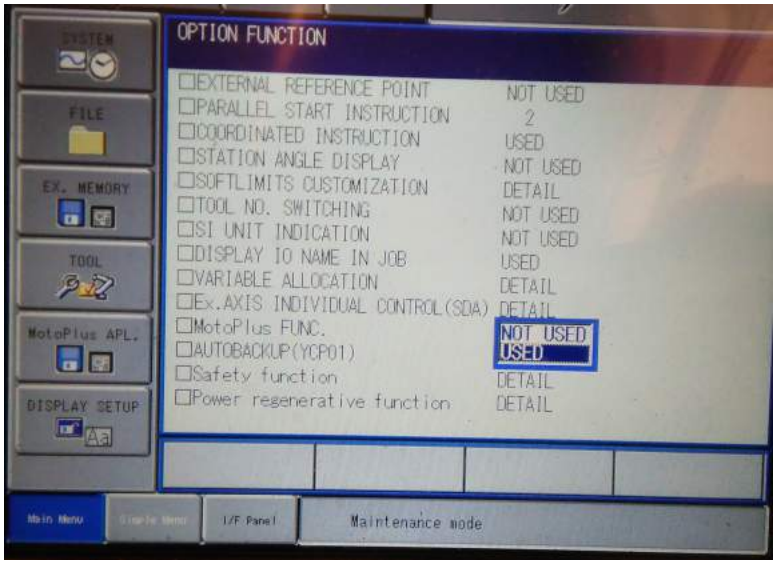


Figura B-2.: Habilitación de la función MotoPlus

C. Anexo: Torques máximos para cada articulación

La medición de torques a través del osciloscopio proporciona una información porcental, donde el 100 % para cada articulación corresponde a los datos referidos en la Tabla **C-1**.

Tabla C-1.: Equivalencia de torques

Eje	**Torque Máximo**
S	2,86 Nm
L	5,39 Nm
U	2,86 Nm
R	0,32 Nm
B	0,32 Nm
T	0,32 Nm
B1	8,34 Nm
S1	8,34 Nm

Esta información es obtenida a partir de los parámetros de configuración *SVM1G032-37* para el robot, *SVM2G032* para la guía lineal y *SVM3G032* para el posicionador.

D. Anexo: Calibración del TCP a través del software MotoCalV EG

Para la calibración del TCP usando MotoCalV EG se requiere la creación de un programa que incluya al menos 7 posturas diferentes en un punto de referencia, como en la Figura **D-1**.

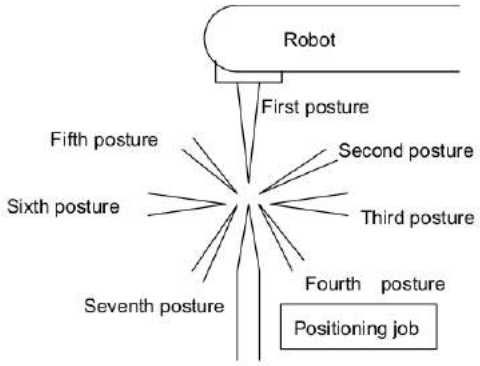


Figura D-1.: Enseñanza de un punto con 7 posturas

El archivo obtenido debe ser llevado al PC y en la opción de *Tool Calibration* del software seleccionarlo para ejecutar la calibración (ver Figura **D-2**)

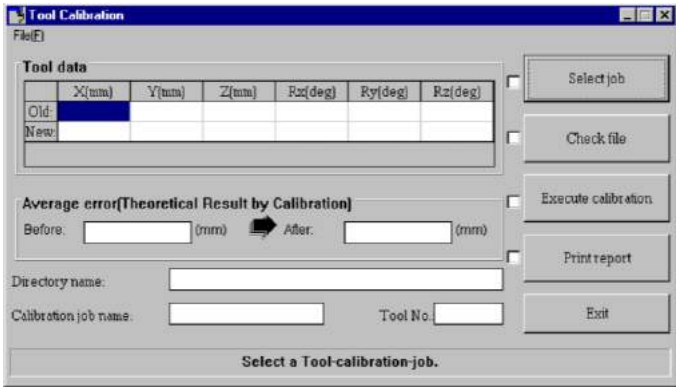


Figura D-2.: Ventana para la calibración de la herramienta

A partir de lo anterior, una nueva información acerca de la calibración de la herramienta es obtenida y debe ser copiada al controlador del robot.

E. Anexo: Modelo CAD del soporte fabricado para la herramienta

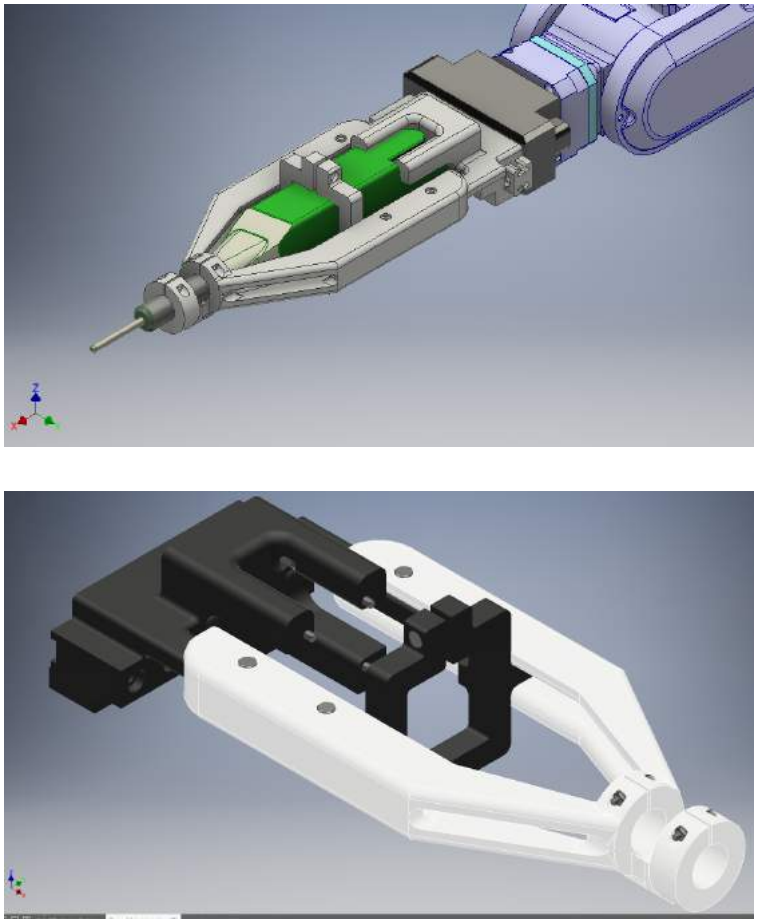


Figura E-1.: Modelo CAD del soporte fabricado

F. Anexo: Presentación en Congreso

congreso AMDM 2016

Universidad del Valle

9 años

Conocimiento que transforma vidas

CUMPLIDO

El Director del "Tercer Congreso Internacional sobre Tecnologías Avanzadas de Mecatrónica, Diseño y Manufactura" AMDM 2016, realizado en las instalaciones de la Universidad del Valle

HACE CONSTAR

Que la persona Julio Cesar García, con No. de documento de identidad 1.022.964.300 asistió y participó en las actividades académicas programadas en el "Tercer Congreso Internacional sobre Tecnologías Avanzadas de Mecatrónica, Diseño y Manufactura", con la ponencia "Planeación de Trayectorias en Vuelo de un Manipulador Industrial para el Laboratorio Fábrica Experimental", desarrollada durante los días miércoles 13, jueves 14 y viernes 15 de abril de 2016, en jornadas comprendidas desde las 8:00 am a 12:00 m y desde las 2:00 pm hasta las 6:00 pm.

Se firma en Santiago de Cali, a los quince (15) días del mes de Abril de 2016.

JOAO LUIS EALO CUELLO, Ph.D
Coordinador AMDM2016
Docente Escuela de Ingeniería Mecánica

Figura F-1.: Presentación en Congreso

Bibliografía

[Andersson et al., 2016] Andersson, N., Argyrou, A., Nägele, F., Ubis, F., Campos, U. E., de Zarate, M. O., and Wilterdink, R. (2016). Ar-enhanced human-robot-interaction - methodologies, algorithms, tools. *Procedia CIRP*, 44:193 – 198.

[Atmosudiro et al., 2014] Atmosudiro, A., Keinert, M., Karim, A., Lechler, A., Verl, A., and Csizar, A. (2014). Productivity increase through joint space path planning for robot machining. In *2014 European Modelling Symposium*, pages 257–262.

[Beeson and Ames, 2015] Beeson, P. and Ames, B. (2015). TRAC-IK: An open-source library for improved solving of generic inverse kinematics. In *Proceedings of the IEEE RAS Humanoids Conference*, Seoul, Korea.

[Carlson et al., 2013] Carlson, J. S., Spensieri, D., Söderberg, R., Bohlin, R., and Lindkvist, L. (2013). Non-nominal path planning for robust robotic assembly. *Journal of Manufacturing Systems*, 32(3):429 – 435. Assembly Technologies and Systems.

[Caro et al., 2014] Caro, S., Garnier, S., Furet, B., Klimchik, A., and Pashkevich, A. (2014). Workpiece placement optimization for machining operations with industrial robots. In *2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 1716–1721.

[Chen et al., 2008] Chen, H., Fuhlbrigge, T., and Li, X. (2008). Automated industrial robot path planning for spray painting process: A review. In *2008 IEEE International Conference on Automation Science and Engineering*, pages 522–527.

[Codd-Downey and Jenkin, 2015] Codd-Downey, R. and Jenkin, M. (2015). Rcon: Dynamic mobile interfaces for command and control of ros-enabled robots. In *2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, volume 02, pages 66–73.

[Crick et al., 2017] Crick, C., Jay, G., Osentoski, S., Pitzer, B., and Jenkins, O. C. (2017). *Rosbridge: ROS for Non-ROS Users*, pages 493–504. Springer International Publishing, Cham.

[Fernández et al., 2015] Fernández, E., Sánchez, L., Mahtani, A., and Martinez, A. (2015). *Learning ROS for Robotics Programming - second edition.* Packt Publishing.

[Ferrati et al., 2016] Ferrati, M., Nardi, S., Settimi, A., Marino, H., and Pallottino, L. (2016). Multi-object handling for robotic manufacturing. In *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 6887–6893.

[Flacco and Luca, 2014] Flacco, F. and Luca, A. D. (2014). Discrete-time velocity control of redundant robots with acceleration/torque optimization properties. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5139–5144.

[Foote, 2013] Foote, T. (2013). tf: The transform library. In *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, Open-Source Software workshop, pages 1–6.

[Ge et al., 2016] Ge, J., Sun, F., and Liu, C. (2016). Rrt-gd: An efficient rapidly-exploring random tree approach with goal directionality for redundant manipulator path planning. In *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1983–1988.

[Guernane and Achour, 2011] Guernane, R. and Achour, N. (2011). Generating optimized paths for motion planning. *Robotics and Autonomous Systems*, 59(10):789 – 800.

[Halbauer et al., 2013] Halbauer, M., Lehmann, C., Städter, J. P., Berger, U., and Leali, F. (2013). Milling strategies optimized for industrial robots to machine hard materials. In *2013 IEEE 18th Conference on Emerging Technologies Factory Automation (ETFA)*, pages 1–4.

[Hanwell et al., 2015] Hanwell, M. D., Martin, K. M., Chaudhary, A., and Avila, L. S. (2015). The visualization toolkit (vtk): Rewriting the rendering code for modern graphics cards. *SoftwareX*, 1–2:9 – 12.

[Kavraki et al., 1996] Kavraki, L. E., Svestka, P., Latombe, J. C., and Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580.

[Kavraki Lab, 2014] Kavraki Lab, R. U. (2014). Open motion planning library: A primer.

[Klimchik et al., 2017] Klimchik, A., Ambiehl, A., Garnier, S., Furet, B., and Pashkevich, A. (2017). Efficiency evaluation of robots in machining applications using industrial performance measure. *Robotics and Computer-Integrated Manufacturing*, 48:12 – 29.

[Klimchik et al., 2016] Klimchik, A., Magid, E., and Pashkevich, A. (2016). Machining with serial and quasi-serial industrial robots: Comparison analysis and architecture limitations. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 252–259.

[Kohrt et al., 2013] Kohrt, C., Stamp, R., Pipe, A., Kiely, J., and Schiedermeier, G. (2013). An online robot trajectory planning and programming support system for industrial use. *Robotics and Computer-Integrated Manufacturing*, 29(1):71 – 79.

[Kubela et al., 2015] Kubela, T., Pochyly, A., and Singule, V. (2015). Investigation of position accuracy of industrial robots and online methods for accuracy improvement in machining processes. In *2015 International Conference on Electrical Drives and Power Electronics (EDPE)*, pages 385–388.

[Kuffner and LaValle, 2000] Kuffner, J. J. and LaValle, S. M. (2000). Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 2, pages 995–1001 vol.2.

[Larsen et al., 2015] Larsen, L., Pham, V. L., Kim, J., and Kupke, M. (2015). Collision-free path planning of industrial cooperating robots for aircraft fuselage production. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2042–2047.

[LaValle and Kuffner, 1999] LaValle, S. M. and Kuffner, J. J. (1999). Randomized kinodynamic planning. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, volume 1, pages 473–479 vol.1.

[Lentin, 2015] Lentin, J. (2015). *Mastering ROS for Robotics Programming*. Packt Publishing.

[Lin et al., 2017] Lin, Y., Zhao, H., and Ding, H. (2017). Posture optimization methodology of 6r industrial robots for machining using performance evaluation indexes. *Robotics and Computer-Integrated Manufacturing*, 48:59 – 72.

[Liu et al., 2016] Liu, J., Huang, X., Fang, S., Chen, H., and Xi, N. (2016). Industrial robot path planning for polishing applications. In *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1764–1769.

[Ma et al., 2016] Ma, C., Zhang, Y., Zhao, Q., and Bai, K. (2016). 6r serial manipulator space path planning based on rrt. In *2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, volume 02, pages 99–102.

[Makris et al., 2016] Makris, S., Karagiannis, P., Koukas, S., and Matthaiakis, A.-S. (2016). Augmented reality system for operator support in human–robot collaborative assembly. {*CIRP*} *Annals - Manufacturing Technology*, 65(1):61 – 64.

[Ming and Yong, 2015] Ming, Z. and Yong, D. (2015). The research of the serial manipulator autonomous path-planning. In *2015 Fifth International Conference on Instrumentation and Measurement, Computer, Communication and Control (IMCCC)*, pages 1761–1765.

[Moll et al., 2015] Moll, M., Sucan, I. A., and Kavraki, L. E. (2015). Benchmarking motion planning algorithms: An extensible infrastructure for analysis and visualization. *IEEE Robotics Automation Magazine*, 22(3):96–102.

[Motoman, 2009] Motoman, Y. (2009). Motoman mh6 instructions.

[Motoman, 2017] Motoman, Y. (2017). Motoplus-ros incremental motion interface, engineering design specifications.

[Munaro et al., 2016] Munaro, M., Lewis, C., Chambers, D., Hvass, P., and Menegatti, E. (2016). *RGB-D Human Detection and Tracking for Industrial Environments*, pages 1655–1668. Springer International Publishing, Cham.

[Olabi et al., 2010] Olabi, A., Bearee, R., Nyiri, E., and Gibaru1, O. (2010). Enhanced trajectory planning for machining with industrial six-axis robots. In *2010 IEEE International Conference on Industrial Technology*, pages 500–506.

[O'Leary et al., 2017] O'Leary, P., Jhaveri, S., Chaudhary, A., Sherman, W., Martin, K., Lonie, D., Whiting, E., Money, J., and McKenzie, S. (2017). Enhancements to vtk enabling scientific visualization in immersive environments. In *2017 IEEE Virtual Reality (VR)*, pages 186–194.

[Rodríguez et al., 2014] Rodríguez, C., Montaño, A., and Suárez, R. (2014). Planning manipulation movements of a dual-arm system considering obstacle removing. *Robotics and Autonomous Systems*, 62(12):1816 – 1826.

[Rubio et al., 2016] Rubio, F., Llopis-Albert, C., Valero, F., and Suñer, J. L. (2016). Industrial robot efficient trajectory generation without collision through the evolution of the optimal trajectory. *Robotics and Autonomous Systems*, 86:106 – 112.

[Speers et al., 2013] Speers, A., Forooshani, P. M., Dicke, M., and Jenkin, M. (2013). Lightweight tablet devices for command and control of ros-enabled robots. In *2013 16th International Conference on Advanced Robotics (ICAR)*, pages 1–6.

[Sucan et al., 2012] Sucan, I. A., Moll, M., and Kavraki, L. E. (2012). The open motion planning library. *IEEE Robotics Automation Magazine*, 19(4):72–82.

[Tan et al., 2016] Tan, J., Chen, J., Wang, Y., Li, L., and Bao, Y. (2016). Design of 3d visualization system based on vtk utilizing marching cubes and ray casting algorithm. In *2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, volume 02, pages 192–197.

[Toris et al., 2015] Toris, R., Kammerl, J., Lu, D. V., Lee, J., Jenkins, O. C., Osentoski, S., Wills, M., and Chernova, S. (2015). Robot web tools: Efficient messaging for cloud

robotics. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4530–4537.

[Toris et al., 2014] Toris, R., Shue, C., and Chernova, S. (2014). Message authentication codes for secure remote non-native client connections to ros enabled robots. In *2014 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*, pages 1–6.

[Valente et al., 2017] Valente, A., Baraldo, S., and Carpanzano, E. (2017). Smooth trajectory generation for industrial robots performing high precision assembly processes. {*CIRP*} *Annals - Manufacturing Technology*, pages–.

[Valle, 2011] Valle, S. M. L. (2011). Motion planning. *IEEE Robotics Automation Magazine*, 18(2):108–118.

[Wagner et al., 2016] Wagner, M., Hess, P., Reitelshoefer, S., and Franke, J. (2016). 3d scanning of workpieces with cooperative industrial robot arms. In *Proceedings of ISR 2016: 47st International Symposium on Robotics*, pages 1–8.

[Wu et al., 2016] Wu, H., Deng, H., Chen, L., Guan, Y., and Zhang, H. (2016). Sequence-modification based collision-free motion planning of multiple robots workcell. In *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1135–1140.

[Zhu et al., 2015] Zhu, B., Song, A., Xu, X., and Li, S. (2015). Research on 3d virtual environment modeling technology for space tele-robot. *Procedia Engineering*, 99:1171 – 1178.

[Şucan and Kavraki, 2012] Şucan, I. A. and Kavraki, L. E. (2012). A sampling-based tree planner for systems with complex dynamics. *IEEE Transactions on Robotics*, 28(1):116–131.