CENTRO DE INGENIERIA Y DESARROLLO INDUSTRIAL

GENERACIÓN AUTOMÁTICA DE TRAYECTORIAS PARA UN ROBOT MANIPULADOR UTILIZANDO PROCESAMIENTO DE IMÁGENES

Tesis

QUE PARA OBTENER EL GRADO ACADÉMICO DE

Maestro en Ciencia y Tecnología en la Especialidad de Mecatrónica

PRESENTA

Ing. Jorge Alberto Soto Cajiga

POSGRADO INTERINSTITUCIONAL CIENCIA Y TECNOLOGIA

Santiago de Querétaro, Qro., México, Agosto del 2006.

RESUMEN

En este trabajo se presenta una propuesta para generar automáticamente trayectorias para un robot manipulador, específicamente el robot manipulador Melfa RV-2AJ de Mitsubishi, a partir del procesamiento digital de imágenes y el uso de herramientas matemáticas.

El proyecto consiste en adquirir una imagen de un objeto (o figura) del cual se desea conocer su contorno asociado. El cual representa la trayectoria que debe reproducir el robot manipulador.

Como primer punto se adquiere la imagen y se le aplica el tratamiento correspondiente, para obtener una imagen binaria y finalmente el contorno asociado al objeto de la imagen. El contorno se guarda en dos vectores x e y, correspondientes a la posición en el plano del contorno, de este par de vectores se discriminan algunos puntos, en función a la geometría del contorno del objeto, ya a que el número inicial de puntos de estos vectores en algunos casos los hace inmanejables para el robot manipulador.

A los puntos seleccionados de la discriminación de puntos de los vectores x e y, se les aplica un par de interpolaciones paramétricas, una interpolación circular segmentaría y otra interpolación polinómica segmentaría (splines). De acuerdo a una evaluación y a sus ventajas la interpolación por splines es mejor que la interpolación circular y por consiguiente es la más utilizada, sobre todo para realizar las interpolaciones en robótica. Sin embargo, el robot RV2-AJ sólo cuenta con la interpolación circular, lo que hace que sea necesario utilizarla, para poder validar todo el procedimiento. Así la interpolación por splines sólo puede ser evaluada hasta una etapa de simulación y la interpolación circular hasta la etapa experimental.

De la interpolación por splines (que son polinomios cúbicos y por consiguiente fácilmente diferenciables), se obtienen los vectores tangentes y normales al contorno. Vectores que representan la orientación en el espacio de dicho contorno.

Para los cálculos de la cinemática inversa y la generación de la trayectoria del robot manipulador, es necesario conocer la posición y orientación en el espacio de la trayectoria que se desea seguir. La posición se conoce procesando la imagen, obteniendo el contorno y la orientación determinando los vectores tangentes y normales de la interpolación, la cinemática se resuelve y se genera la trayectoria del robot manipulador.

Los resultados de los cálculos de la cinemática inversa son escalados a las dimensiones reales y se utilizan para simular el seguimiento del contorno por el robot, la simulación se realiza en 3-D, de manera que los resultados de la simulación sean muy aproximados a lo que el robot realice en el mundo real. Una vez que se tenga una buena simulación de la trayectoria y que ésta no presente ningún problema, se generan automáticamente los programas requeridos para programar el robot y basta con compilarlos, cargarlos en el robot manipulador y este efectué la tarea a realizar, a partir de dichos programas.

CONTENIDO

CONTENIDO

CONTENIDO

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE FIGURAS

LISTA DE FIGURAS

CAPÍTULO 1

INTRODUCCIÓN

CAPÍTULO 1
INTRODUCCIÓN

En la actualidad, hablar de producción automatizada en la industria es hablar de problemas complejos de ingeniería, asociados a calidad y productividad. Es por esto que existe un gran esfuerzo en investigación y desarrollo en todos sus campos de estudio, en función de producir más, utilizar lo mejor posible los recursos y mejorar la calidad de los productos.

En los procesos de producción actuales son muy utilizadas las máquinas especiales, dentro las cuales destacan sin duda los robots manipuladores, ya que por su arquitectura flexible son fáciles de programar y adecuar a un gran número de tareas, especialmente repetitivas, tediosas y/o peligrosas. Una de las grandes tendencias de la robótica de manipulación, es utilizar todos los recursos tecnológicos, para lograr que el robot aparente inteligencia y desarrolle tareas cada vez más complicadas.

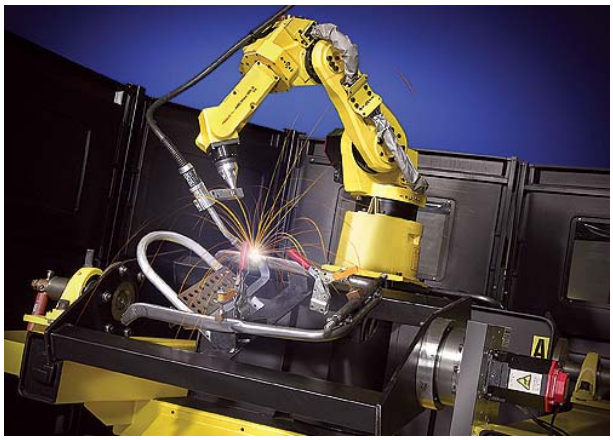


Figura 1.1. Foto de aplicación.

Dentro de estos recursos tecnológicos uno de los más complicados es el sistema de sensado, ya que cada tarea debe ser llevada del medio donde se encuentra el robot a un lenguaje entendible por éste. La tendencia es utilizar visión, ya que gracias a trabajos de investigación y desarrollos que se han venido realizando en este campo, hoy en día el procesamiento digital de imágenes presenta una gran versatilidad y funcionalidad, que traducida a sensado, se puede

considerar como una amplia gama de sensores flexibles encapsulados en una cámara digital. De esta forma de los sistemas de visión se pueden obtener representaciones gráficas (una imagen o secuencia de imágenes) muy parecidas a la realidad, para posteriormente tratarlas y adecuarlas al leguaje que más convenga.

Existen diferentes trabajos referentes a este tema de tesis donde se mezclan diferentes técnicas y se buscan resultados específicos. Uno muy parecido del autor Christophe Collewet, consiste en que a partir de un objeto con forma compleja, aproximar el contorno y finalmente generar el control de un robot manipulador en función del contorno obtenido [15], una desventaja de este método es que se genera el contorno utilizando una descripción polar, la cual no puede representar cualquier forma compleja del contorno, por otro lado, una aportación valiosa de este método es que además de extraer el contorno del objeto se extraen características visuales para poder hacer una representación no planar de éste. Otras técnicas para representar al contorno o reconstruir la superficie de diferentes formas de imágenes planares o no planares es la utilización de sensores de fuerza [16], [17], [18].

1.1 DEFINICIÓN DEL PROYECTO

En el Centro de Ingeniería y Desarrollo Industrial, trabajando en colaboración con el Dr. Emilio Vargas y con el Dr. Carlos Pedraza, quienes trabajan en desarrollos de robótica y procesamiento digital de imágenes respectivamente, surgió la idea de integrar conocimientos y transformarlos en una propuesta de solución para facilitar y agilizar el proceso de programación de robots manipuladores, problema que se presenta en la planeación y programación de trayectorias complicadas de los mismos.

El proceso de seguimiento de algún contorno es sencillo de realizar para un ser humano, sin embargo, el mismo proceso llevado a un robot involucra algunas complicaciones, que no son tan sencillas de resolver. Primero porque se requieren conocimientos multidisciplinarios y segundo porque hay que integrar estos conocimientos. De lograrse exitosamente, se obtiene

como resultado un sistema mucho más eficiente en cuanto a repetibilidad, velocidad y precisión, que lo que podría realizar un ser humano.

El proceso que se plantea es el equivalente a que un ser humano se aprenda el contorno de algún objeto con dimensiones y proporciones precisas, para que posteriormente él mismo reproduzca este contorno las veces que sean necesarias, con un margen de error despreciable y una velocidad alta, situación que es prácticamente imposible. Lo ideal para resolver este problema es utilizar un sensor que reconozca el objeto (cámara CCD), un sistema de procesamiento y almacenamiento y un robot manipulador. En la figura 1.2 se presenta un diagrama a bloques que ejemplifica el sistema completo.

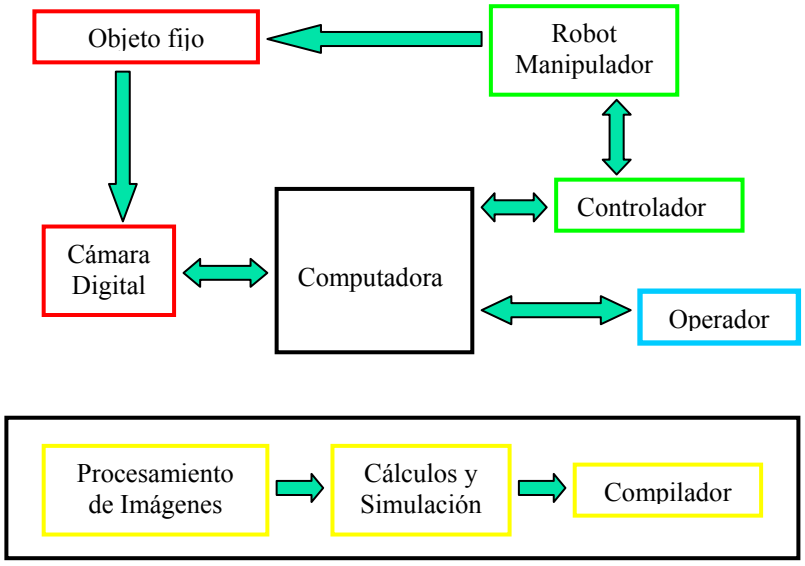


Figura 1.2. Diagrama a bloques del sistema.

1.2 JUSTIFICACIÓN

El tema presentado es importante, no por una aplicación en particular, si no por todas las posibles que se pueden resolver utilizando este procedimiento. Por ejemplo, en el trabajo se desarrolla un algoritmo para la digitalización del contorno de algún objeto, éste puede tener distintas aplicaciones, tales como la digitalización de patrones, entre otras. Otro desarrollo

interesante es la generación de curvas paramétricas del contorno, las cuales son útiles por ejemplo, para suavizar el mismo contorno, para generar trayectorias de máquinas o robots que corten, suelden, etc., ya que a partir de las curvas paramétricas se pueden generar las posiciones, velocidades y aceleraciones que controlen los movimientos de la máquina o el robot.

Otra aportación y una de las principales justificaciones de este proyecto, es la reducción de tiempos de programación de robots manipuladores. Esto porque existe una gran problemática en el tiempo que consume la programación de trayectorias complicadas, ya que normalmente hay que enseñarle al robot punto por punto la trayectoria, la cual muchas veces está determinada hasta por centenas de puntos. Una alternativa es desarrollar cuidadosamente todos los cálculos asociados a la cinemática inversa de la trayectoria deseada para llevarlos al programa del robot, pero de igual forma, si son una gran cantidad de puntos el estar capturando punto por punto, sin cometer errores, es un proceso tedioso y hasta peligroso.

Es por esto que al tener un proceso automático que genere el código del robot, genere todos los puntos necesarios y que además sea simulado previamente, es de gran ayuda para evitar errores de factor humano y para ahorrar tiempo de programación, tiempo que es un recurso no renovable y fundamental para toda empresa. Por último, otra de las aportaciones de este proyecto es la integración de diferentes disciplinas para la generación de las trayectorias en robots manipuladores.

1.3 OBJETIVOS

El objetivo principal de este trabajo es desarrollar un método que simplifique la programación de trayectorias para un robot manipulador, automatizando la generación de la trayectorias a partir de una imagen digital utilizando técnicas de procesamiento digital de imágenes, herramientas matemáticas y de modelación y simulación 3-D. Para finalmente mostrar los resultados del proceso planteado y analizar la factibilidad de su uso.

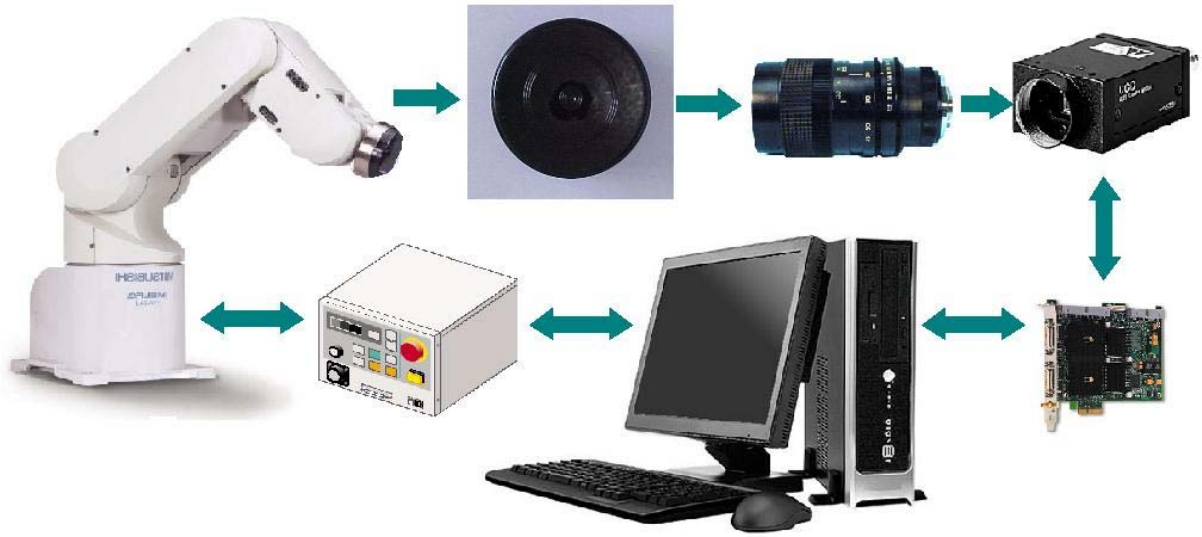


Figura 1.3. Diagrama general del sistema.

Como segundo objetivo se planteó el desarrollo y evaluación de un software de simulación del sistema en 3-D, para el cual se trabaja con una técnicas novedosa, poco utilizada y poco conocida en México, que es la utilización de librerías de OpenGL [1], [2], [19]. Estas librerías se utilizan en el lenguaje de programación C++ y su principal apoyo radica en la optimización para visualización de gráficos, su facilidad para el manejo de transformación de matrices (muy útil en robótica), su portabilidad, sus posibilidades de expansión y su rendimiento.

Por ultimo, el tercer objetivo es el desarrollo y la utilización de diferentes herramientas tanto matemáticas como de software. En las herramientas matemáticas se hace uso de métodos numéricos para calcular interpolaciones y calcular errores, se manejan matrices para el manejo de la cinemática, entre otras. En cuanto al software se usaron algunos algoritmos ya implementados, otros se implementaron, y algo interesante fue el manejo de una interfaz entre Matlab y C++, figura 1.4.

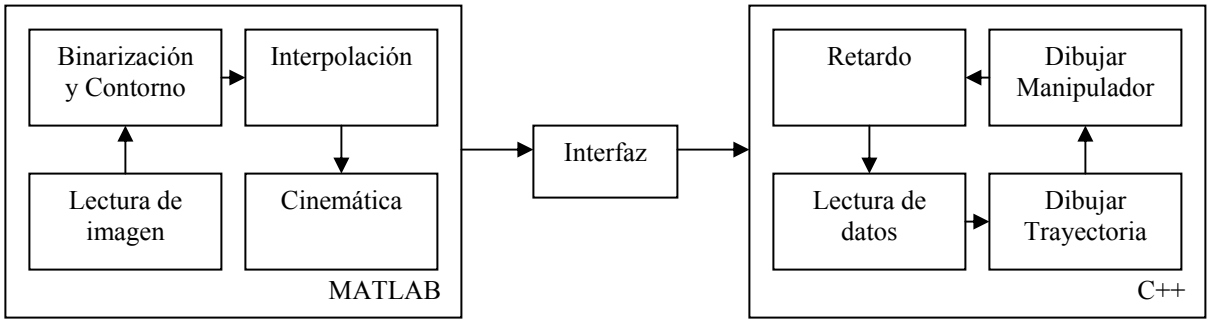


Figura 1.4. Interfaz entre Matlab y C++.

1.4 ALCANCES Y LIMITACIONES

El alcance de este proyecto se enfoca a realizar los objetivos planteados, y aunque hubiera sido sencillo dejar el sistema en pruebas y resultados de modelación y simulación, se llevó el desarrollo hasta la implementación con un robot manipulador real. El proyecto como cualquier otro cuenta con limitaciones, donde su principal fuente es debido al manejo de tres disciplinas con un amplio campo de estudio, como lo son, el procesamiento digital de imágenes, la modelación y simulación, y la robótica, lo cual hace que se limite el proyecto, trabajando sólo en sectores específicos de estos campos.

Los alcances y las limitaciones del proyecto se restringen de acuerdo al manejo del proyecto, por lo que es bueno representar los puntos que se consideran importantes así como las limitaciones que se presentan en el trabajo.

Alcances:

- Desarrollo del algoritmo de detección y parametrización del contorno de la imagen, que aunque existe uno parecido, conocido como el código de la cadena [6], el que se desarrolla presenta ventajas básicas, pero considerables.

- Implementación del algoritmo de interpolación paramétrica utilizando trazadores cúbicos (splines), que debido a sus características matemáticas, son clave para resolver la cinemática inversa y calcular los errores de ajuste de curvas.
- Implementación del algoritmo de interpolación paramétrica utilizando interpoladores circulares, que es el tipo de interpolación que realiza el robot.
- Desarrollo de los cálculos del error de adquisición de la imagen, la representación gráfica y numérica del error de interpolación, en función de los puntos seleccionados y finalmente los cálculos del error de la trayectoria real del robot comparada con el contorno real del objeto.
- Desarrollo de un programa de simulación en 3-D del robot manipulador, que ilustra el seguimiento de la trayectoria que se genera a partir de la imagen y representa lo que el robot hace en la realidad.
- La generación automática de dos programas del robot manipulador a partir de la imagen, uno que representa la lista de comandos a ejecutarse y uno que representa la lista de puntos asociados a los comandos, que finalmente son las posiciones por donde pasa la trayectoria.

Limitaciones:

- No se maneja la etapa de adquisición de la imagen muy a fondo, ya que no se tiene una aplicación específica. Por otro lado la adquisición de una buena imagen depende de las características del objeto u objetos, del sistema de iluminación, del fondo de la imagen, de la resolución de la cámara, del tipo de lente, entre otros factores, que son propios de cada aplicación.
- Las dimensiones de los contornos de los objetos manejados deben estar dentro del espacio de trabajo del robot manipulador.
- El contorno de los objetos debe estar compuesto por trayectorias suaves, que finalmente es lo que se genera de la interpolación y lo que se le programa al robot. Esto implica que no se incluyen ni la detección de aristas en el contorno, ni la interpolación lineal. Estas características se pueden considerar en un trabajo futuro.

- Sólo se desarrolla el procedimiento completo, para un robot tipo articular de 5 grados RV-2AJ de Mitsubishi. Si se desea desarrollar el procedimiento para otro manipulador o máquina hay que considerar su cinemática asociada. En otras palabras, la metodología aquí desarrollada puede escalarse para desarrollar el procedimiento para otros manipuladores.
- Finalmente cabe señalar que no se tiene una aplicación específica, ya que el desarrollo del proyecto es una propuesta de una metodología. Involucrando diferentes disciplinas tales como; la robótica, el procesamiento de imágenes, los métodos numéricos, la modelación y simulación y la interfase entre diferentes lenguajes de programación. El hecho que no haya una aplicación específica no quiere decir que el desarrollo del proyecto no sea aplicable, por el contrario, puede tener un sin fin de aplicaciones, basta con analizar algunos algoritmos desarrollados y ver los resultados experimentales para buscar las diferentes aplicaciones que se pueden generar del desarrollo de este trabajo.

1.5 ORGANIZACIÓN DE TESIS

La organización del trabajo se presenta en 8 capítulos y 2 anexos. En los primeros 4 capítulos se presenta una breve introducción a los fundamentos teóricos de las herramientas y técnicas utilizadas. Del capitulo 5 al 6 se presenta el desarrollo paso a paso del procedimiento que se propuso y se siguió para implementar el método para la generación automática de trayectorias, de acuerdo a los objetivos. En el capítulo 7 se presentan las pruebas y resultados obtenidos del procedimiento. En el capítulo 8 se presentan los resultados y conclusiones finales. En el anexo 1 se presenta una breve explicación de OpenGL y en el anexo 2 se presenta el procedimiento utilizado para la calibración del robot manipulador. Además se incluye un CD como anexo 3, donde se presentan de forma muy general los diferentes diagramas de flujo de los programas desarrollados en este trabajo, se incluyen los programas ejecutables de Matlab y C++, y algunos archivos donde se explica como instalar, como correr y que hacen estos programas.

En el capítulo 1 se presentan las generalidades y direcciones de la tesis, en donde se presenta una introducción para tener una idea general del proyecto y conocer las consideraciones que se tomaron para llevarlo a cabo.

En el capítulo 2 se presentan los fundamentos para el procesamiento digital de imágenes, dando una breve introducción. Se presentan los fundamentos de las técnicas para la binarización por la detección del umbral óptimo y para la detección de contorno, técnicas que se utilizan para el desarrollo del proyecto.

En el capítulo 3 se presentan los fundamentos de la interpolación por trazadores cúbicos (splines), la interpolación circular y finalmente el manejo de curvas paramétricas.

En el capítulo 4 se presentan algunas conceptos de la robótica, desde un punto de vista muy general, principalmente para entender en que consiste la cinemática directa, la cinemática inversa y la generación de trayectorias.

En el capitulo 5 se muestran los primeros pasos del desarrollo del proyecto utilizando el procesamiento digital de imágenes y las interpolaciones, circular y por splines. Se desarrollan las etapas que consisten en la adquisición y binarización de la imagen, la detección del contorno del objeto (o figura), la generación de las curvas paramétricas, tanto por la interpolación circular como por la interpolación por splines y la adecuación de las dimensiones de la imagen al mundo real (píxeles a mm). Dando como resultado de estos primeros pasos un conjunto de datos, que son necesarios para los siguientes puntos.

En el capítulo 6 se presenta el desarrollo de los cálculos de la cinemática del manipulador incluyendo la generación de la trayectoria, tomando como referencia los datos arrojados de la generación del contorno y su interpolación.

En el capítulo 7 se desarrollan los cálculos para obtener los errores asociados al procedimiento, obteniendo resultados de tres tipos de error, el error de adquisición, el error de interpolación y finalmente el error de ejecución del robot. Este último error es debido a

desajustes mecánicos, descalibración, en conclusión a cualquier problema asociado al robot manipulador y que influye en la ejecución de la trayectoria.

En el capítulo 8 se analizan los resultados obtenidos a lo largo del desarrollo del proyecto, donde se incluyen resultados y conclusiones de la diferentes etapas involucradas así como las perspectivas del trabajo a futuro y los artículos publicados.

En el anexo 1 se presenta una breve introducción a lo que son las librerías de OpenGL, en el anexo 2 se describe el procedimiento de calibración del robot manipulador y el porqué de la necesidad de calibrarlo y finalmente se anexa un CD, como anexo 3, donde se presentan de forma muy general los diagramas de flujo de los algoritmos desarrollados, se presentan los programas ejecutables de Matlab y C++, un archivo en forma digital donde se explica donde ubicar estos programas, qué hacer para que corran y un archivo de ayuda para saber que función realiza cada uno de los programas.

CAPÍTULOS 2-4

FUNDAMENTOS TEÓRICOS

CAPÍTULO 2
PROCESAMIENTO DIGITAL DE IMAGÉNES

En este capítulo se presenta una breve introducción al procesamiento digital de imágenes, se analizan las etapas fundamentales en las que se divide y finalmente, se hace referencia a un par de técnicas: La binarización por detección de umbral y la detección de contorno por el código de la cadena, que son de gran interés para este trabajo.

2.1 VISIÓN ARTIFICIAL

En visión artificial se suelen distinguir tres procesos, *Procesamiento, Análisis y Aplicaciones* [3]. El procesamiento implica la manipulación de las imágenes vistas como señales digitales, para extraer la información más elemental subyacente. El análisis se encamina a determinar ciertas estructuras elementales tales como bordes o regiones así como las relaciones entre ellas. Las aplicaciones tratan de dar solución a los problemas relacionados con ciertas situaciones del mundo real, como reconocimiento, movimiento, reconstrucción 3-D, etc.

El interés de los métodos de procesamiento de imágenes digitales se fundamenta en dos áreas principales de aplicación [3]: *a*) mejora de la calidad para la interpretación humana; *b*) procesamiento de los datos de la escena para la percepción de las máquinas de forma autónoma. En el intento de dotar a las máquinas de un sistema de visión aparece el concepto de Visión Artificial. En la mayoría de los sistemas la capacidad sensorial de la visión es complementada con otros sistemas sensoriales tales como detectores de alcance o proximidad. Posteriormente es necesario realizar una integración multisensorial para completar el proceso de percepción global.

Desde una perspectiva general, la visión artificial por computadora es la capacidad de las máquinas para ver el mundo que las rodea, más precisamente, para deducir la estructura y las propiedades del mundo tridimensional a partir de una o más imágenes bidimensionales.

Las imágenes adquiridas pueden ser segmentadas para obtener de ellas características de interés tales como bordes o regiones. Posteriormente, de las características se obtienen las propiedades subyacentes mediante el correspondiente proceso de descripción, tras lo cual se consigue la estructura de la escena requerida para la aplicación de interés.

Aunque la distinción entre procesamiento y análisis de imágenes digitales no es obvio de forma inmediata, el procesamiento de imágenes puede ser visto como una transformación de una imagen a otra imagen. Por otro lado, el análisis es una transformación de una imagen a algo distinto a una imagen, en consecuencia el análisis es un determinado tipo de información representando una descripción o una decisión. En cualquier caso, las técnicas de análisis de imágenes digitales son aplicadas a imágenes que han sido procesadas previamente. Para los sistemas de visión artificial el único propósito del procesamiento de imágenes es hacer el análisis posterior más simple y fiable [3].

2.2 PASOS FUNDAMENTALES DEL PROCESAMIENTO DE IMÁGENES

En este apartado se presentan los conceptos básicos necesarios para el procesamiento digital de imágenes [4], y en especial una breve descripción de los métodos utilizados para realizar el procesamiento en la detección de contorno de un objeto en una imagen.

Un sistema de procesamiento digital de imágenes puede ser utilizado para desarrollar diferentes tipos de tareas entre las que se encuentran; Inspección de procesos, Visión Artificial, Control de Calidad, entre otras. El presente trabajo utiliza algunas técnicas de procesamiento digital de imágenes para desarrollar un sistema de Visión Artificial que permita reconocer y parametrizar el contorno de un objeto a partir de una imagen adquirida por medio de una cámara digital.

El contenido de esta sección se centra en los métodos utilizados para la manipulación de imágenes para la extracción de información [4]. Estos sistemas comprenden las siguientes etapas:

- Adquisición de Imágenes
- Preprocesamiento
- Segmentación
- Representación y Descripción
- Reconocimiento e Interpretación

En la figura 2.1 se ilustran estas etapas sin incluir la escena real de donde se obtiene la imagen, ni la etapa de despliegue o visualización de la información.

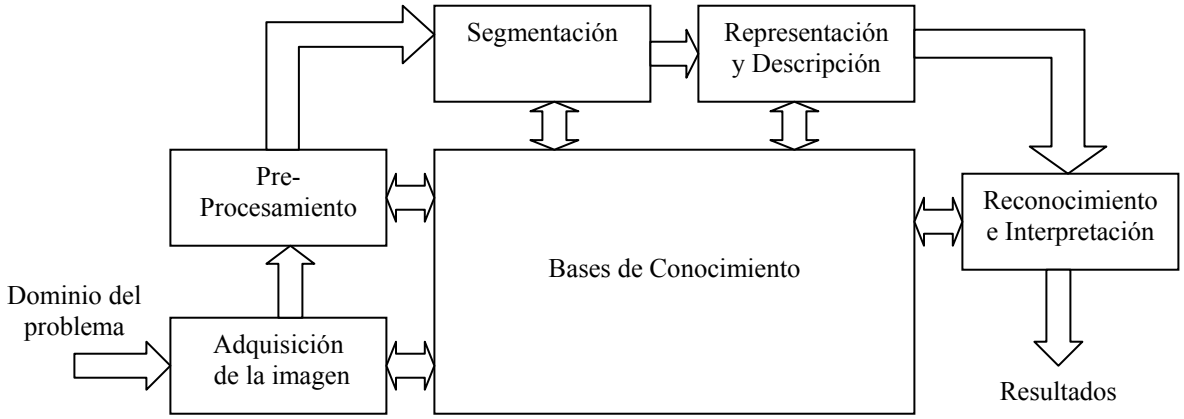


Figura 2.1. Etapas Fundamentales del Procesamiento Digital de Imágenes.

2.2.1 Adquisición de Imágenes

Para realizar la adquisición de imágenes, es necesario contar con un dispositivo físico sensible a una determinada banda del espectro electromagnético. Actualmente existen en el mercado diferentes tipos de dispositivos que pueden ser utilizados para este fin, desde los más básicos como por ejemplo los que responden únicamente a la presencia o no presencia de luz, hasta las cámaras CCD que son utilizadas más frecuentemente en los sistemas de procesamiento de

imágenes, debido a que brindan una mejor resolución que las vídeo cámaras normales. La señal de salida de estos dispositivos puede ser digital o analógica. Para efectos del procesamiento digital, debe usarse un convertidor de analógico a digital para las señales que no se encuentren en formato digital.

2.2.2 Preprocesamiento

El objetivo de la etapa de preprocesamiento de una imagen, es mejorar las características en la imagen de forma que sea más sencillo aplicarle otros procesos. Para obtener estas mejoras se utilizan diferentes técnicas como mejorar el contraste, remover ruidos, aislamiento de regiones, etc., cuya aplicación se ajuste a las necesidades del proceso.

Las cualidades de las imágenes que se van a analizar en este trabajo, no hacen necesaria la utilización de técnicas especiales, ya que la imagen adquirida está compuesta básicamente de el fondo y el objeto con un contraste considerablemente alto entre ambos.

2.2.3 Segmentación

En un sistema de visión artificial, la segmentación es una etapa determinante en el procesamiento digital de imágenes. El objetivo de esta etapa es aislar los objetos de interés, para luego realizar el análisis de sus características. Existen una gran cantidad de algoritmos para realizar la segmentación de objetos en una imagen, pero su utilización depende de la aplicación específica.

De alguna manera un proceso de segmentación adecuado lleva el procesamiento de la imagen por un camino adecuado y es más sencillo darle solución al problema. De otra manera la utilización de algoritmos erráticos en la segmentación casi siempre garantiza eventual falla en la solución del problema.

2.2.4 Representación y Descripción

En el proceso de descripción se extraen características que resulten en información cuantitativa de interés o características básicas, con el objetivo de diferenciar una clase de objetos de otra, reconocer objetos, parametrizar objetos, etc.

2.2.5 Reconocimiento e Interpretación

La ultima etapa es la de reconocimiento e interpretación. Reconocimiento es el proceso en el que se asigna una etiqueta a un objeto, basándose en la información obtenida del proceso de descripción. Interpretación involucra la asignación del significado a objetos reconocidos, de igual forma con base al proceso de descripción.

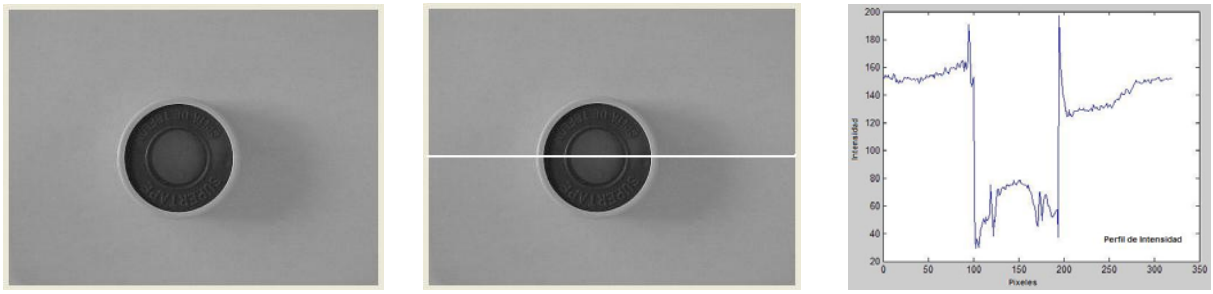
Como nos podemos dar cuenta hasta este momento, las etapas del procesamiento digital de imágenes están bien definidas, sin embargo, su definición fue en base a algunas aplicaciones ya desarrolladas, por lo que podría darse el caso de que en algún proceso no se puedan aplicar, en especial a alguna aplicación especifica. Es por esto, que la base de conocimientos del tema de aplicación que se esté desarrollando es muy importante, ya que los conocimientos son la base para guiar el proyecto a una solución exitosa. En la figura 2.1 se muestra un diagrama general de las etapas fundamentales del procesamiento digital de imágenes, donde se puede observar cómo la base de conocimientos del problema está íntimamente ligado a cada etapa del procesamiento.

En los siguientes apartados se da una breve introducción al procesamiento digital de imágenes y se documentan los algoritmos de procesamiento digital de imágenes que se utilizaran para el desarrollo de este trabajo.

2.3 PROCESAMIENTO BÁSICO DE IMÁGENES

Aunque hay filosofías de que existen diferentes tipos de imágenes [3], en cualquier caso, tras su captura se tendrá una matriz 2-D de valores, es decir, una imagen digital. Independientemente del tipo de sensor utilizado y del tipo de imagen que se maneje, el tipo de imagen se presenta digitalizada en forma de matriz con una resolución de MxN elementos. Cada elemento de la matriz denominado píxel (picture element), tendrá un valor asignado que corresponde con el nivel de luminosidad del punto correspondiente en la escena captada. Dicho valor es el resultado de la cuantización de intensidad o nivel de gris (se trabajará sólo con imágenes en escala de grises). Los términos intensidad y nivel de gris se suelen usar indistintamente.

Por ejemplo la imagen de la figura 2.2(a) presenta una resolución espacial de 320x240 píxeles y con niveles de gris en el rango de 0 a 255 (8 bits) donde el 0 representa el negro absoluto y el 255 el blanco absoluto.



(a) *(b)* *(c)*

Figura 2.2. (a) imagen original de 320x240 píxeles y niveles de gris entre 0 y 255, (b) Línea transversal, cuyo perfil de intensidad se representa en (c).

Los niveles de gris correspondientes a la fila 130 (línea blanca en la figura 2.2(b)) se pueden observar en la figura 2.2(c), ésta representación gráfica de valores se conoce como perfil de intensidad.

Las imágenes, una vez adquiridas, son enviadas a una computadora para su tratamiento mediante programas específicos o desarrollados por el usuario, para finalmente visualizar los resultados a través de monitores de PC o TV, impresoras, alguna aplicación específica, etc.

El procesamiento de imágenes puede enfocarse desde dos perspectivas [4]:

1) Alteración píxel a píxel de los datos en una escala global (individuales).
2) Operaciones basadas en múltiples puntos (vecindad).

A continuación se describen las perspectivas con las operaciones correspondientes.

2.3.1 Operaciones Individuales

Las operaciones individuales implican la generación de una nueva imagen, modificando el valor del píxel en una simple localización basándose en una regla global aplicada a cada localización de la imagen original. El proceso consiste en obtener el valor del píxel de una localización dada en la imagen, modificándolo por una operación lineal o no lineal y colocando el valor del nuevo píxel en la correspondiente localización de la nueva imagen. El proceso se repite por todas y cada una de las localizaciones de los píxeles en la imagen original, figura 2.3.

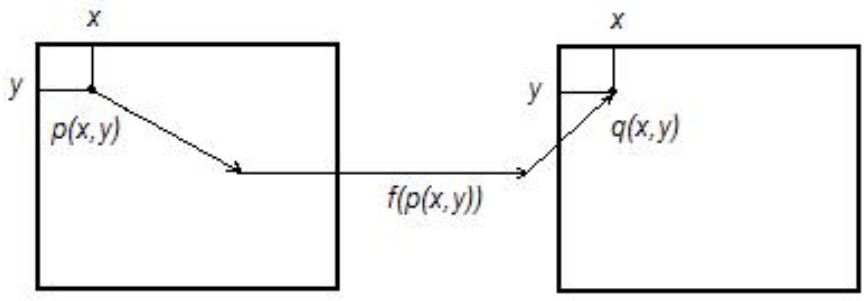


Figura 2.3. Operación Individual.

Como se aprecia en la figura 2.3, el operador individual es transformado uno a uno. El operador *f* se aplica a cada píxel en la imagen o sección de la imagen y la salida depende

únicamente de la magnitud del correspondiente píxel de entrada; la salida es independiente de los píxeles adyacentes. La función transforma el valor del nivel de gris de cada píxel en la imagen y el nuevo valor se obtiene a través de la ecuación,

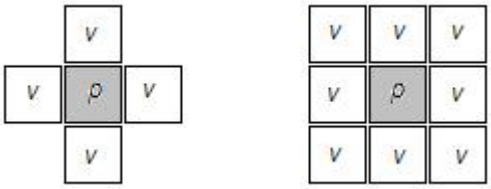
$$q(x, y) = f(p(x, y)) \tag{2.1}$$

la función *f* puede ser un operador lineal o no lineal. El proceso matemático es relativamente simple. La imagen resultante es de la misma dimensión que la original. A continuación se listan los operadores más utilizados para las operaciones individuales [3]:

- Operador identidad
- Operador inverso o negativo
- Operador umbral
- Operador intervalo de umbral binario
- Operador intervalo de umbral binario invertido
- Operador de umbral de la escala de grises
- Operador de escala de grises invertido
- Operador de extensión
- Operador reducción de nivel de gris
- Transformación de imágenes punto a punto

2.3.2 Operaciones de Vecindad

Las operaciones de vecindad utilizan el mismo procedimiento excepto que el nuevo valor del píxel en la imagen de salida, depende de una combinación de los valores de los píxeles en la vecindad del píxel de la imagen original que está siendo transformada. Algunas de estas transformaciones son operaciones de convolución. En la figura 2.4 se ilustran las vecindades más utilizadas.



(a) *(b)*

Figura 2.4. (a) vecindad 4 $E_4(p)$, (b) vecindad 8 $E_8(p)$.

Básicamente consiste en transformar el valor de un píxel *p* en la posición *(x,y)* teniendo en cuenta los valores de los píxeles vecinos. Por ejemplo si consideramos una vecindad $E_8(p)$, realizamos una suma ponderada con los valores de lo 8 vecinos y el resultado de dicha suma es el valor del nuevo píxel *q* de la imagen de salida en la misma posición *(x,y)*. Lo único que resta es definir los valores de ponderación, lo cual se hace generalmente definiendo una máscara con valores constantes. Dicha máscara es realmente un filtro, por lo que dependiendo de la naturaleza del mismo, será el resultado de la operación. Por ejemplo si definimos la siguiente máscara,

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2.2}$$

y aplicamos esta transformación sobre todos los píxeles de la imagen original, obtendremos una nueva imagen, en la figura 2.5 se puede observar el resultado de aplicar este filtro a una imagen.

(a) *(b)*

Figura 2.5. (a) Imagen original (b) Imagen filtrada con la máscara 2.1.

En la figura 2.5(a) se muestra la imagen original y en la figura 2.5(b) el resultado de la operación de vecindad con la máscara 2.1 sobre la imagen original. Como se puede apreciar el resultado es una especie de repujado en relieve donde se marcan los bordes respecto del resto de la imagen. Esta máscara es uno de los operadores de Sobel, [3].

Como se puede intuir con esta introducción al procesamiento digital de imágenes, existe un gran numero de transformaciones y operaciones que se aplican en este campo, desde operaciones muy básicas, como lógicas o aritméticas hasta transformaciones en frecuencia que implican conocimientos matemáticos más avanzados.

El procesamiento digital de imágenes cuenta con material muy vasto, como se ha visto hasta ahora. Con esta breve introducción se pretendió dar una idea básica al procesamiento digital de imágenes. En las siguientes secciones se presentan los fundamentos de dos técnicas utilizadas para el desarrollo de la tesis.

2.4 BINARIZACIÓN MEDIANTE DETECCIÓN DE UMBRAL

El preprocesamiento utilizado o primer paso para la obtención del contorno de un objeto es la binarización. En esta sección se describe una técnica de binarización, que fue la que se utilizó para el desarrollo del proyecto.

En las imágenes aparecen ciertas áreas o zonas caracterizadas por el hecho de que constituyen agrupaciones de píxeles conectados entre sí, pero además de la conexión, dichos píxeles presentan propiedades o características comunes como las regiones. En esta sección se obtendrá una imagen binaria basada en el hecho de que los píxeles de una determinada región presentan una distribución de intensidad similar, por tanto a partir del histograma de niveles de gris se determina cuál es la zona del mismo que corresponde a una determinada región.

El uso de los umbrales en el tratamiento de imágenes constituye una de las principales técnicas en los sistemas de visión industrial para la detección de objetos, especialmente en las aplicaciones que requieran procesar una cantidad elevada de datos [3].

Supongamos que el histograma de la figura 2.6 corresponde a una imagen *f(x,y)* compuesta por objetos claros sobre un fondo oscuro, teniendo los píxeles del objeto y del entorno intensidades agrupadas en dos tonos dominantes. Una forma obvia de extraer los objetos del entorno es seleccionar un nivel *T* que separe los dos tonos de intensidad. De esta forma, un píxel *(x,y)* para el cual *f(x,y)>T* será un píxel del objeto, en caso contrario será un píxel del entorno.

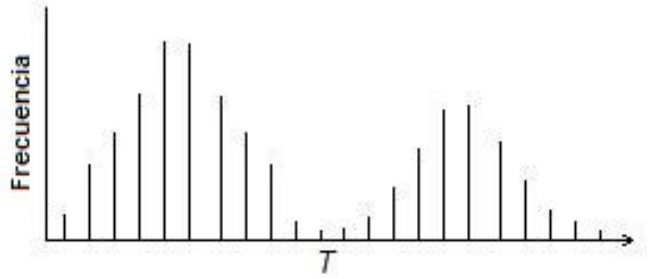


Figura 2.6. Histograma de intensidad, umbral único.

Con base en la figura 2.6, se puede considerar la fijación del umbral como una operación que implica pruebas con respecto a una función *T* de la forma,

$$T = T[x, y, p(x, y), f(x, y)] \tag{2.3}$$

donde *f(x,y)* es la intensidad en el punto *(x,y)* y *p(x,y)* es alguna propiedad local del punto; por ejemplo, la intensidad media de un entorno de vecindad centrado en *(x,y)*. Se creará una imagen binaria *g(x,y)* definiendo,

$$g(x, y) = \begin{cases} 0, & f(x, y) > T \\ 1, & f(x, y) \le T \end{cases} \tag{2.4}$$

Examinando *g(x,y)* se ve que los píxeles a los que se les asigna el valor 0 corresponden a los objetos, mientras que los correspondientes al entorno tienen valor 1. Puede ser a la inversa.

Cuando *T* depende sólo de *f(x,y)*, el umbral se llama global. Si *T* depende tanto de *f(x,y)* como de *p(x,y)*, entonces el umbral se llama local. Si *T* depende de las coordenadas espaciales *x* e *y*, se llama umbral dinámico.

2.4.1 Selección del umbral óptimo

Normalmente no es tan fácil obtener el nivel de umbral apropiado para la binarización. Una forma de seleccionar el umbral óptimo es aproximar el histograma por la suma de funciones de densidad de probabilidad, figura 2.7, donde se sabe que el error medio de clasificar equivocadamente un píxel de objeto como fondo, o viceversa, se minimiza si el umbral se fija en un valor en el cual las funciones de densidad de probabilidad se cruzan figura 2.7.

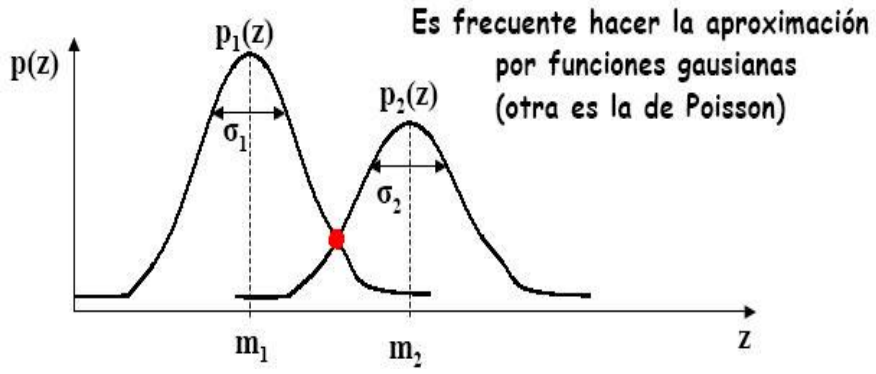
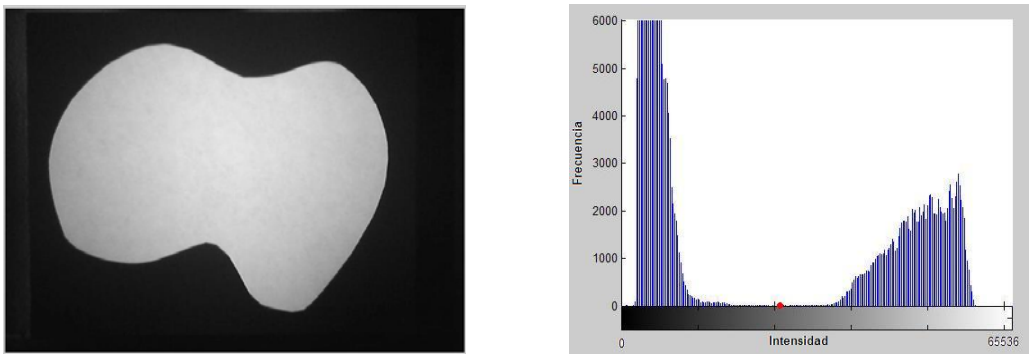


Figura 2.7. Funciones de densidad de probabilidad.

En este caso la selección del umbral parece sencillo, sin embargo, las funciones de probabilidad para esta figura son ideales y normalmente no es de esta forma, en la figura 2.8 se muestra la imagen real y su histograma correspondiente. Lo interesante es deducir cual es el umbral óptimo para esta figura.



(a) *(b)*

Figura 2.8. (a) Imagen de objeto, (b) histograma de la imagen.

Existen diferentes métodos para obtener el umbral óptimo. La selección del método adecuado depende en gran medida de la imagen que se desea binarizar, a continuación se mencionan algunos de estos métodos:

- El método de Otsu
- El método del gradiente
- El método de relajación
- El método de la distribución de curvatura
- El método de la entropía
- El método isodata
- El método de triangulación
- Métodos locales adaptivos HALT

De los métodos mencionados se selecciono el de Otsu [5]. Cabe señalar que no se probaron todos los métodos, sólo se realizo una búsqueda bibliográfica, se observaron los resultados y el tipo de problemas para los que se utilizan, para poder deducir que el método de Otsu es suficiente, por su eficiencia, para la aplicación que se tiene, ya que esta basado en seleccionar el umbral óptimo que separe dos clases Gaussianas, en este caso al fondo del objeto.

2.4.2 El método de Otsu

El método de Otsu [5] suele estar referenciado en un gran numero de trabajos que involucren el preprocesamiento o la segmentación.

Otsu propuso un método de selección de umbral óptimo automático de histogramas en escala de grises usando un criterio de discriminación. El método es no supervisado y no paramétrico, propiedades que podrían ser deseables en algunas aplicaciones de procesamiento de imágenes, no en nuestro caso.

En un caso ideal el histograma de una imagen tiene un profundo valle entre dos picos representando objetos y fondo. Sin embargo, este no es el caso de la mayoría de las escenas reales. Normalmente se tienen dos picos producidos por una distribución no ideal de escala de grises, además de que en ocasiones las distribuciones tanto del fondo como del objeto se traslapan y esto hace difícil la obtención del umbral óptimo. Algunas técnicas tratan de resolver el problema aproximando el histograma en la obtención del mínimo cuadrado por una suma de distribuciones Gaussianas y entonces aplicar procedimientos de decisión estáticos. La implementación de estas técnicas requiere demasiados y, algunas veces, cálculos inestables. La conclusión de la aplicación de estos métodos de selección de umbral es que las distribuciones Gaussianas son buenas para aproximar los modos a un histograma.

Siguiendo esta dirección, la aproximación de Otsu está basada en un análisis discriminante. La operación de umbral es considerada como la partición de los píxeles de una imagen en dos clases Gaussianas C_0 y C_1 (ej. Objetos y fondo) discriminadas a una escala de grises t. Esto es $C_0 = \{0,1...t\}$ y $C_1 = \{t+1,t+2...l-1\}$, donde l es el numero total de los distintos niveles de gris. Definiendo σ_W^2, σ_B^2, σ_T^2 como la varianza dentro de clases, la varianza entre clases y la varianza total respectivamente, las cuales alcanzan el máximo a un umbral equivalente t. Así un umbral t^* óptimo puede determinarse por la maximización de una de las siguientes funciones criterio,

$$\lambda = \frac{\sigma_B^2}{\sigma_W^2}, \qquad \eta = \frac{\sigma_B^2}{\sigma_T^2}, \qquad \kappa = \frac{\sigma_T^2}{\sigma_W^2}.$$

Tomando $\boldsymbol{\eta}$ por ejemplo, el umbral óptimo $\boldsymbol{t}^*$ está determinado como,

$$t^* = ArgMax_{0<t<l}\{\eta\}$$

donde,

$$P_i = \frac{n_i}{n}, \qquad \omega_0 = \sum_{i=0}^{t} P_i, \qquad \omega_1 = 1 - \omega_0$$

$$\mu_{\mathrm{T}} = \sum_{l=0}^{l-1} iP_i, \qquad \mu_t = \sum_{i=0}^{t} iP_i, \qquad \mu_0 = \frac{\mu_t}{\omega_0},$$

$$\mu_1 = \frac{\mu_{\mathrm{T}} - \mu_t}{1 - \mu_0}, \qquad \sigma_{\mathrm{T}}^2 = \sum_{i=0}^{l-1} (i - \mu_T)^2 Pi, \qquad \sigma_B^2 = \omega_0\omega_1(\mu_1 - \mu_0)^2.$$

Aquí, n_i es el numero de píxeles en el *i*-ésimo nivel de gris, $n = \sum_{i=0}^{l-1} n_i$ es el numero total de píxeles en la imagen y $P_i = n_i / n$ es la probabilidad de ocurrencia a la escala de gris *i*. Para un umbral seleccionado t^* de una imagen dada, las probabilidades de clases ω_0 y ω_1 representan las porciones de áreas ocupadas por las clases del objeto y las del fondo respectivamente. El máximo valor de η, denotado por η^*, puede servir como la medida de la separabilidad entre las dos clases y la bimodalidad del histograma. La separabilidad de clase η cae en el rango *[0, 1]*; la frontera inferior es obtenida cuando la imagen tiene un nivel de gris uniforme, y la frontera superior es obtenida cuando la imagen consiste de sólo dos niveles de gris. Conforme η es más grande, el fondo es más sencillo. El propósito de esta aplicación es clasificar la imagen original de acuerdo a la separabilidad de sus clases, para posteriormente tratarla con diferentes técnicas. Para un caso ideal cuando los objetos y el fondo están compuestos de sólo dos niveles de gris distintos, la separabilidad entre clases alcanza el máximo, $\eta^*=1$.

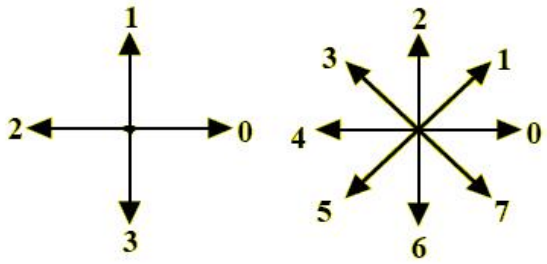
El algoritmo aquí presentado no se desarrolló en este trabajo, ya que está implementado en Matlab como función, teniendo como entrada la imagen y dando como salida el nivel óptimo de umbral para la binarización entre dos clases.

2.5 DETERMINACIÓN DE CONTORNOS

Existen diferentes técnicas para la detección y el seguimiento de contornos, algunas están más enfocadas a la visualización, con objetivos de marcar para poder extraer regiones de interés; otras, como la que nos interesa, se enfocan más a la parametrización.

2.5.1 El código de la cadena

Los códigos de cadena son utilizados para representar una frontera o contorno, a partir de una secuencia conectada de segmentos de líneas rectas de longitud y dirección especificas [6]. Típicamente, esta representación se basa en la conectividad de los segmentos 4 u 8. La dirección de cada segmento es codificada usando un número como se muestra en la figura 2.9.



(a) *(b)*

Figura 2.9. (a) Conectividad 4, (b) Conectividad 8.

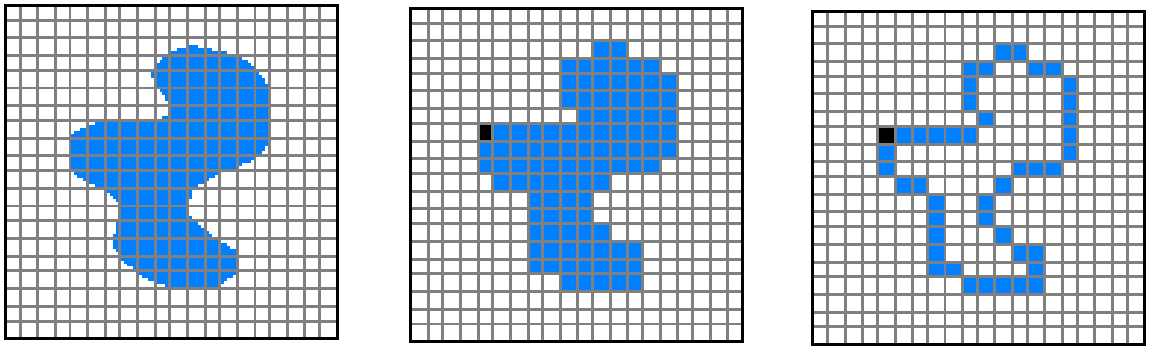
Las imágenes digitales son adquiridas y procesadas usualmente en un formato de rejilla (grid) con el mismo espacio en las direcciones *x* e *y*. De esta forma un código de cadena puede ser generado siguiendo un contorno, ya sea en dirección horario o antihorario. Este método es poco aceptado por dos principales razones: Primero, los códigos de cadena resultantes son generalmente muy grandes y segundo, cualquier pequeño disturbio a lo largo del contorno se

puede convertir en ruido que puede generar cambios en el código, que no necesariamente están relacionados al contorno del objeto o que sencillamente corten la cadena.

Una técnica para resolver estos problemas es utilizar una rejilla con espacios más grandes y realizar un algoritmo que relacione las intensidades de los píxeles dentro de cada rejilla y así determinar el contorno. Para este proyecto no importa si los códigos de la cadena resultantes son muy grandes, ya que se almacenará la posición de la trayectoria del contorno en un par de vectores (*x,y*), y aunque sea muy grande es considerablemente menor que toda la imagen, la cual puede ser desechada después de obtener el contorno y sólo trabajar con el contorno que es lo único que interesa. Para el problema de que existan disturbios a lo largo del contorno hay que tener más cuidado, ya que si el ruido es considerable el contorno se puede desproporcionar o la cadena se puede cortar sin lograr obtener el contorno deseado. Esto se resuelve fácilmente al tener un buen contraste entre el objeto y el fondo, y aplicándole una modificación al algoritmo.

Por otro lado, el código de cadena tiene algunas ventajas considerables para la aplicación que se tiene, y es que además de ser rápido, se pueden ir guardando las posiciones del contorno en un par de vectores (*x,y*) de forma precisa, a diferencia de otras técnicas que son aproximaciones, más ilustrativas y trabajan sobre toda la imagen. Esta fue la razón principal por la que se utilizó esta técnica.

Un ejemplo del código de cadena se muestra en la figura 2.10, en 2.10(a) se representa la rejilla sobre el objeto real, en 2.10(b), se representa la digitalización del objeto y finalmente en 2.10(c), se representa el contorno digitalizado del objeto.



(a) *(b)* *(c)*

Figura 2.10. (a) Objeto, (b) Digitalización, (c) Contorno.

El código de cadena para la figura 2.10(c), utilizando una conectividad 8 (mostrada en la figura 2.9(b)) y tomando como inicio el punto negro de la figura 2.10(b), se muestra a continuación:

CC = 0,0,0,0,0,1,3,2,2,2,0,1,0,7,0,7,6,6,6,6,5,4,4,5,5,6,7,7,0,6,6,4,4,4,4,3,4,2,2,2,2,3,4,3,2,2.

Cabe mencionar que una de las contribuciones de este trabajo de tesis es el desarrollo de un algoritmo parecido al código de la cadena, el cual se desarrolló en Matlab. Este algoritmo se adecuó a la aplicación en puntos de interés, como lo son; ubicar los puntos del contorno sobre el plano de trabajo, así como la consideración del criterio de seguimiento del contorno y el criterio de paro, buscando que la salida sean siempre un par de vectores (x,y) para el posterior procesamiento matemático. En el capítulo 5 se explica este algoritmo más a fondo.

CAPÍTULO 3

INTERPOLACIÓN Y CURVAS PARAMÉTRICAS

En este capítulo se hace referencia a dos técnicas de interpolación muy utilizadas para realizar las trayectorias de diferentes robots manipuladores. Estas técnicas son la interpolación circular y la interpolación por trazadores cúbicos, también conocida como splines. Otro tema que se toca en este capítulo es el manejo de curvas paramétricas, las cuales son usadas entre otras cosas para representar trayectorias, formas, figuras, etc., en forma de funciones de una sola variable.

3.1 INTERPOLACIÓN CIRCULAR

La interpolación circular consiste en construir la ecuación de un círculo (fórmula 3.1), a partir de tres puntos dados de éste [20], en la figura 3.1 se representan los tres puntos y el círculo que corresponde a éstos.

$$r^2 = x^2 + y^2 \tag{3.1}$$

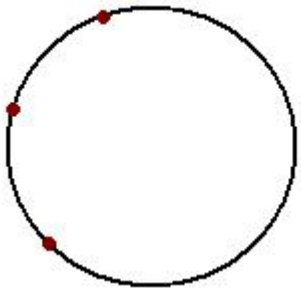
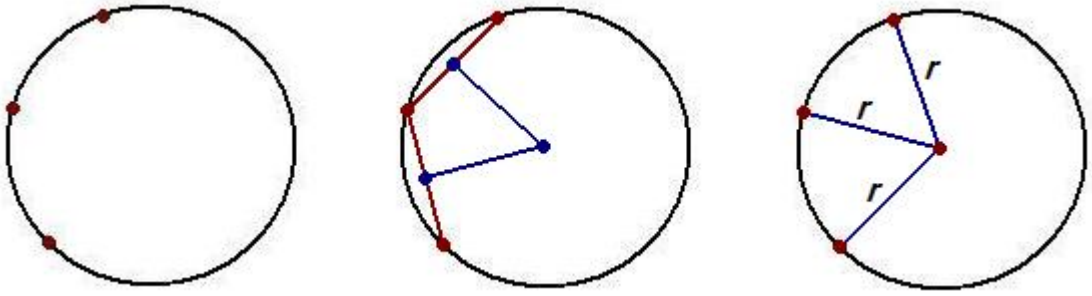


Figura 3.1. Puntos sobre círculo.

Como es conocido, para representar un círculo es suficiente conocer su radio, por lo que lo interesante de esta técnica es la obtención de este parámetro. Para determinar el radio del círculo se utiliza el siguiente procedimiento:

- Se trazan dos líneas rectas entre los puntos dados (figura 3.2 (b)).
- Se determina el centro de cada una de las dos líneas (figura 3.2 (b)).
- Del centro de cada una de las líneas se traza un vector ortogonal hacia el centro del círculo (figura 3.2 (b)).
- La intersección de los vectores determina el centro del círculo (figura 3.2 (b)).
- Finalmente la distancia del centro del círculo a cualquiera de los puntos iniciales determina su radio (figura 3.2 (c)).



(a) *(b)* *(c)*

Figura 3.2. Determinación del radio del círculo a partir de tres puntos.

Posteriormente en el desarrollo de la tesis se mostrará como se utilizan estos resultados, trabajando finalmente con las ecuaciones paramétricas del círculo.

3.2 TRAZADORES CÚBICOS (SPLINES)

Los splines son tiras flexibles guiadas por puntos, usadas para interpolar o aproximar superficies. En 1946, Schoenberg introdujo la interpolación segmentaría utilizando polinomios, los Splines. Pero ni su poder de cálculo ni su flexibilidad geométrica fue apreciada (o necesitada) en ese entonces.

En la película Terminator 2, las transformaciones del adversario de Arnold Schwarzenegger son realmente un poco de magia matemática: Son Splines ó B-splines, que también son

responsables de la transformación más importante que ha tenido el diseño geométrico desde la era de los modelos de arcilla. Hoy en día, los Splines permiten representaciones matemáticas de las superficies que sería imposible realizar a mano. Por ejemplo, en la empresa Boeing, cálculos que implican de 30 a 50,000 puntos de referencia son rutinarios. ¡Imagínese el medir, el trazar y el suavizar 50,000 puntos a mano!

3.2.1 Interpolación Polinomial

La interpolación consiste en obtener una función que corresponda a una serie de datos conocidos. Una de las clases de funciones más útiles y mejor conocidas es la de los polinomios algebraicos, es decir el conjunto de funciones de la forma,

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 \tag{3.2}$$

donde n es un entero no negativo y $a_n, a_{n-1}, \ldots, a_1, a_0$ son constantes reales.

Una de las razones principales, por la cual se utilizan polinomios en la interpolación de funciones, es que la derivada y la integral de un polinomio son fáciles de determinar y también son polinomios. Por esta y otras razones más, con frecuencia se usan los polinomios para aproximar a las funciones continuas.

Existen varios tipos de interpolación polinomial, entre ellas las más comunes son:
1. Polinomio de Lagrange.
2. Polinomio de Diferencias Divididas de Newton.
3. Polinomio de Hermite.
4. Mínimos Cuadrados.

Hay que tomar en cuenta que para interpolar un polinomio de orden n, se necesitan al menos $n+1$ puntos. Además, para un conjunto numeroso de puntos no es muy útil calcular el polinomio interpolante que pasa por estos puntos, pues éste tiende a tener grandes

oscilaciones. Es más aconsejable hacer una interpolación secuencial de grado bajo sobre subconjuntos más pequeños del total de puntos, definiendo así una función segmentaria.

3.2.2 Interpolación por Splines

La interpolación segmentaria más útil y de uso generalizado en diversos campos tales como, la robótica, el diseño, los gráficos por computadora, la economía, etc., es la que se realiza mediante polinomios de grado tres llamados trazadores cúbicos ó *splines*. La gran ventaja es que la conjunción de los trazadores (interpolantes ó polinomios cúbicos) forman una función continua (diferenciable en todo su rango), sin la necesidad de más datos de entrada que los mismos puntos por donde pasa la función [7]. Lo anterior no es posible para otras técnicas, como por ejemplo, los polinomios de Hermite, donde sí se puede formar una función a trozos continuamente diferenciables, sin embargo, hay que conocer la derivada de la función en los puntos de intersección de cada trazo, dato que muchas veces se desconoce.

La selección de un polinomio cúbico para realizar la interpolación es debido a que ofrecen suficiente flexibilidad para garantizar que el interpolante no sólo sea continuamente diferenciable, sino que además tenga una segunda derivada continua en el intervalo. Sin embargo, en la construcción del trazador cúbico, no se supone que las derivadas del interpolante concuerdan con las de la función (función que muchas veces no se conoce), ni siquiera en los nodos (o intersecciones). En la figura 3.3 se muestra una comparación de las Splines contra la interpolación de Lagrange, donde se puede notar la diferencia en cuanto a suavidad.

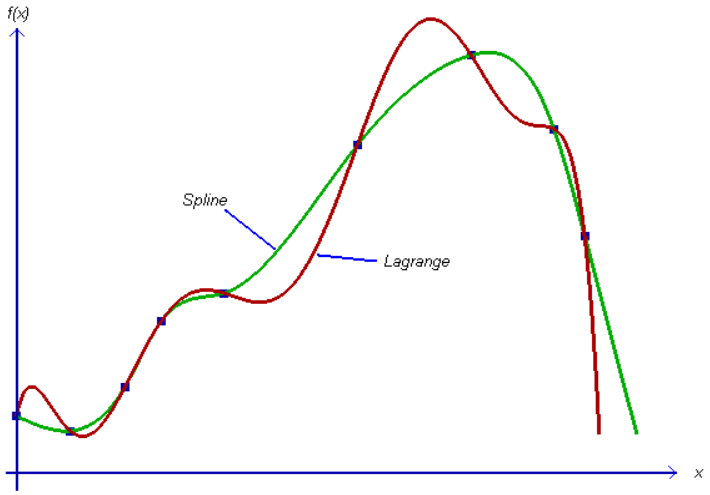


Figura 3.3. Splines vs Lagrange.

Definición de Spline:

Dada una función f definida en $[a, b]$ y un conjunto de nodos $a = x_0 < x_1 < \ ... < x_n = b$ un interpolante de trazador cúbico S para f es una función que cumple con las condiciones siguientes [7]:

a. $S(x)$ es un polinomio cúbico denotado por $S_j(x)$ en el intervalo $[x_j, x_{j+1}]$ para cada $j = 0,1,2,..., n-1;$

b. $S(x_j) = f(x_j)$ para cada $j = 0, 1, 2,..., n;$

c. $S_{j+1}(x_{j+1}) = S_j(x_{j+1})$ para cada $j = 0, 1, 2,..., n-2;$ (lo que asegura la continuidad)

d. $S'_{j+1}(x_{j+1}) = S'_j(x_{j+1})$ para cada $j = 0, 1, 2,..., n-2$; (lo que asegura diferenciabilidad en los puntos).

e. $S''_{j+1}(x_{j+1}) = S''_j(x_{j+1})$ para cada $j = 0, 1, 2,..., n-2$; (lo que asegura que no hay cambios de concavidad en los nodos o puntos).

f. Se satisface una de las siguientes condiciones a la frontera:

   (i) $S''(x_0) = S''(x_n) = 0$ (frontera libre o natural)

   (ii) $S'(x_0) = f'(x_0)$ y $S'(x_n) = f'(x_n)$ (frontera sujeta)

Cuando se presentan las condiciones de frontera libre, el trazador recibe el nombre de **trazador natural** y su gráfica se aproxima a la forma que adoptaría una varilla larga y flexible si la hiciéramos pasar por los puntos dados.

En términos generales, en las condiciones de frontera sujeta se logran aproximaciones más exactas, ya que abarcan más información acerca de la función. Pero para que se cumpla este tipo de condición de frontera, se requiere tener los valores de la derivada en los extremos o bien, una buena aproximación de ellos.

Si se quiere construir el interpolante del trazador cúbico de determinada función *f*, se aplican las condiciones de la definición a los polinomios cúbicos [7].

3.2.3 Algoritmo de Spline natural

Para construir el interpolante de trazador cúbico *S* de la función *f*, que se define en los números $x_0 < x_1 < ... < x_n$ y que satisface $S''(x_0) = S''(x_n) = 0$:

Entrada $n; x_0, x_1, \ldots, x_n;$ $a0 = f(x0),\ a1 = f(x1),\ \ldots,\ an = f(xn)$.

Salida $a_j,\ b_j,\ c_j,\ d_j,$ para $j = 0, 1, \ldots, n-1$.

(*nota:* $S(x) = S_j(x) = a_j + b_j (x - x_j) + c_j (x - x_j)^2 + d_j (x - x_j)^3$ para $x_j \le x \le x_{j+1}$) .

Paso 1 Para $i = 0, 1, \ldots, n-1$ tomar $h_i = x_{i+1} + x_i$.

Paso 2 Para $i = 1, 2, \ldots, n-1$ tomar

$$\alpha_i = \frac{3}{h_i}(a_{i+1} - a_i) - \frac{3}{h_{i-1}}(a_i - a_{i-1}) .$$

Paso 3 Tomar $l_0 = 1$;

$\mu_0 = 0$;

$z_0 = 0$.

Paso 4 Para $i = 1, 2, \ldots, n-1$ tomar

$l_i = 2 (x_{i+1} + x_i) - h_{i-1} \mu_{i-1}$;

$\mu_i = h_i / l_i$;

$z_i = (\alpha i - h_{i-1}\, z_{i-1})/\, l_i$.

Paso 5 Tomar $l_n = 1$;

$z_0 = 0$;

$c_n = 0$.

Paso 6 Para $j = n\text{-}1, n\text{-}2, \ldots, 0$ tomar

$c_j = z_j - \mu_j\, c_{j+1}$;

$b_j = (a_{j+1} - a_j)/h_j - h_j\,(c_{j+1} + 2c_j)/3$;

$d_j = (c_{j+1} - c_j)/3h_j$.

Paso 7 Salida (a_j, b_j, c_j, d_j, para $j = 0, 1, \ldots, n\text{-}1$);

Parar.

Este algoritmo fue implementado en Matlab, sin embargo, es importante mencionar que debido a que la curva o trayectoria no es específicamente una función, fue necesaria la aplicación de curvas paramétricas.

3.3 CURVAS PARAMÉTRICAS

La interpolación por Spline por si sola no sirve para representar curvas coma la de la figura 3.4, ya que esta curva no puede expresarse en función de una variable coordenada a partir de otra, esto es; a cada elemento de *x* no le corresponde uno y únicamente un elemento de *y*.

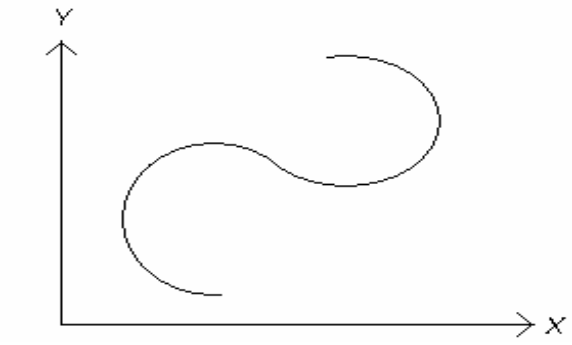


Figura 3.4. Curva paramétrica.

En esta sección se explicara cómo representar curvas generales aplicando un parámetro para expresar las variables de las coordenadas *x* e *y*. Este procedimiento se puede aplicar también para representar las curvas y superficies generales en el espacio.

Un método paramétrico sencillo con el que se determina un polinomio o un polinomio fragmentario para conectar los puntos (x_0, y_0), (x_1, y_1), . . . , (x_n, y_n), sin la condición necesaria para una función de que $x_0 < x_1 < \ ... < x_n$, consiste en usar un parámetro t en un intervalo $[t_0, t_n]$ con $t_0 < t_1 < \ ... < t_n$, y construir las funciones de aproximación mediante [7]:

$$x_i = x(t_i) \qquad \text{y} \qquad y_i = y(t_i) \qquad \text{para cada} \qquad i = 0, 1, \ldots, n.$$

El siguiente ejemplo demuestra la técnica en el caso de que las dos funciones aproximantes sean polinomios interpolantes de Lagrange:

Para obtener un par de polinomios de Lagrange para aproximar la curva que pase por los puntos dados, la selección del parámetro admite flexibilidad y se escogen los puntos $\{t_i\}^{\infty}_{i=0}$ igualmente espaciados en el intervalo dado $[a, b]$. En este ejemplo se cuenta con los datos de la tabla 3.1.

Tabla 3.1. Datos para interpolación de Lagrange.

i	0	1	2	3	4
t_i	0	0.25	0.5	0.75	1
x_i	-1	0	1	0	1
y_i	0	1	0.5	0	-1

Estos datos generan los polinomios interpolantes de Lagrange

$$x(t) = (((64t - \frac{352}{3})t + 60)t - \frac{14}{3})t - 1$$

y

$$y(t) = (((-\frac{64}{3}t + 48)t - \frac{116}{3})t + 11)t$$

Al graficar este sistema paramétrico se obtiene la gráfica de la figura 3.5. Aunque esta curva pasa por los puntos requeridos es una aproximación burda a una trayectoria requerida. Una aproximación más exacta requeriría más nodos, con el consecuente aumento de cálculos.

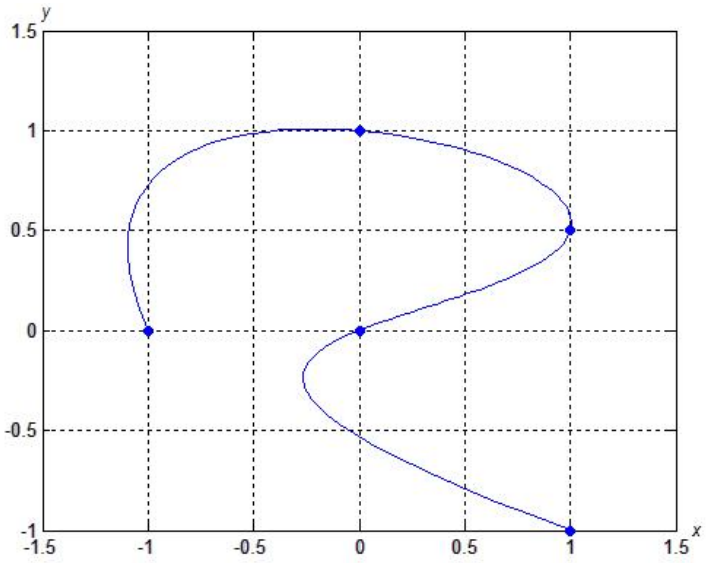


Figura 3.5. Paramétrica con Lagrange.

En la figura 3.6, se muestra la gráfica resultante al realizar la misma interpolación paramétrica, pero esta vez utilizando trazadores cúbicos (Splines).

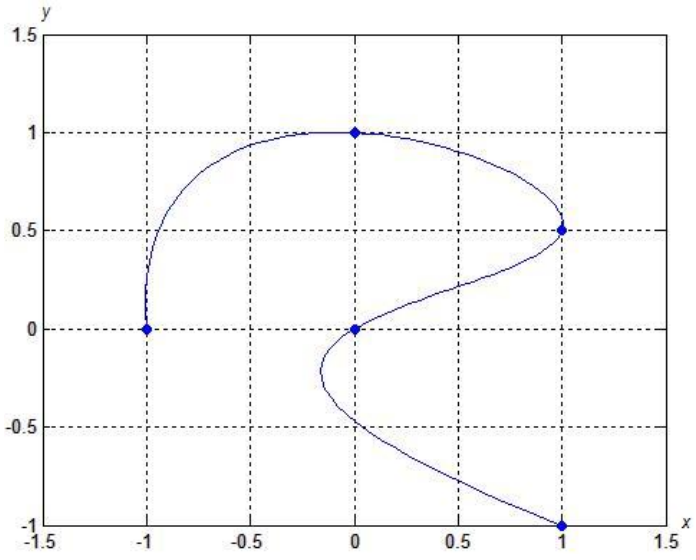


Figura 3.6. Paramétrica con Splines.

Finalmente en la figura 3.7, se muestra una comparación de ambas curvas paramétricas, notando cómo para las Splines la curva es más suave que la curva de Lagrange.

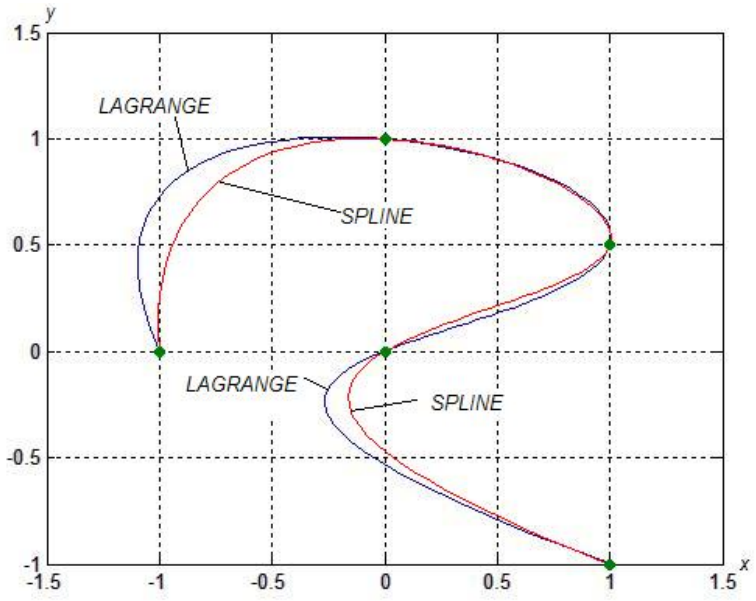


Figura 3.7. Paramétricas, Lagrange vs Splines.

Así se puede concluir de este capítulo que al utilizar interpolación paramétrica, ya sea la circular o con Splines, se garantiza que es posible generar prácticamente cualquier trayectoria en el plano o en el espacio y más aún, para las Splines la interpolación tiene las propiedades de estar compuesta por curvas suaves y continuas.

CAPÍTULO 4
ROBÓTICA DE MANIPULADORES

En este capítulo se presenta una breve introducción a la robótica, específicamente la robótica de manipuladores. Se muestran temas de interés para este trabajo, aunque de manera muy general, sirven para entender algunos conceptos básicos asociados a la robótica de un manipulador como lo son la cinemática y la generación de trayectorias. Se da una breve introducción a la cinemática directa e inversa, así como al tipo de trayectorias más utilizadas en la robótica de manipuladores.

4.1 INTRODUCCIÓN A LA ROBÓTICA

¿ Pueden los robots mejorar la vida de las personas? Esta es una vieja interrogante desde la aparición de máquinas con apariencia humana. No es sencillo definir lo que es un Robot, esta palabra tiene su origen en el idioma checo con dos palabras: Robota y Robotnik, que significan "trabajo forzado" y "esclavo" respectivamente. La palabra ***"Robot"*** aparece por primera vez en una obra satírica denominada: "The Rossum Universal Robots", escrita por el dramaturgo checo Karel Capel, en 1921. En dicha obra los robots son unos personajes con apariencia humanoide que son fieles obedientes a las ordenes de su maestro.

El término robótica tiene su origen en la palabra robot. De esta forma, lo que hoy conocemos como Robótica se refiere al área del conocimiento asociado al estudio, desarrollo y aplicación de los robots.

En 1950, Isaac Asimov presenta su famosa novela "Yo, Robot", en donde define las tres leyes de la robótica [21]:

- Primera Ley:

Un robot no puede lastimar a un ser humano o por su inacción dejar que un ser humano sea lastimado.

- Segunda Ley:

Un robot debe obedecer las ordenes dadas por un ser humano excepto cuando estas ordenes estén en oposición con la primera ley.

- Tercera Ley:

Un robot debe proteger su propia existencia siempre y cuando ésta protección no difiera con las dos primeras leyes.

Muchos han sido los esfuerzos para que los desarrollos tecnológicos puedan acercarse un poco a las habilidades de los robots de las películas y libros de ciencia-ficción. En la vida real hay una serie de problemas tecnológicos que han dificultado la evolución de este tipo de máquinas. Sin embargo, con el desarrollo de la electrónica y los sistemas informáticos se han logrado avances significativos propiciando en todo el mundo el uso de los robots en un entorno industrial, principalmente.

Hoy en día, los robots industriales son máquinas que flexibilizan la automatización de muchos procesos o bien, en la mayoría de los casos, permiten la realización de tareas bajo situaciones de alto riesgo e inadecuadas para los seres humanos. Actualmente, uno de los parámetros que ayudan a determinar el nivel tecnológico de un país, es su grado de robotización en sus instalaciones productivas. Lo que ayuda a lograr una estimación de competitividad y solidez tecnológica en las empresas. Sin embargo, hay que tener mucho cuidado en este aspecto, ya que pudiera darse una situación de alta robotización bajo un esquema ineficiente, en donde el robot pasa la mayoría del tiempo sin realizar trabajo.

En México se realiza el estudio de los robots industriales aunque no con suficiente profundidad. Existen una serie de problemas asociados a la investigación y al desarrollo de la

robótica que no ha podido ser resuelto. En general, los intereses de la Academia son diferentes a los intereses del sector de la transformación, y no se ha logrado una vinculación que propicie un acercamiento de necesidades y soluciones. Los robots se estudian en México desde hace poco más de dos décadas. Sin embargo, hay escasos desarrollos que han logrado una aplicación industrial exitosa [14]. En este trabajo se propone una metodología, que es una pequeña aportación a la robótica de manipuladores, que consiste en facilitar la programación de trayectorias de robots industriales. Además, se da una breve introducción a la robótica de manipuladores, como lo es la cinemática y la generación de trayectorias.

4.2 CINEMÁTICA DE UN MANIPULADOR

La cinemática de un manipulador estudia el movimiento del mismo con respecto a un sistema de referencia. Así, la cinemática se interesa por la descripción analítica del movimiento espacial del robot como una función del tiempo, y en particular por las relaciones entre la posición y la orientación del extremo final del robot con los valores que toman sus coordenadas articulares [8].

Existen dos problemas fundamentales a resolver en la cinemática del manipulador; el primero de ellos se conoce como el problema de la ***cinemática directa*** y consiste en determinar cuál es la posición y orientación del extremo final del robot (herramienta), con respecto a un sistema de coordenadas que se toma como referencia, conocidos los valores de las articulaciones y los parámetros geométricos de los elementos del robot; el segundo, es el de la ***cinemática inversa***, que resuelve la configuración (posición de las articulaciones) que debe adoptar el robot para una posición y orientación del extremo conocidas.

4.2.1 Cinemática Directa

Denavit y Hartenberg [8] propusieron un método sistemático para describir y representar la geometría espacial de los elementos de una cadena cinemática, y en particular de un robot, con respecto a un sistema de referencia fijo. Este método utiliza una matriz de transformación

homogénea para describir la relación espacial entre dos elementos rígidos adyacentes, reduciéndose el problema de cinemática directa a encontrar una matriz de transformación homogénea de 4 x 4 que relacione la localización espacial del extremo del robot con respecto al sistema de coordenadas de su base.

Por otra parte, la cinemática del robot trata también de encontrar las relaciones entre las velocidades del movimiento de las articulaciones y las del extremo. Esta relación viene dada por el modelo diferencial expresado mediante la matriz Jacobiana.

Se utilizan fundamentalmente el álgebra vectorial y matricial para representar y describir la localización de un objeto en el espacio tridimensional con respecto a un sistema de referencia fijo. Dado que un robot se puede considerar como una cadena cinemática formada por objetos rígidos o eslabones unidos entre si mediante articulaciones, se puede establecer un sistema de referencia fijo situado en la base del robot y describir la localización de cada uno de los eslabones con respecto a dicho sistema de referencia. De esta forma el problema de cinemática directa se reduce a encontrar una matriz de transformación homogénea *T* que relacione la posición y orientación del extremo del robot respecto del sistema de referencia fijo situado en la base del mismo. Esta matriz *T* será función de las coordenadas articulares.

Para encontrar la matriz *T* generalmente se utiliza el método de Denavit Hartenberg (**D-H**), esto no quiere decir que sea la única técnica, sin embargo, es la más popular y no tan complicada para entender.

Algoritmo de Denavit Hartenberg:

D-H 1.- Numerar los eslabones comenzando con *1* (primer eslabón móvil de la cadena) y acabando con *n* (último eslabón móvil). Se numerará como eslabón *0* a la base fija del robot.

D-H 2.- Numerar cada articulación comenzando por *1* (la correspondiente al primer grado de libertad) y acabando en *n.*

D-H 3.- Localizar el eje de cada articulación. Si ésta es rotativa, el eje será su propio eje de giro. Si es prismática, será el eje a lo largo del cual se produce el desplazamiento.

D-H 4.- Para *i* de *0* a *n-1* situar el eje z_i sobre el eje de la articulación *i+1*.

D-H 5.- Situar el origen del sistema de la base $\{S_0\}$ en cualquier punto del eje z_0. Los ejes x_0 e y_0 se situarán de modo que formen un sistema dextrógiro con z_0.

D-H 6.- Para i de 1 a $n-1$, situar el sistema $\{S_i\}$ (solidario al eslabón i) en la intersección del eje z_i con la línea normal común a z_{i-1} y z_i. Si ambos ejes se cortasen se situaría $\{S_i\}$ en el punto de corte. Si fuesen paralelos $\{S_i\}$ se situaría en la articulación $i+1$.

D-H 7.- Situar x_i en la línea normal común a z_{i-1} y z_i.

D-H 8.- Situar y_i de modo que forme un sistema dextrógiro con x_i y z_i.

D-H 9.- Situar el sistema $\{S_n\}$ en el extremo del robot de modo que z_n coincida con la dirección de z_{n-1} y x_n sea normal a z_{n-1} y z_n.

D-H 10.- Obtener $\boldsymbol{\theta_i}$ como el ángulo que hay que girar en torno a z_{i-1} para que x_{i-1} y x_i queden paralelos.

D-H 11.- Obtener $\mathbf{d_i}$ como la distancia, medida a lo largo de z_{i-1}, que habría que desplazar $\{S_{i-1}\}$ para que x_i y x_{i-1} quedasen alineados.

D-H 12.- Obtener $\mathbf{a_i}$ como la distancia medida a lo largo de x_i (que ahora coincidiría con x_{i-1}) que habría que desplazar el nuevo $\{S_{i-1}\}$ para que su origen coincidiese con $\{S_i\}$.

D-H 13.- Obtener $\boldsymbol{\alpha_i}$ como el ángulo que habría que girar entorno a x_i (que ahora coincidiría con $\boldsymbol{x_{i-1}}$), para que el nuevo $\{S_{i-1}\}$ coincidiese totalmente con $\{S_i\}$.

D-H 14.- Obtener las matrices de transformación $\boldsymbol{^{i-1}A_i}$. Definidas como:

$$ {}^{i-1}A_i = \begin{bmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i S\theta_i & a_i C\theta_i \\ S\theta_i & C\alpha_i C\theta_i & -S\alpha_i C\theta_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1} $$

D-H 15.- Obtener la matriz de transformación entre la base y el extremo del robot

$$ T = {}^{0}A_1 \, {}^{1}A_2 \ldots {}^{n-1}A_n \tag{4.2} $$

D-H 16.- La matriz T define la orientación (submatriz de rotación) y posición (submatriz de traslación) del extremo referido a la base en función de las n coordenadas articulares.

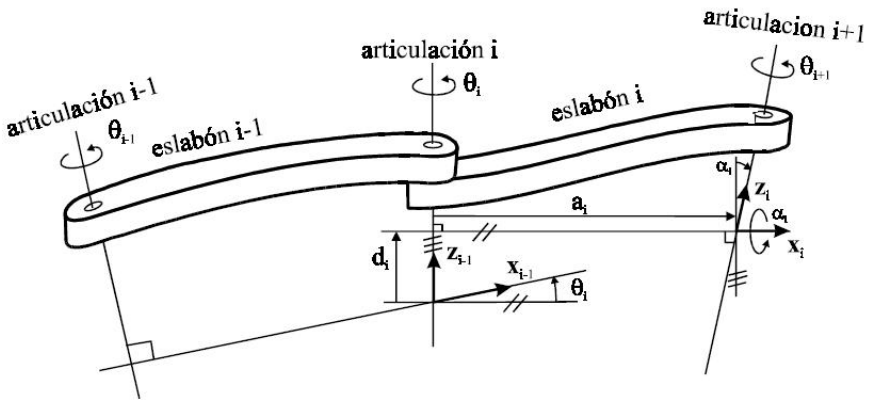


*Figura 4.1. Ejemplo de parámetros **D-H** para un eslabón giratorio.*

Los cuatro parámetros de **D-H** (θ_i, d_i, a_i, α_i) dependen únicamente de las características geométricas de cada eslabón y de las articulaciones que le unen con el anterior y siguiente, en concreto estos representan:

- **θ_i**: Es el ángulo que forman los ejes x_{i-1} y x_i medido en un plano perpendicular al eje z_{i-1}, utilizando la regla de la mano derecha. Se trata de un parámetro variable en articulaciones giratorias.
- **d_i**: Es la distancia a lo largo del eje z_{i-1} desde el origen del sistema de coordenadas (i-1)-ésimo hasta la intersección del eje z_{i-1} con el eje x_i. Se trata de un parámetro variable en articulaciones prismáticas.
- **a_i**: Es la distancia a lo largo del eje x_i que va desde la intersección del eje z_{i-1} con el eje x_i hasta el origen del sistema i-ésimo, en el caso de articulaciones giratorias. En el caso de articulaciones prismáticas, se calcula como la distancia más corta entre los ejes z_{i-1} y z_i.
- **α_i**: Es el ángulo de separación del eje z_{i-1} y el eje z_i, medido en un plano perpendicular al eje x_i, utilizando la regla de la mano derecha.

Ejemplo:

Con fin de ilustrar la utilización del método de **D-H**, para la solución del problema cinemático directo, se realiza el desarrollo para un robot cilíndrico.

En primer lugar se localizan los sistemas de referencia de cada articulación del robot como se ilustra en la figura 4.2.

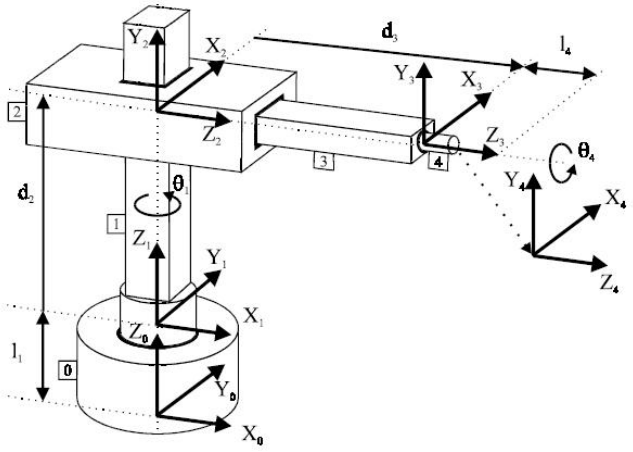


*Figura 4.2. Bosquejo **D-H**, robot cilíndrico.*

Posteriormente se determinan los parámetros de **D-H** del robot, con los que se construye la tabla 4.1.

Tabla 4.1. Parámetros **D-H** para el robot cilíndrico de la figura 4.2.

Articulación	**θ**	**d**	**a**	**α**
1	q_1	l_1	*0*	*0*
2	$\pi/2$	d_2	*0*	$\pi/2$
3	*0*	d_3	*0*	*0*
4	q_4	l_4	*0*	*0*

Una vez calculados los parámetros de cada eslabón, se calculan las matrices ${}^{i-1}A_i$, sustituyendo en la fórmula 4.1, de la siguiente manera:

$${}^{0}A_1 = \begin{bmatrix} Cq_1 & -Sq_1 & 0 & 0 \\ Sq_1 & Cq_1 & 0 & 0 \\ 0 & 0 & 1 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^{1}A_2 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$ {}^{2}A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^{3}A_4 = \begin{bmatrix} Cq_4 & -Sq_4 & 0 & 0 \\ Sq_4 & Cq_4 & 0 & 0 \\ 0 & 1 & 1 & l_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Así se puede calcular la matriz *T*, que indica la localización del sistema final con respecto al sistema de referencia de la base del robot utilizando la fórmula 4.2.

$$ T = {}^{0}A_4 = {}^{0}A_1 {}^{1}A_2 {}^{2}A_3 {}^{3}A_4 = \begin{bmatrix} -Sq_1Cq_4 & Sq_1Sq_4 & Cq_1 & Cq_1(d_3 + l_4) \\ Cq_1Cq_4 & -Cq_1Sq_4 & Sq_1 & Sq_1(d_3 + l_4) \\ Sq_4 & Cq_4 & 0 & d_2 + l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

4.2.2 Cinemática Inversa

En esta sección se plantea el segundo problema de la cinemática del robot; la cinemática inversa. Con el fin de controlar la posición y orientación del efector final de un robot para alcanzar su objetivo, es muy importante la solución de la cinemática inversa. En otras palabras, dada la posición y orientación del efector final de un brazo robot de *n* ejes y sus parámetros de articulación y elementos, es de interés encontrar los ángulos de articulación correspondientes $q = (q_1, q_2, \ldots, q_n)$ del robot de manera que el efector final se posicione y oriente de la forma deseada.

En general el problema cinemático inverso se puede resolver por diversos métodos, tales como la transformación inversa (Paul y col., 1981), el álgebra de tornillo (Kohli y Soni, 1975), matrices duales (Denavit, 1956), cuaterniones duales (Yang y Freudenstein, 1964), iterativo (Uicker y col., 1964), y métodos geométricos (Lee y Ziegler, 1984) [8].

En esta sección se presenta como ejemplo un método geométrico para la solución del problema cinemático inverso de un manipulador de tres elementos con articulaciones giratorias o tres grados de libertad articulados, figura 4.3.

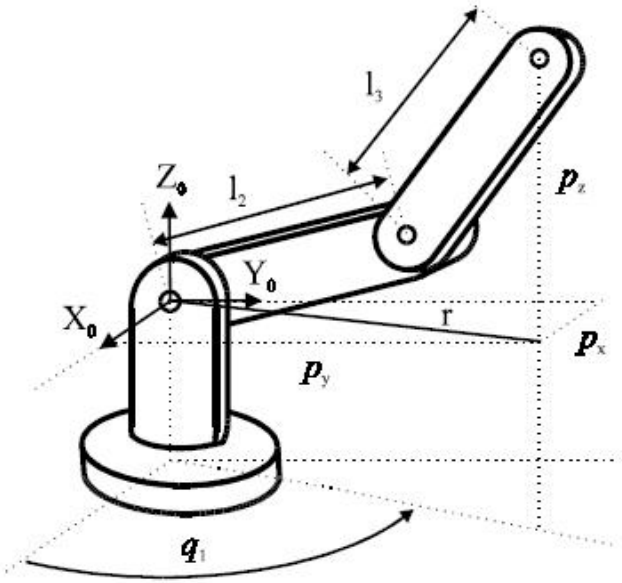


Figura 4.3. Robot articular de 3 grados de libertad.

El procedimiento en este caso es sencillo, debido a que sólo son tres grados de libertad, primero se calcula q_1 utilizando la tangente del ángulo que forman la posición (*px, py*),

$$q_1 = \tan^{-1}(\frac{py}{px})$$

Posteriormente con ayuda de la ley de coseno, igualdades trigonométricas y el teorema de Pitágoras se obtiene el valor de q_3 como se muestra a continuación,

$$r^2 = px^2 + py^2$$

$$r^2 + pz^2 = l_2^2 + l_3^2 + 2l_2 l_3 \cos q_3$$

$$\cos q_3 = \frac{px^2 + py^2 + pz^2 - l_2^2 - l_3^2}{2l_2 l_3}$$

$$\sin q_3 = \pm\sqrt{1 - \cos^2 q_3}$$

$$q_3 = \arctan\left(\frac{\sin q_3}{\cos q_3}\right)$$

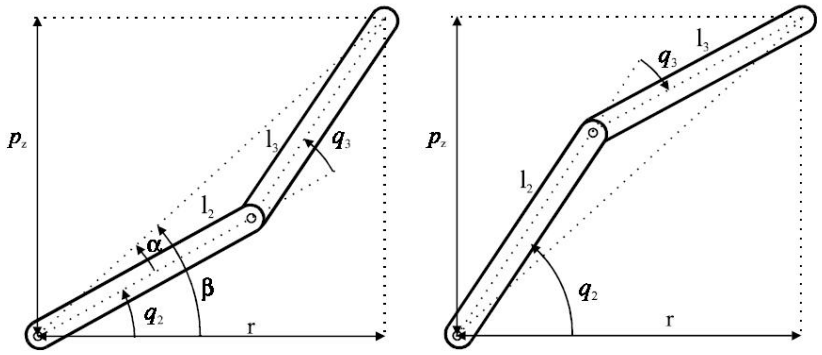


Figura 4.4. Bosquejo de dos brazos del robot articular.

Finalmente calculando q_2, con ayuda de los diagramas de la figura 4.4 se tiene que,

$$q_2 = \beta - \alpha$$

$$\beta = \arctan\left(\frac{pz}{r}\right) = \arctan\left(\frac{pz}{\pm\sqrt{px^2 + py^2}}\right)$$

$$\alpha = \arctan\left(\frac{l_3 senq_3}{l_2 + l_3 \cos q_3}\right)$$

$$q_2 = \arctan\left(\frac{pz}{\pm\sqrt{px^2 + py^2}}\right) - \arctan\left(\frac{l_3 senq_3}{l_2 + l_3 \cos q_3}\right)$$

De esta forma se obtienen los valores de los ángulos q_1, q_2, q_3 en función de la posición *px, py, pz*. En este caso el desarrollo es sencillo debido a que sólo son tres grados de libertad, sin embargo, normalmente un robot manipulador es de más de tres grados, lo cual implica que

a mayor numero de grados de libertad más complicada es la obtención de los parámetros de la cinemática inversa.

Para tener una idea de lo que implica el resolver la cinemática inversa de un manipulador más complejo, considere el caso de un robot tipo Puma (figura 4.5) de seis grados de libertad. Hay que resolver el sistema y considerar que existen cuatro posibles soluciones para las tres primeras articulaciones y para cada una de las cuatro soluciones hay dos soluciones posibles para las tres últimas, lo que da un total de 8 posibles soluciones para la cinemática inversa de este robot.

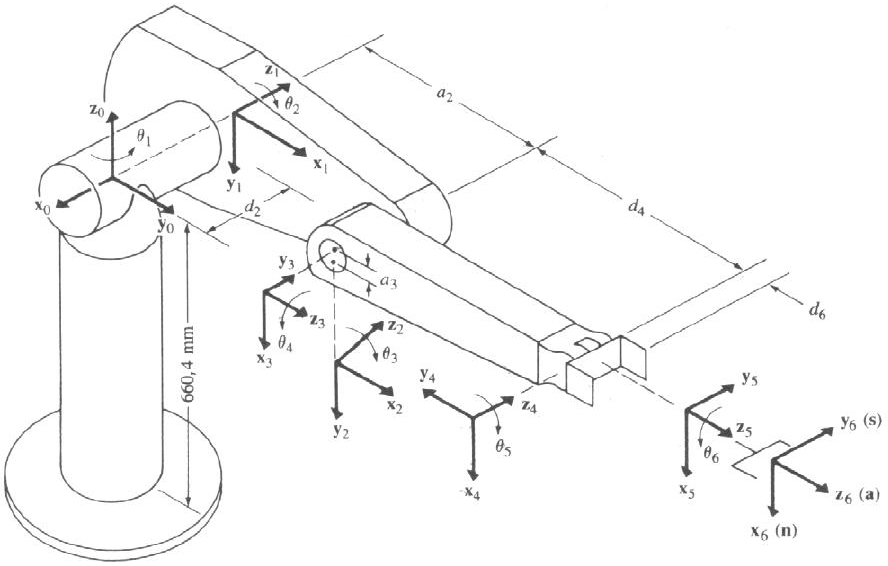


Figura 4.5. Robot tipo PUMA de 6 grados de libertad.

Un término fuertemente involucrado en la robótica es la dinámica. El modelo dinámico de un robot tiene por objetivo conocer la relación entre el movimiento del robot y las fuerzas implicadas en el mismo. Sin embargo, para este trabajo no se involucra el modelo dinámico del manipulador, ya que no es necesario para el objetivo que se requiere.

Otros términos involucrados son el control cinemático y el control dinámico. Donde el control cinemático se encarga de generar las trayectorias que idealmente debe seguir el robot, mientras que el control dinámico tiene por misión procurar que las trayectorias seguidas sean lo más parecidas a la trayectoria deseada en control cinemático. Al igual que sólo se maneja el

modelo cinemático, sólo se manejará el control cinemático, específicamente la generación de trayectorias asociadas al modelo cinemático.

4.2.3 Generación de Trayectorias

Para realizar una tarea determinada, el robot debe moverse de un punto inicial a un punto final. Este movimiento puede ser realizado según infinitas trayectorias especiales [8]. De todas ellas, sólo las más utilizadas en diferentes tareas y aplicaciones son las que en la práctica incorporan los robots comerciales. De esta forma se puede decir que las trayectorias más utilizadas son, punto a punto, coordinadas y continuas.

En las *trayectorias punto a punto*, cada articulación evoluciona desde su posición inicial hasta la final, sin realizar consideración alguna sobre el estado o evolución de las demás articulaciones. El movimiento puede ser eje por eje o simultáneo de ejes.

Las *trayectorias coordinadas*, también llamadas isócronas, son aquellas donde se toma como base al eje más lento, averiguando cuál es esta articulación y cuanto tiempo invertirá, para que el resto de los ejes realicen su movimiento en el mismo tiempo. Se tiene así que todas las articulaciones se coordinan comenzando y terminado su movimiento a la vez, adaptándose todas a la más lenta.

Las *trayectorias continuas*, son aquellas en las que el usuario determina la trayectoria que desea que el robot describa, determinada por líneas o curvas. El resultado será que cada articulación sigue un movimiento aparentemente complicado con posibles cambios de dirección y velocidad y sin aparente coordinación con el resto de las articulaciones, teniendo como resultado que el robot describirá la trayectoria deseada.

En la figura 4.6, se muestra un ejemplo del tipo de trayectorias que puede seguir un robot manipulador tipo Scara.

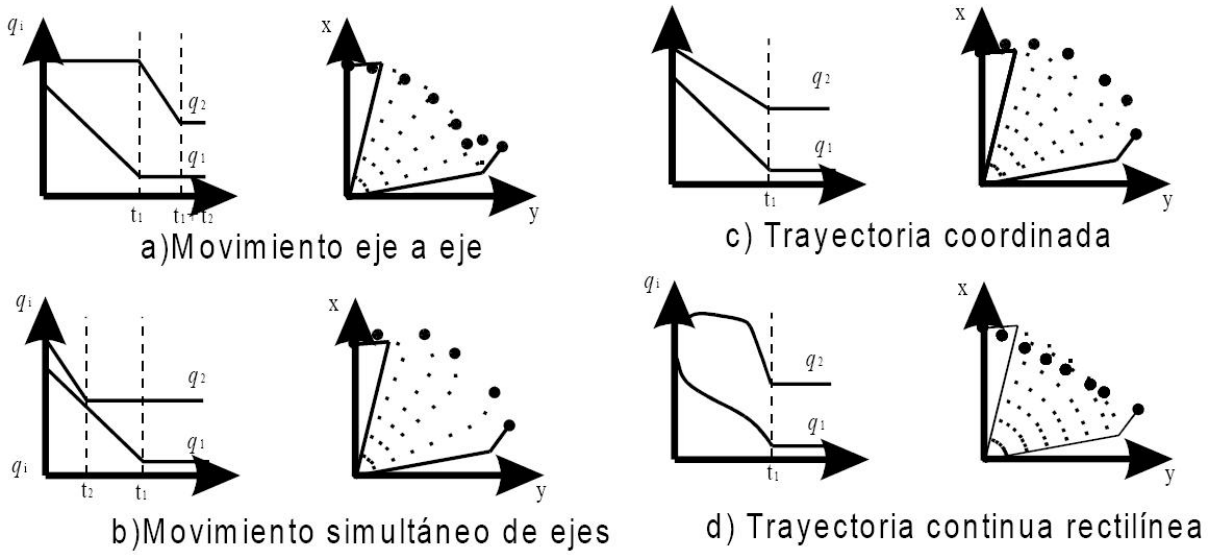


Figura 4.6. Ejemplo de trayectorias para un robot Scara.

La generación de trayectorias tiene la función de unir una sucesión de puntos en el espacio articular por los que se quiere que pasen las articulaciones del robot en un instante determinado. Además, junto con las condiciones de tiempo-posición, es conveniente añadir restricciones en la velocidad y aceleración de paso por los puntos, de manera que *se asegure la suavidad de la trayectoria y se limiten las velocidades y aceleraciones máximas*. Para ello deberá seleccionarse algún tipo de función (frecuentemente polinómica) cuyos parámetros se ajustarán al imponer las condiciones de contorno: posiciones, velocidades y aceleraciones. Las funciones interpoladoras utilizadas con mayor frecuencia son los interpoladores lineales, interpoladores cúbicos e interpoladores a tramos.

Debido a la naturaleza de interdisciplinariedad de este trabajo de tesis, hasta estos momentos se han dado solamente las bases para el entendimiento de la tesis, que es un buen preámbulo para poder entrar al desarrollo de la misma en los capítulos siguientes.

CAPÍTULOS 5-6

DESARROLLO

CAPÍTULO 5

GENERACIÓN DEL CONTORNO E INTERPOLACIÓN

En este capítulo se describe el procedimiento utilizado para generar la trayectoria del robot manipulador a partir de la imagen digital. Se divide en dos partes: El procesamiento de la imagen, donde se aplican las técnicas descritas, tales como la binarización por detección de umbral y la detección del contorno con un algoritmo propio pero similar al del código de la cadena. Por otro lado se le da seguimiento a los datos obtenidos del procesamiento de imágenes, adecuando los resultados y aplicándoles dos tipos de interpolaciones, la interpolación circular y la interpolación por splines, que a futuro servirán para generar la trayectoria para el robot manipulador.

Antes de comenzar con el desarrollo se describen las características de adquisición de la imagen, que sirve entre otras cosas, para determinar el factor de escalamiento que básicamente describe la correspondencia del espacio real (milímetros) al espacio de la imagen (píxeles).

5.1 ADQUISICIÓN DE IMAGEN

Se utilizan imágenes obtenidas en laboratorio, con una cámara CCD con resolución de 16 bits en escala de grises, acoplada a una tarjeta de adquisición para la computadora. Las imágenes fueron adquiridas bajo condiciones normales de iluminación, lo único que se controló fueron los brillos y sombras para que no afectaran el procesamiento posterior.

Las imágenes de intensidad (niveles de gris), que es el tipo de imágenes que se manejan en este trabajo, están íntimamente ligadas al concepto de luminosidad. Sin embargo, el tema de la iluminación y su óptica es una ciencia muy amplia, que sí es necesario aplicarla a la adquisición de imágenes hay que estudiarla muy detenidamente. En este trabajo no se trata el

tema de la adquisición muy a fondo, ya que este punto no es el principal para el desarrollo de la tesis, sin embargo, a continuación se da una breve descripción del sistema de adquisición.

La cámara utilizada es una CCD modelo XC-ST50 de Sony con las siguientes características:

Tabla 5.1 Características de la cámara.

Dispositivo de recolección de imagen	CCD de 1/2"
Numero de píxeles efectivos	768(H) x 494(V)
Tamaño de celda (píxel)	8.4(H) x 9.8(V) μm
Tamaño de imagen	640(H) x 480(V)
Resolución píxel en imagen	16 bits

Estas características son útiles para realizar algunos cálculos, que son necesarios para calibrar o parametrizar el espacio de trabajo de la cámara, que a final de cuentas es el espacio que ocupa la imagen, dado en milímetros/píxel y que debe estar parametrizado de acuerdo al espacio de trabajo del robot.

En la figura 5.1 se muestra una imagen de la cámara utilizada y en la figura 5.2 una imagen del lente utilizado.

Figura 5.1. Cámara Sony CCD XC-ST50.

Las características del lente utilizado se muestran en la tabla 5.2:

Tabla 5.2. Características del lente.

Longitud focal		12.5~75 mm
Máximo cociente de apertura		1:1.2
Máximo formato de imagen		8.6 x 6.6 mm (Ø 11 mm)
Dimensión de objeto	12.5 mm	688 x 516 mm
	75 mm	121 x 90 mm
Operación	Iris	Manual, F1.2~16C
	Foco	Manual, 1 m~Inf.
	Zoom	Manual, 12.5~75 mm

Figura 5.2. Lente Computar M6Z1212

Las características tanto de la cámara como del lente son necesarias para conocer algunos parámetros que influyen en la adquisición de la imagen, estos parámetros se describen a continuación.

Primero se calcula la magnificación (*m*), que es una medida de la dimensión relativa de la imagen visual del objeto en el mundo físico, a la dimensión de la imagen formada en el plano detector en la cámara [3]. Entonces, la magnificación es la relación entre la dimensión de la imagen y la del objeto dada por,

$$m = \frac{H_i}{H_o} = \frac{D_i}{D_o} \tag{5.1}$$

donde:

D_i es la distancia entre el lente y el plano de la imagen, fijado por el fabricante.

D_o es la distancia entre el lente y el objeto.

H_i se determina por la dimensión de la matriz de sensores.

H_o se determina por la dimensión del plano del objeto.

Teniendo estas relaciones se puede calcular la magnificación para nuestra aplicación, considerando las dimensiones del plano del objeto y la matriz de sensores.

Las dimensiones del plano del objeto se obtienen en base a las dimensiones de los objetos que se estén manejando y del espacio de trabajo del robot. En la figura 5.3 se muestran las dimensiones del espacio de trabajo del robot y se observa cómo el objeto se ubica dentro de las mismas.

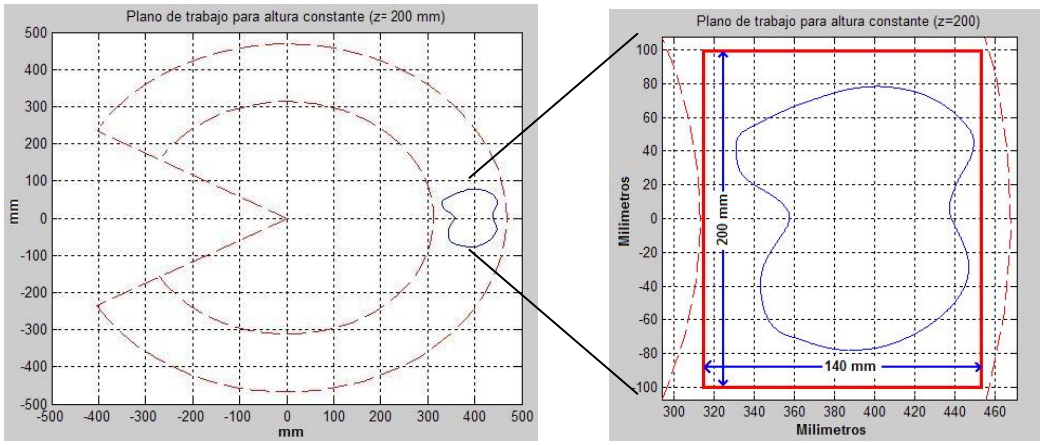


Figura 5.3. Espacio de trabajo donde actuará el robot.

De la figura 5.3 se puede observar que el espacio de trabajo para el robot es de 140 x 200 *mm*, por lo que el plano del objeto debe ser como máximo de las mismas dimensiones.

Para obtener las dimensiones del plano del objeto, primero se coloca la cámara a una distancia ajustable D_0 sobre una mesa de trabajo, utilizando un vernier se obtienen las medidas del plano del objeto corroborándolo con la imagen en pantalla de la computadora, finalmente se ajusta D_0 hasta que las dimensiones del plano del objeto estén dentro de las dimensiones del espacio de trabajo del robot.

De esta forma se obtienen las dimensiones del plano del objeto, que para este caso son de 183(H) x 137(V) *mm*. Cabe señalar que no son ninguna medida en especial, basta que estén dentro del espacio de trabajo del robot y que sean mayores que las dimensiones de los objetos que se estén manejando. En la figura 5.4 se representan estas dimensiones.

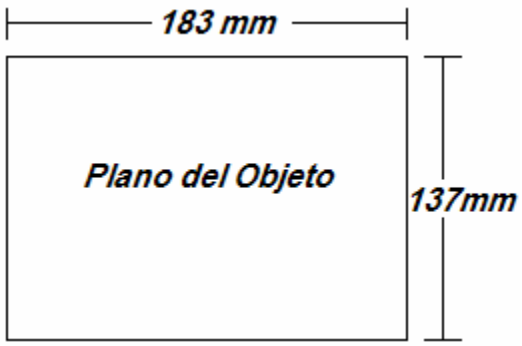


Figura 5.4. Plano del objeto.

Para obtener las dimensiones del plano del sensor hay que considerar el numero de píxeles efectivos y la dimensión de cada píxel, estos datos se obtienen de la tabla 5.1 (características de la cámara),

Numero de píxeles 768(H) x 494(V)
Dimensión de píxel 8.4(H) x 9.8(V) μm

Con estos datos se obtiene la dimensión del sensor, que es de 6.4512(H) x 4.8412(V) *mm*, ver figura 5.5.

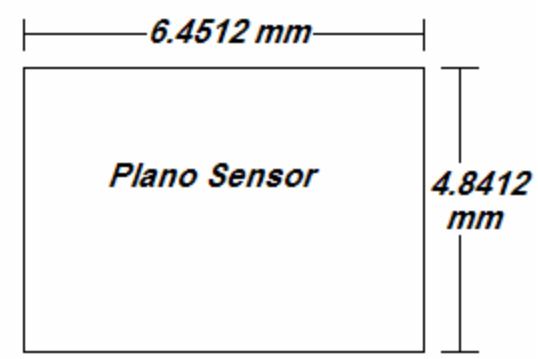


Figura 5.5. Plano del sensor.

Una vez que se obtienen estos datos, se tienen dos magnificaciones una horizontal (H) y una vertical (V), que al final se observará que son de la misma magnitud, condición que es deseable.

Los cálculos de las magnificaciones se describen a continuación; primero de la fórmula 5.1 considerando H_i y H_o se tiene que,

$$m(H) = \frac{6.4512mm}{183mm} = 0.0353$$

y,

$$m(V) = \frac{4.8412mm}{137mm} = 0.0353$$

de donde se puede notar que la magnificación carece de unidades y es menor que 1, lo que indica que es solamente una razón y que normalmente las dimensiones del sensor son mucho menores que las dimensiones del plano del objeto (la excepción es cuando se manejan planos microscópicos).

Otro termino interesante es la distancia focal *f*, que está relacionada con dos términos; la magnificación *m* y la distancia entre el lente y el objeto D_o. La fórmula que relaciona estos términos se muestra a continuación [3], [9],

$$f = \frac{D_o}{1+1/m} \tag{5.2}$$

de la cual se conocen m y f. El parámetro m está dado por la magnificación y f, la distancia focal, está dada por el ajuste manual del lente, que va de 12.5 a 75 mm (ver tabla 5.2). Llevando f de la lente hasta 12.5 mm y utilizando la fórmula 5.2 se obtiene la distancia entre la lente y el objeto D_o,

$$D_o = \frac{f}{m} + f = \frac{12.5}{0.0353} + 12.5 = 366.6mm$$

Por último se obtiene el término de la profundidad de campo, que es la distancia en la que el objeto permanece enfocado sin variar ningún parámetro de la cámara, ver figuras 5.6 y 5.7.

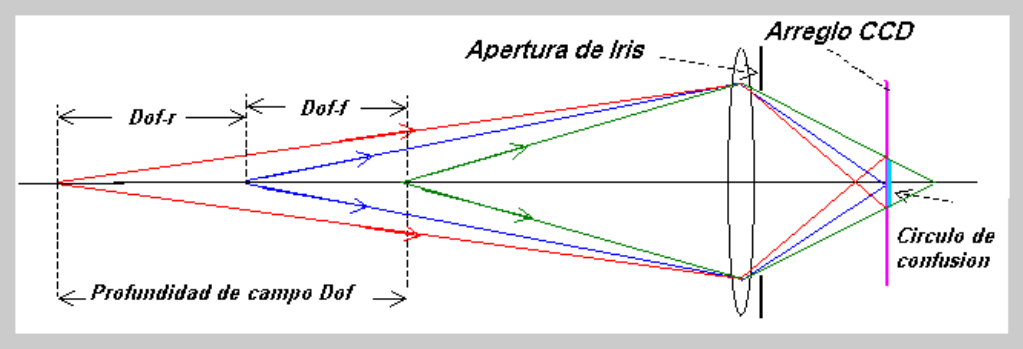


Figura 5.6. Profundidad de campo 1.

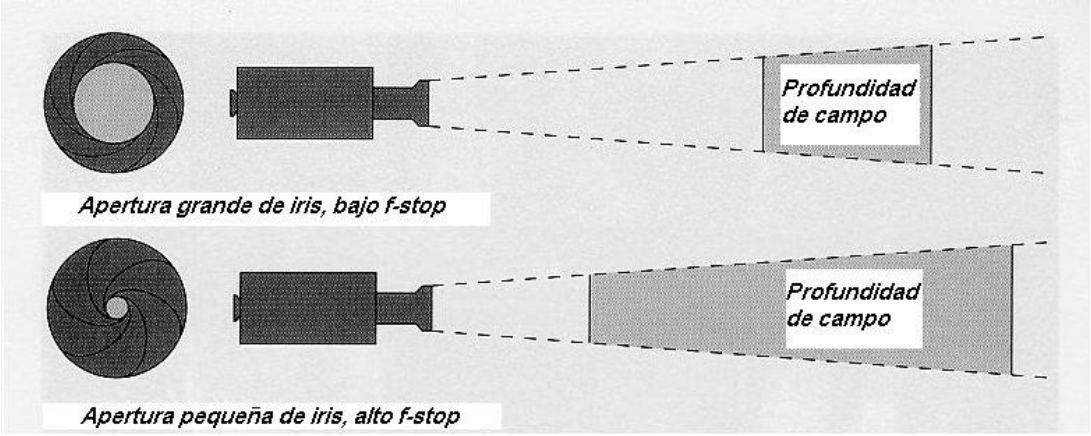


Figura 5.7. Profundidad de campo 2.

En la fórmula 5.3 se representa la profundidad de campo, *DoF* por sus siglas en ingles [9],

$$DoF = \frac{2pf_{stop}(m+1)}{m^2} \tag{5.3}$$

donde p es la dimensión del píxel (ver tabla 5.1), f_{stop} es un termino que está relacionado con la apertura del diafragma (o iris, ver tabla 5.2) y m es la magnificación, término ya explicado.

Dado que la dimensión del píxel es de 8.4(H) x 9.8(V) μm y sólo se necesita un valor de dimensión, se hará la suposición de que es cuadrado y se considerará como dimensión el valor mínimo 8.4 μm. En cuanto al valor del f_{stop}, se obtiene de la apertura de la lente que se controla manualmente y está en el rango de 1.2~16 C, (C: cerrado) para la adquisición de las imágenes de prueba el f_{stop} se sitúo en un valor aproximado de 12, finalmente utilizando estos valores y la fórmula 5.3 se tiene que,

$$DoF = \frac{2(0.0084)(12)(0.0353+1)}{0.0353^2} = 167.5mm$$

por otro lado en la figura 5.6 se observan dos parámetros relacionados con la profundidad de campo, denominados *DoFf* y *DoFr* por sus siglas en inglés, que representan la profundidad de campo frontal y posterior respectivamente, a partir del plano del objeto. La suma de estos términos representan la profundidad de campo total [10],

$$DoF = DoFf + DoFr \tag{5.4}$$

donde

$$DoFf = \frac{cf_{stop}D_o(D_o - f)}{f^2 + cf_{stop}(D_o - f)} \tag{5.5}$$

y

$$DoFr = \frac{cf_{stop} D_o (D_o - f)}{f^2 - cf_{stop} (D_o - f)} \tag{5.6}$$

de las dos últimas fórmulas el único valor desconocido es el de *c*, que se conoce como círculo de confusión, y que generalmente se considera de acuerdo a las características físicas de la cámara, para nuestro caso se toma un valor de *c*=0.0077 *mm* que se obtiene de las hojas de datos del fabricante.

Evaluando las fórmulas 5.5 y 5.6 con los valores conocidos se tiene que,

$$DoFf = \frac{(0.0077)(12)(366.6)(366.6 - 12.5)}{(12.5)^2 + (0.0077)(12)(366.6 - 12.5)} = 65.32mm$$

$$DoFr = \frac{(0.033)(12)(366.6)(366.6 - 12.5)}{(12.5)^2 - (0.033)(12)(366.6 - 12.5)} = 101.48mm$$

observando que la profundidad de campo en frente del plano del objeto es menor que la de la parte posterior, como se aprecia en la figura 5.6. Finalmente se evalúa la fórmula 5.4 obteniendo una profundidad de campo de,

$$DoF = 65.32 + 101.48 = 166.8mm$$

que es muy aproximada al valor obtenido por la fórmula 5.3.

Todos estos cálculos sólo se realizaron para tener una idea de las características de adquisición de la imagen, sin embargo, hay un parámetro que es fundamental para nuestra aplicación; la relación de escala entre el plano del objeto y el plano de la imagen (escalamiento). Para obtener esta relación, primero de la figura 5.4 se conoce la dimensión del plano del objeto que es de 183(H) x 137(V) mm y de la tabla 5.1 se conoce el tamaño de la imagen que es de 640(H) x 480(V) píxeles, para finalmente con estos datos obtener la relación de escala, que es de 0.2859(H) x 0.2854(V) mm/píxel. La relación de escala es importante

para transformar los resultados obtenidos del procesamiento de imágenes dados en píxeles a milímetros, unidades de dimensión que se manejan para realizar los cálculos para el robot manipulador.

Una vez que se tiene claro el concepto de adquisición de imagen y se conocen algunos parámetros involucrados, se obtiene la imagen o imágenes necesarias para realizar el procedimiento y todas las pruebas necesarias. En seguida se presenta la binarización de la imagen, paso posterior a la adquisición.

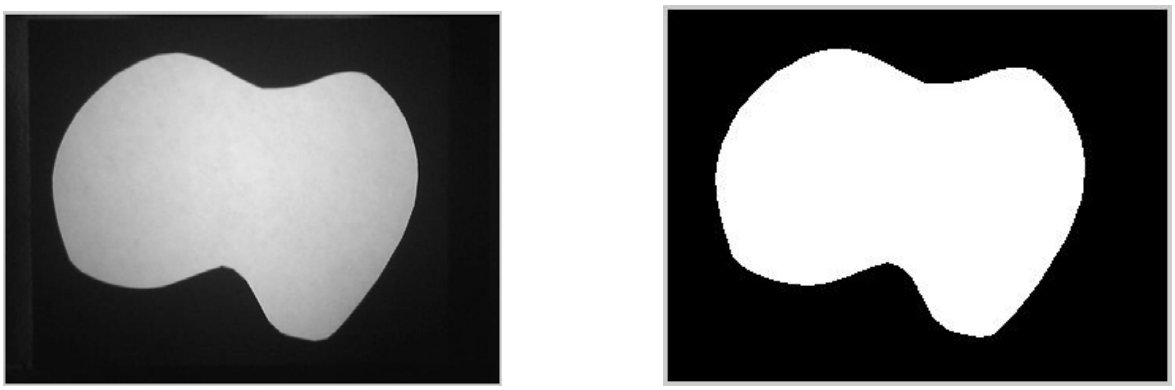
5.2 BINARIZACIÓN DE LA IMAGEN

En la sección 2.4 se mencionó el proceso de binarización, que se realiza mediante la detección del umbral óptimo, el cuál se selecciona por el método de Otsu [5]. Por esta razón en esta sección no se enfatiza en el tema, sólo se describe el procedimiento que se utilizó para binarizar la imagen, y se muestran las funciones implementadas.

Primero, recordando algunas de las características de interés de la imagen adquirida; está en escala de grises de 16 bits y tiene una dimensión de 640 x 480, características que se deben tomar en cuenta para realizar la binarización, la cual se realiza en Matlab utilizando las siguientes funciones:

```
imagenA = imread('figura.png');          //Lee un archivo tipo imagen.
umbral = graythresh(imagenA);            //Obtiene el umbral óptimo del histograma.
imagenB = im2bw(imagenA,umbral);         //Binariza la imagen en función del umbral.
```

Donde la función *graythresh* entrega el umbral óptimo de la imagen original utilizando el método de Otsu, en una escala de 0 a 1, representando 0 el 0% y 1 el 100%. El resultado de evaluar estas funciones se muestra en las figuras 5.8, donde en la figura (a) se tiene la imagen original y en la figura (b) el resultado de la binarización.



(a) *(b)*

Figura 5.8. (a) Imagen real, (b) Imagen binaria.

Un sistema de representación de distribución de intensidades en la imagen es mediante un histograma, en la figura 5.9 se representa el histograma de la imagen de la figura 5.8(a), donde el umbral óptimo obtenido con el método de Otsu es de 0.5765 que corresponde a un valor de 37780 en escala de 16 bits, lo que significa que los valores que estén por encima de este valor se ajustan a 1 y los que estén por debajo se ajustan a 0, obteniendo finalmente una imagen binaria representado por 0's y 1's, figura 5.8(b).

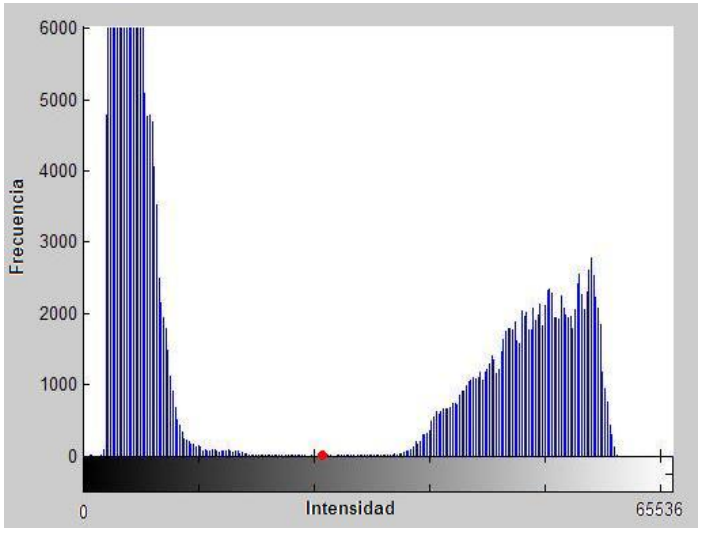


Figura 5.9. Histograma de la figura 5.8(a).

En el CD anexo a la tesis se muestran el diagrama de flujo y la función ***binarizar*** en Matlab.

Una vez que se tiene la imagen binaria, el siguiente paso es representar su contorno en el plano de la imagen. A continuación se describe el procedimiento para obtener la representación de dicho contorno.

5.3 DETECCIÓN DEL CONTORNO

Una vez que se tiene la imagen binaria se procede a determinar su contorno en un par de vectores *(x,y)* que representen la posición del contorno en el plano de la imagen. Para obtener esta representación se utiliza una variante del algoritmo del código de la cadena [6], que se describió en la sección 2.5. Las variantes del algoritmo son tres principalmente: El criterio de seguimiento, el criterio de paro y finalmente la forma de obtener los resultados. El algoritmo propio se describe a continuación:

Primero se busca un punto inicial, que consiste en un cambio de intensidad dentro de la imagen binaria, la búsqueda se hace realizando un barrido vertical a lo largo de la imagen, como se ilustra en la figura 5.10. Este punto es sólo de referencia y no necesariamente es el punto de inicio de la trayectoria. El punto de inicio de la trayectoria se puede seleccionar realizando un corrimiento de los vectores que se obtienen del contorno al punto que más convenga.

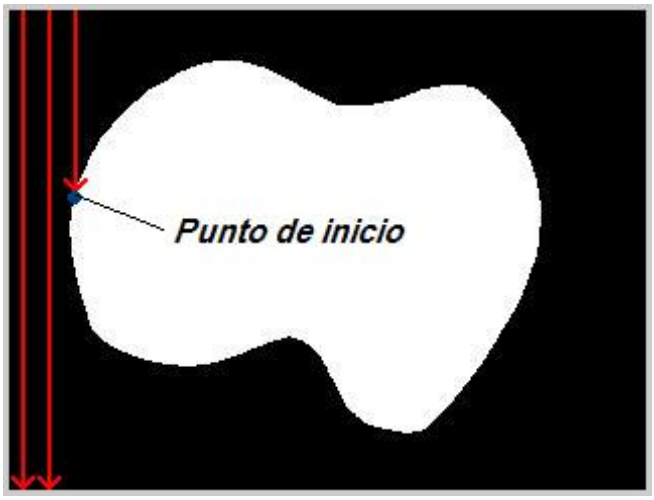
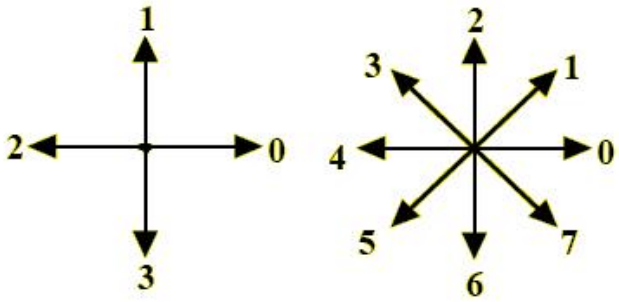


Figura 5.10. Búsqueda del punto de inicio del contorno.

Una vez que se tiene el punto de inicio del contorno, se guarda como (x_0,y_0), el cual sirve como referencia para decidir que el seguimiento del contorno se completó. El seguimiento del contorno se realiza considerando una vecindad de píxeles de 8, figura 5.11(b).



(a) *(b)*

Figura 5.11. (a) Vecindad 4, (b) Vecindad 8.

Posteriormente se pasa a un punto interesante, el criterio de seguimiento. El cual cuenta con algunas consideraciones; la obtención del píxel que determina la posición de inicio y paro, se tienen 4 posibles direcciones de búsqueda (figura 5.11(a)), 8 posibles píxeles donde se puede encontrar el cambio de intensidad (figura 5.11(b)) y dos posibles sentidos de búsqueda (horario y antihorario). Con estas consideraciones se puede determinar el contorno cerrado de alguna imagen binaria.

Por ejemplo, para la figura 5.12 el punto de inicio se encuentra detectando un cambio de intensidad de 0 a 1 en la dirección 3 de la figura 5.11(a), considerando el sentido de búsqueda en sentido horario, se comienza la búsqueda en dirección contraria a la que se detecto el cambio de nivel de intensidad de 0 a 1, dirección 1 representada en la figura 5.11(a), correspondiente al punto 2 de la figura 5.11(b), si el nivel de intensidad sigue siendo 1 se guarda la posición *(x,y)* y se continúa la búsqueda, de lo contrario se sigue la búsqueda en sentido horario al punto 1 representado en la figura 5.11(b), si se encuentra el nivel de intensidad 1 se guarda la posición *(x,y)* y se continua la búsqueda, pero ahora partiendo de la dirección contraria a la que se encontró el nivel de intensidad 1 anterior, en este caso en la dirección 2 representada en la figura 5.11(a) y así sucesivamente hasta terminar en el punto de inicio.

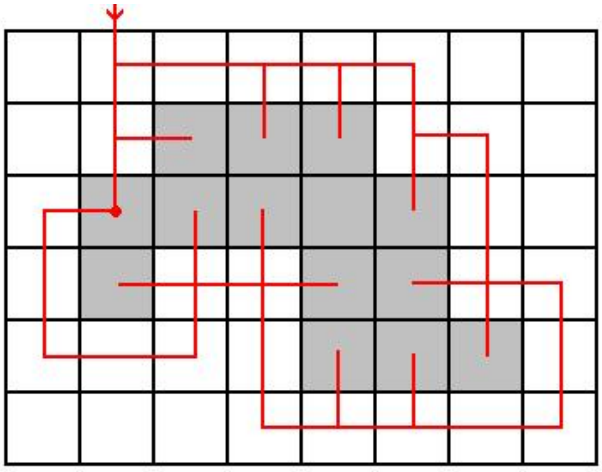


Figura 5.12. Ejemplo de seguimiento de contorno.

El criterio de paro se utiliza cuando se ha hecho la búsqueda en las 8 posibles localidades de píxeles y no se haya encontrado el nivel de intensidad adecuado, esto quiere decir que hay al menos un píxel que representa ruido para el contorno. Si es éste el caso, lo ideal es buscar el punto donde se detuvo el seguimiento y ubicarlo en la imagen binaria para corregirlo digitalmente, otra opción es adquirir nuevamente la imagen procurando mejorar su calidad. En la figura 5.13 se ilustra el contorno correspondiente a la imagen binaria de la figura 5.8(b).

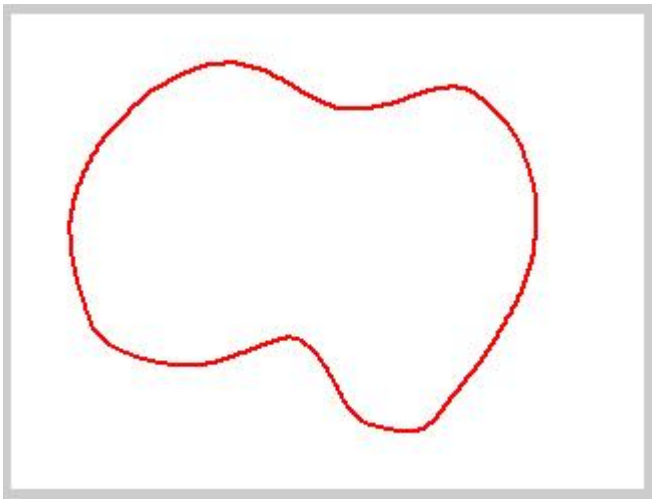


Figura 5.13. Contorno de la imagen de la figura 5.8(b).

En el CD anexo se muestran el diagrama de flujo y la función ***contorno*** en Matlab que genera el contorno del objeto (o figura).

Una vez que se tiene el contorno representado en los vectores *(x,y)*, es conveniente discriminar algunos puntos ya que si el contorno es considerablemente grande, se pueden obtener algunos miles de puntos para representarlo, situación no deseada para el procesamiento y para la generación del programa del robot manipulador.

Otra consideración para discriminar puntos, puede ser el suavizado del contorno de la imagen, ya que muchas veces no es necesario tener en cuenta todos los puntos para representar alguna trayectoria o contorno. Puede bastar con algunos puntos estratégicos y la aplicación de algún tipo de interpolación, para finalmente sólo tener un punto de inicio, uno de paro y una función que represente dicho contorno.

Para este caso se discriminan los puntos por ambas consideraciones y el criterio de discriminación es por índice, por ejemplo, si la longitud inicial de los vectores es de 1000 puntos y sólo se quieren considerar 10 puntos del contorno, el índice de selección debe ser de 100, que corresponde a la división del numero de puntos totales entre los puntos que se consideran, y representa que cada 100 puntos se considera sólo uno.

El criterio de índice se utiliza por simplicidad, sin embargo, podría no ser el adecuado ya que el contorno puede presentar regiones suaves en un intervalo y otras irregulares en otro. Lo ideal seria considerar puntos estratégicos, dependiendo de lo complicado que sea el contorno en cada región de éste. La selección de puntos estratégicos se puede hacer por ejemplo, por medio de una interfaz gráfica donde el usuario seleccione los puntos que crea necesarios o considerando algún criterio automático donde se consideren los cambios drásticos en la trayectoria del contorno (un criterio de derivada).

Siguiendo con el procedimiento, en la figura 5.14 se muestra un ejemplo de la discriminación de puntos por el criterio del índice. La discriminación se basa en la imagen que representa al contorno (figura 5.13), donde el contorno está representado por 1284 puntos, de los cuales se consideran finalmente 30 puntos.

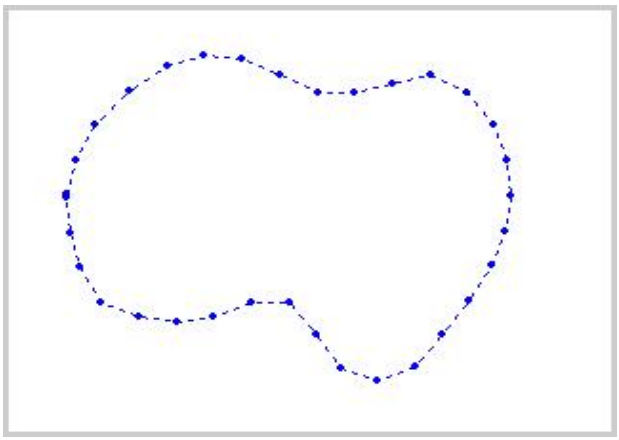


Figura 5.14. Ejemplo de discriminación de puntos.

En el CD anexo a la tesis se presentan el diagrama de flujo y la función ***discriminar*** en Matlab que realiza la discriminación de puntos.

Una vez que se tienen los puntos considerados, de igual forma se almacenan en un par de vectores *(x,y)* y se procede a realizar las interpolaciones consideradas para el proyecto. Estas interpolaciones son; la interpolación por splines, que se considera como la más adecuada para esta aplicación, debido a sus ventajas sobre otro tipo de interpolaciones, además de que es la más utilizada para realizar la interpolación para diferentes robots industriales. Sin embargo, dado que el robot con el que se cuenta sólo puede realizar interpolaciones circulares, fue necesario implementar también el tipo de interpolación circular. En los siguientes temas se tratan estas interpolaciones.

5.4 INTERPOLACIÓN POR SPLINES

Continuando con el procedimiento para la generación de la trayectoria, el siguiente paso es interpolar los puntos obtenidos de la discriminación, utilizando curvas paramétricas. Primero se presenta el desarrollo utilizando splines y posteriormente utilizando interpolación circular.

En los apartados 3.2 y 3.3 se presentaron los algoritmos para el manejo de splines y curvas paramétricas, estos algoritmos son los mismos que aquí se utilizan para la generación de la

trayectoria. Considerando a los puntos obtenidos de la discriminación como los puntos a interpolar y considerando que se interpola en curvas paramétricas, se obtendrán tantos polinomios cúbicos como puntos considerados para cada curva paramétrica.

Por ejemplo, si se consideran sólo 30 puntos del contorno, es necesario obtener 60 polinomios que representen el contorno, 30 para la curva paramétrica de las x en función de t y 30 para la curva paramétrica de las y en función de la misma t. Donde t es el parámetro que está determinado en un intervalo $[t_0, t_n]$ con $t_0 < t_1 < \ldots < t_n$. En las fórmulas 5.7 y 5.8 se representan estos polinomios.

$$P_{xi}(t) = d_i t^3 + c_i t^2 + b_i t + a_i \tag{5.7}$$

$$P_{yi}(t) = h_i t^3 + g_i t^2 + f_i t + e_i \tag{5.8}$$

donde el algoritmo de interpolación por splines entrega como resultado los valores de los coeficientes (a, b, c y d para $P_{xi}(t)$ y e, f, g y h para $P_{yi}(t)$) para cada uno de los puntos. Evaluando los polinomios independientes en los intervalos $t_0 \leq t < t_1$, $t_1 \leq t < t_2$ ... $t_{n-1} \leq t < t_n$, se obtienen las curvas paramétricas $x(t)$ e $y(t)$. En la figura 5.15 se representan las curvas paramétricas correspondientes a la figura 5.8(b), considerando 30 puntos, donde en el eje de las abscisas se representan los puntos considerados en el intervalo t y en el eje de las ordenadas la posición en píxeles para cada eje.

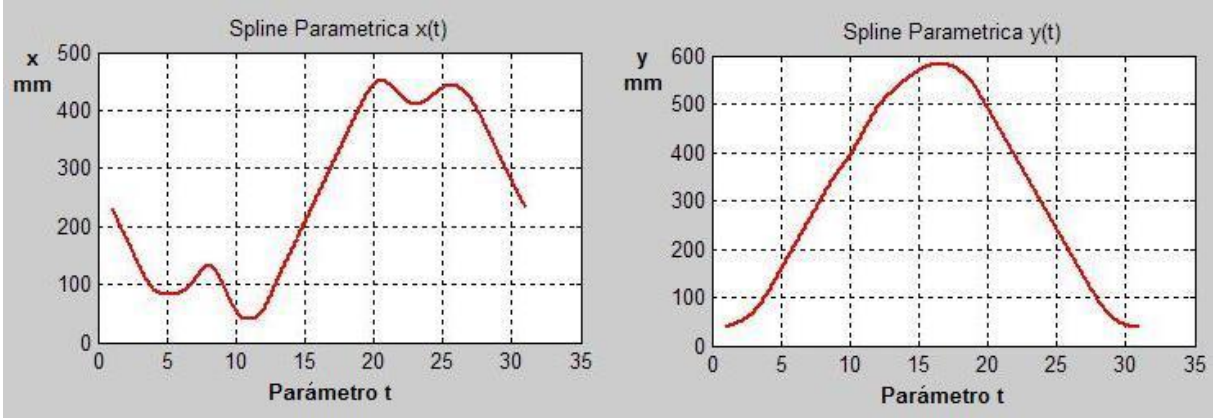


Figura 5.15. Curvas paramétricas.

En la figura 5.16 se grafica la interpolación de las dos curvas paramétricas, teniendo como resultado una trayectoria muy aproximada al contorno inicial. Y en la figura 5.17 se muestra la comparación de las splines contra el contorno real, así como los puntos que se consideraron y algunas ampliaciones en secciones consideradas como complicadas.

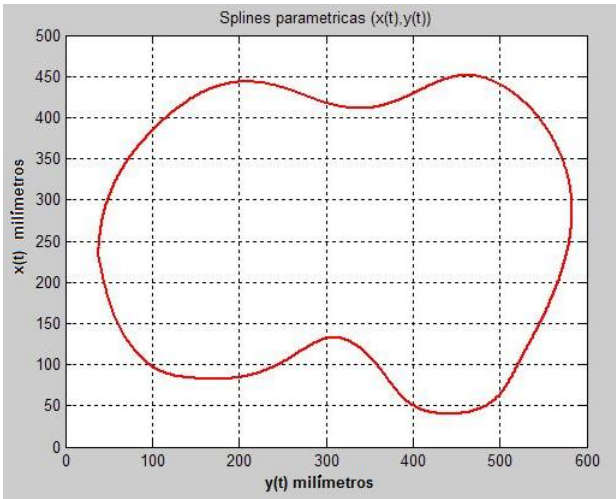


Figura 5.16. Splines paramétricas (x(t),y(t)).

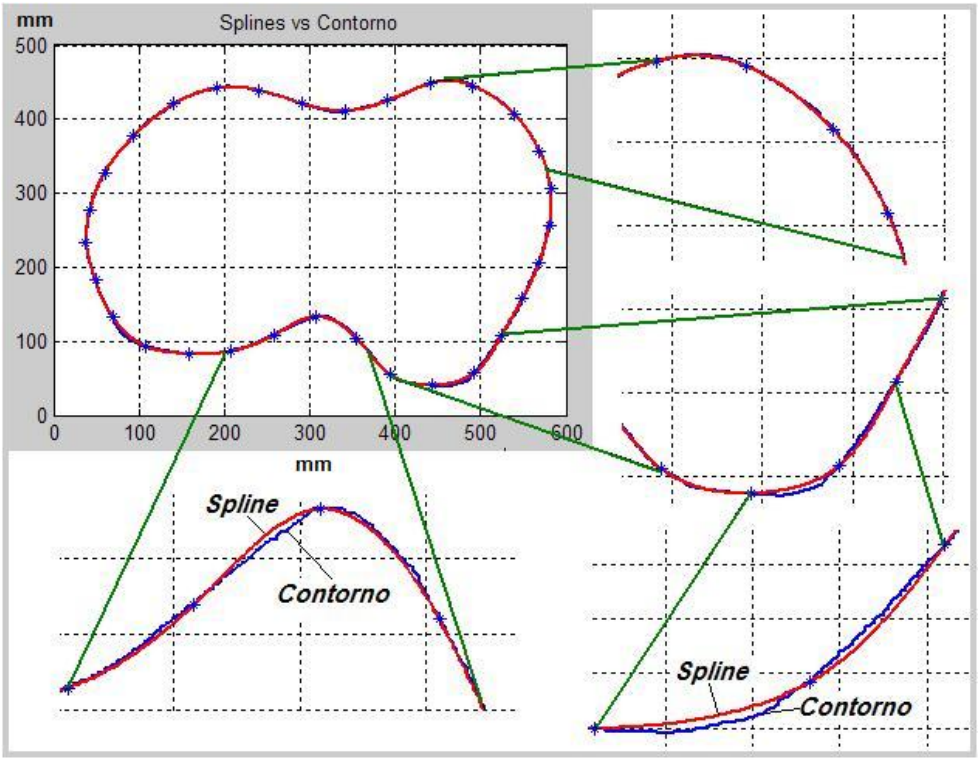


Figura 5.17. Spline vs Contorno y ampliaciones.

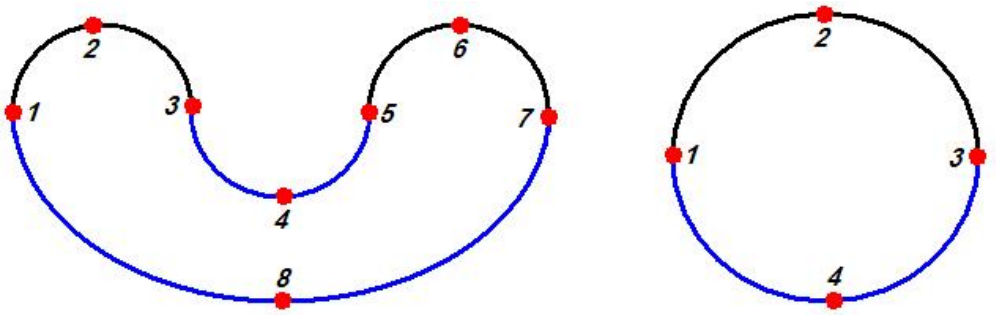
De la figura 5.17, se observa cómo la interpolación por splines se aproxima considerablemente al contorno. En la sección de pruebas y resultados, se hacen comparaciones con diferentes formas de contorno y se parametriza el error de aproximación.

En el CD anexo se muestran el diagrama de flujo y la función ***spline_int*** en Matlab que realiza la interpolación paramétrica por splines.

5.5 INTERPOLACIÓN CIRCULAR

En sección 3.1 se mostró el algoritmo para construir un círculo a partir de tres puntos, el mismo algoritmo se utiliza para construir la trayectoria del contorno, tomando en cuenta el numero de puntos considerados para determinar el numero de fragmentos de círculos que son necesarios para representar al contorno completo.

Como cada círculo requiere tres puntos y los círculos están seguidos, uno tras otro, el punto final de cada círculo es el punto inicial para el siguiente y así sucesivamente hasta el último círculo, cuyo punto final es el punto inicial del primero, para así cerrar la trayectoria, ver figura 5.18 (a) y (b).



(a) *(b)*

Figura 5.18. Ejemplo de interpolación circular.

Tomando en cuenta estas consideraciones se deduce que el numero de círculos necesarios para generar la trayectoria del contorno, es igual al numero de puntos considerados dividido

entre dos. Esto implica un par de restricciones; el número de puntos considerados siempre debe ser múltiplo de dos y el número de puntos considerado debe ser mayor o igual a cuatro.

En la misma figura 5.18 se representan un par de ejemplos de la interpolación circular, donde se ilustran las consideraciones mencionadas. Donde para la figura 5.18(a) se tienen 8 puntos, los cuales corresponden a 4 fragmentos de círculos y para la figura 5.18(b) se tienen 4 puntos, los cuales representan 2 fragmentos de círculos.

Para realizar la interpolación circular es necesario representar las ecuaciones paramétricas asociadas al círculo, la forma de hacerlo es manejar la posición x e y en función del ángulo θ como se ilustra en las fórmulas 5.9 y 5.10,

$$x(\theta) = r\cos\theta \tag{5.9}$$

$$y(\theta) = r\sin\theta \tag{5.10}$$

donde r es el radio del círculo, obtenido del algoritmo de la sección 3.1. Al igual que para las splines, para representar el contorno en fragmentos de círculos hay que evaluar cada fragmento desde su punto inicial hasta su punto final en función del ángulo θ, como se muestra en los siguientes intervalos, $\theta_0 \leq \theta < \theta_1$, $\theta_1 \leq \theta < \theta_2$, . . . $\theta_{n-1} \leq \theta \leq \theta_n$, donde θ_i es,

$$\theta_i = \arctan\frac{y_i}{x_i} \tag{5.11}$$

y x_i e y_i son los puntos considerados, que se obtuvieron de la discriminación. Cabe señalar que para calcular θ_i se utilizó la función *atan2* de Matlab, que obtiene el valor del ángulo aplicando la tangente inversa, que además considera sólo el cuarto cuadrante y entrega los resultados en el intervalo *-pi <= atan2(y,x) <= pi*.

Una vez realizada la interpolación y evaluando las ecuaciones en todo el intervalo para incrementos pequeños de θ, se obtiene el resultado de la interpolación circular paramétrica.

Como ejemplo se considera la misma figura que se ha venido manejando (figura 5.8(b)); en la figura 5.19 se muestra el resultado de la interpolación circular, de igual forma utilizando 30 puntos y finalmente en la figura 5.20 se muestra la comparación de la interpolación circular con el contorno y algunas ampliaciones.

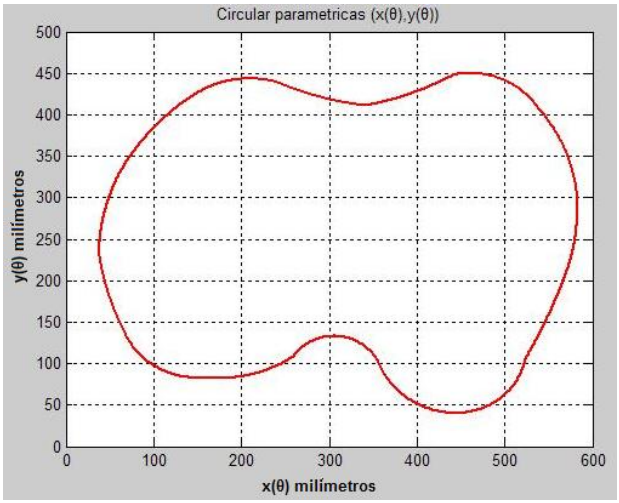


Figura 5.19. Interpolación circular.

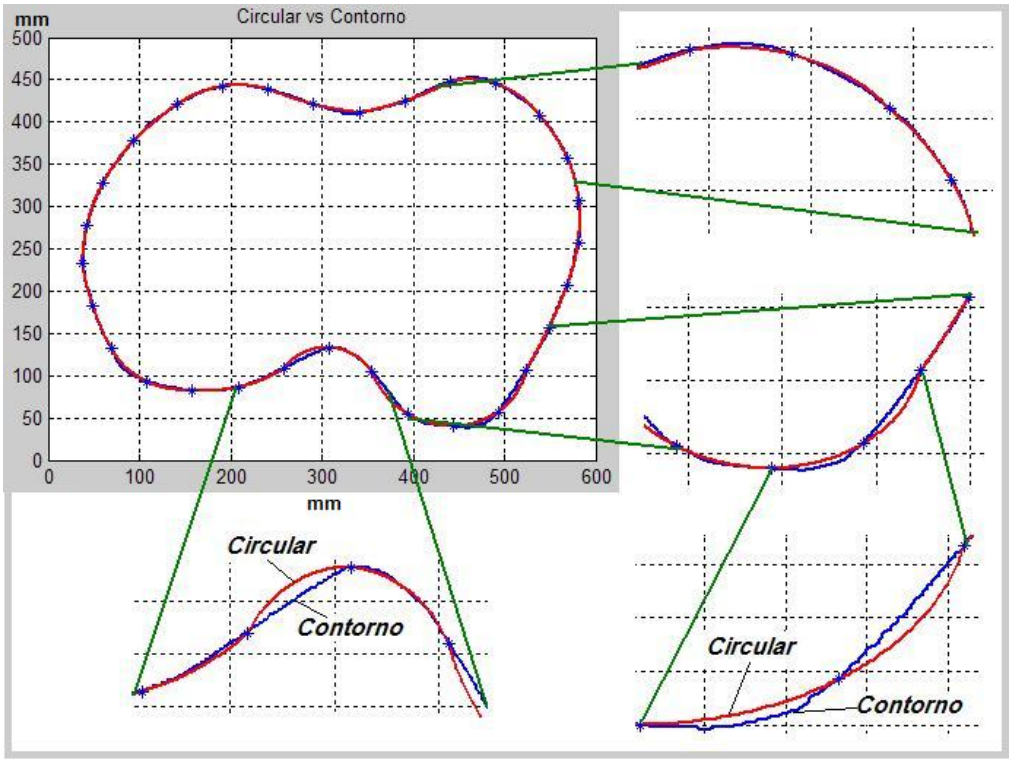


Figura 5.20. Circular vs Contorno y amplificaciones.

En la sección de pruebas y resultados, se harán comparaciones con diferentes formas, donde se parametrizará el error de aproximación de la interpolación circular.

En el CD anexo se muestran el diagrama de flujo y la función ***circular_int*** en Matlab que realiza la interpolación circular en curvas paramétricas.

Haciendo una recapitulación del procedimiento desarrollado hasta aquí; se definió la adquisición de la imagen, se binarizó utilizando el método de Otsu , posteriormente se obtuvo el contorno del objeto (o figura) en un par de vectores, se discriminaron algunos puntos de los vectores y finalmente estos puntos se interpolaron por dos técnicas, splines y circular.

Una vez que se tienen los resultados del procesamiento de imágenes y de las interpolaciones, el siguiente punto es ligar estos resultados con el robot manipulador. El punto donde se ligan los datos es precisamente en la generación de la trayectoria del robot manipulador, sin embargo, para poder llegar a este punto primero es necesario desarrollar la cinemática asociada al manipulador, por lo que el siguiente capítulo se enfoca al desarrollo de la cinemática asociada al robot manipulador Melfa RV2-AJ de Mitsubishi.

CAPÍTULO 6

CINEMÁTICA Y GENERACIÓN DE TRAYECTORIA

En este capítulo se desarrolla la cinemática asociada al robot manipulador Melfa RV-2AJ de Mitsubishi, que es el robot con el que se cuenta para el proyecto. Se generará la trayectoria a partir de los resultados de las interpolaciones y se hará notar la importancia de la modelación y la simulación para comprobar el comportamiento del sistema así como los cálculos desarrollados.

6.1 CARACTERÍSTICAS DEL ROBOT MANIPULADOR

Antes de comenzar con el procedimiento de la generación de trayectoria, es necesario conocer las características del robot manipulador que se va a utilizar, así como su cinemática asociada. Esta sección se considera como un punto intermedio antes de generar la trayectoria, ya que es aquí donde se resuelve el problema cinemático asociado al manipulador para obtener el modelo que se utilizará para generar la trayectoria y finalmente desarrollar la simulación en base a los resultados de la interpolación. En la figura 6.1 se muestra una foto del robot manipulador.

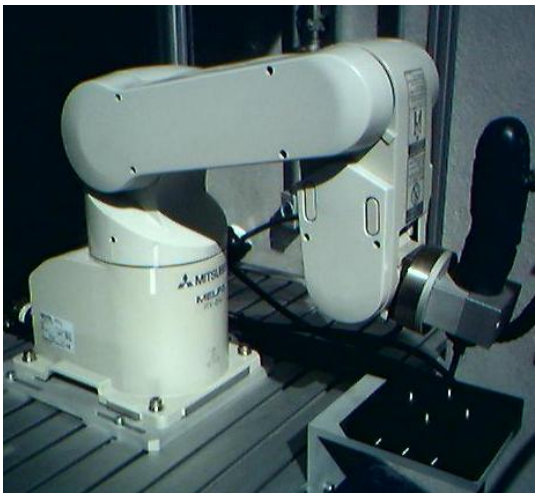


Figura 6.1. Robot Manipulador.

El robot RV-2AJ de Mitsubishi cuenta con cinco grados de libertad, los cinco son angulares y cae dentro de los robots clasificados como articulados o antropomórficos. En la figura 6.2 se representan los grados de libertad y sus restricciones de movimiento.

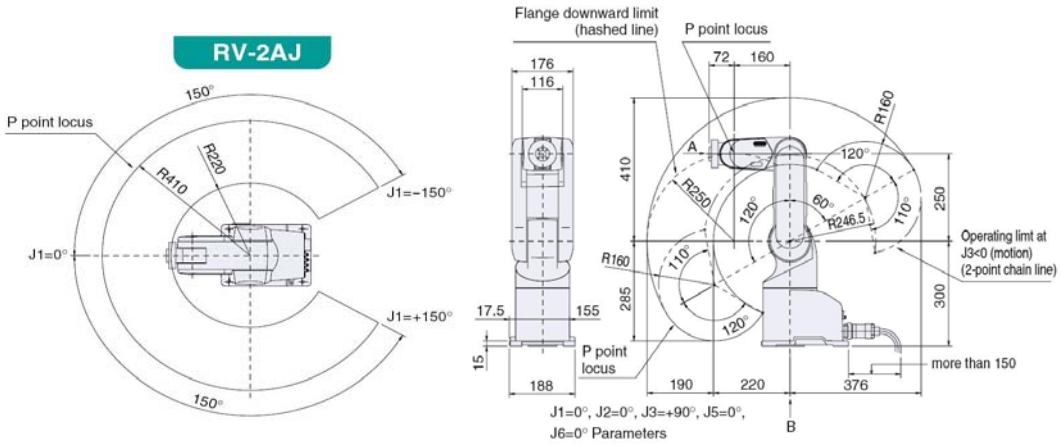


Figura 6.2. Restricciones físicas y configuración del Robot.

En la tabla 6.1, se describen las características principales del robot manipulador, tal como se muestran en las hojas de datos.

Tabla 6.1. Características principales del Robot.

Grados de Libertad			5
Tipo de Motores			Servomotores de C.A.
Sensor de Posición			Encoder Absoluto
Capacidad de Carga			2 kg
Limites de operación	J1	Grad.	+/-150
	J2		180 (-60 ~ +120)
	J3		230 (-110 ~ +120)
	J4		-
	J5		+/-90
	J6		+/-200
Velocidades máximas	J1	Grad./s	180
	J2		90
	J3		135
	J4		-
	J5		180
	J6		210
Cuasi-Velocidad mm/s			Aprox. 2100
Resolución mm			+/-0.02

Otra consideración que hay que tener en cuenta son las aceleraciones máximas. El manual de programación del robot se refiere a que cada articulación alcanza su velocidad máxima en 0.2 segundos, por lo que si se dividen las velocidades máximas entre éste tiempo se obtienen las aceleraciones máximas del robot.

En la tabla 6.2 se muestran las aceleraciones máximas, considerando las velocidades máximas y el tiempo mínimo de aceleración de 0.2 segundos ya mencionados.

Tabla 6.2. Aceleraciones Máximas.

Articulación	J1	J2	J3	J4	J5	J6
Aceleración Máxima Grad./s^2	900	450	675	-	900	1050

Ya conocidas la principales características del manipulador se procede a desarrollar los cálculos de cinemática directa e inversa.

6.2 CINEMÁTICA DIRECTA

Para realizar los cálculos de la cinemática es necesario conocer la configuración del robot manipulador, la cual ya se describió, así como tener muy claro algunos aspectos matemáticos, como el manejo de matrices y trigonometría.

Los cálculos de la cinemática directa se desarrollan con ayuda de la metodología de Denavit – Hartenberg [8] que se presentó en la sección 4.2, comprobando las ecuaciones obtenidas en Matlab. En la figura 6.4 se muestra un bosquejo del robot y su representación en un diagrama de cuerpo libre. Para dibujar este diagrama de cuerpo libre, se considera el algoritmo de Denavit – Hartenberg:

D-H 1.- Numerar los eslabones comenzando con *1* (primer eslabón móvil de la cadena) y acabando con *n* (último eslabón móvil). Se numerará como eslabón *0* a la base fija del robot.

D-H 2.- Numerar cada articulación comenzando por *1* (la correspondiente al primer grado de libertad) y acabando en *n*.

D-H 3.- Localizar el eje de cada articulación. Si ésta es rotativa, el eje será su propio eje de giro. Si es prismática, será el eje a lo largo del cual se produce el desplazamiento.

D-H 4.- Para *i* de *0* a *n-1* situar el eje z_i sobre el eje de la articulación *i+1*.

D-H 5.- Situar el origen del sistema de la base $\{S_0\}$ en cualquier punto del eje z_0. Los ejes x_0 e y_0 se situarán de modo que formen un sistema dextrógiro con z_0, figura 6.3

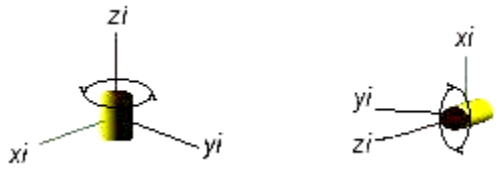


Figura 6.3. Ejes y giros de articulaciones.

D-H 6.- Para *i* de *1* a *n-1*, situar el sistema $\{S_i\}$ (solidario al eslabón *i*) en la intersección del eje z_i con la línea normal común a z_{i-1} y z_i. Si ambos ejes se cortasen se situaría $\{S_i\}$ en el punto de corte. Si fuesen paralelos $\{S_i\}$ se situaría en la articulación *i+1*.

D-H 7.- Situar x_i en la línea normal común a z_{i-1} y z_i.

D-H 8.- Situar y_i de modo que forme un sistema dextrógiro con x_i y z_i.

D-H 9.- Situar el sistema $\{S_n\}$ en el extremo del robot de modo que z_n coincida con la dirección de z_{n-1} y x_n sea normal a z_{n-1} y z_n.

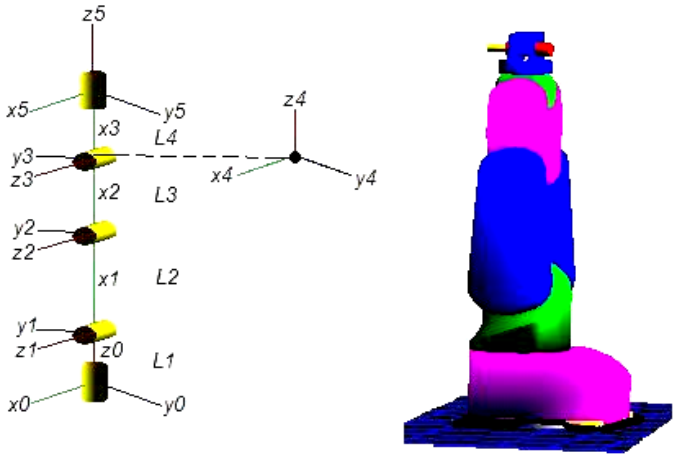


Figura 6.4. Diagrama de cuerpo libre.

D-H 10.- Obtener $\boldsymbol{\theta_i}$ como el ángulo que hay que girar en torno a z_{i-1} para que x_{i-1} y x_i queden paralelos.

D-H 11.- Obtener $\mathbf{d_i}$ como la distancia, medida a lo largo de z_{i-1}, que habría que desplazar $\{S_{i-1}\}$ para que x_i y x_{i-1} quedasen alineados.

D-H 12.- Obtener $\mathbf{a_i}$ como la distancia medida a lo largo de x_i (que ahora coincidiría con x_{i-1}) que habría que desplazar el nuevo $\{S_{i-1}\}$ para que su origen coincidiese con $\{S_i\}$.

D-H 13.- Obtener $\boldsymbol{\alpha_i}$ como el ángulo que habría que girar entorno a x_i (que ahora coincidiría con x_{i-1}), para que el nuevo $\{S_{i-1}\}$ coincidiese totalmente con $\{S_i\}$.

Tabla 6.3. Parámetros de Denavit-Hartenberg.

Articulación	θ	d	a	α
1	***q1***	*L1*	***0***	***π/2***
2	***q2***	***0***	***L2***	***0***
3	***q3***	***0***	***L3***	***0***
4	***q4***	***0***	***0***	***π/2***
5	***q5***	***L4***	***0***	***0***
6	***0***	***0***	***L5***	***π/2***

En esta tabla se consideró una articulación extra, la numero 6, la cual en realidad no representa una articulación como tal, ya que carece de movimiento, sino que representa una traslación, la cual corresponde a la longitud de la herramienta.

D-H 14.- Obtener las matrices de transformación ${}^{i-1}A_i$. Como lo muestra la fórmula 6.1:

$$ {}^{i-1}A_i = \begin{bmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i S\theta_i & a_i C\theta_i \\ S\theta_i & C\alpha_i C\theta_i & -S\alpha_i C\theta_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.1} $$

donde C y S representan el *coseno* y el *seno* del ángulo respectivamente, α y θ representan ángulos, a y d representan longitudes e i representa el índice. Con base a la fórmula 6.1 y a la tabla 6.3, se desarrollan las matrices de transformación para cada articulación.

$$ {}^{0}A_1 = \begin{bmatrix} Cq_1 & 0 & Sq_1 & 0 \\ Sq_1 & 0 & -Cq_1 & 0 \\ 0 & 1 & 0 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^{1}A_2 = \begin{bmatrix} Cq_2 & -Sq_2 & 0 & L_2Cq_2 \\ Sq_2 & Cq_2 & 0 & L_2Sq_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.2, 6.3) $$

$$ {}^{2}A_3 = \begin{bmatrix} Cq_3 & -Sq_3 & 0 & L_3Cq_3 \\ Sq_3 & Cq_3 & 0 & L_3Sq_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^{3}A_4 = \begin{bmatrix} Cq_4 & 0 & Sq_4 & 0 \\ Sq_4 & 0 & -Cq_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.4, 6.5) $$

$$ {}^{4}A_5 = \begin{bmatrix} Cq_5 & -Sq_5 & 0 & 0 \\ Sq_5 & Cq_5 & 0 & 0 \\ 0 & 0 & 1 & L_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^{5}A_6 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & L_5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.6, 6.7) $$

Donde *qi* es el ángulo correspondiente a la articulación *i*.

L1: Longitud del primer eslabón. Distancia del eje de coordenadas *0* al *1*. *L1=300mm.*

L2: Longitud del segundo eslabón. Distancia del eje de coordenadas *1* al *2*. *L2=250mm.*

L3: Longitud del tercer eslabón. Distancia del eje de coordenadas *2* al *3*. *L3=160mm.*

L4: Longitud del cuarto eslabón. Distancia del eje de coordenadas *3* al *4*. *L4=102mm.*

L5: Longitud de la herramienta. Distancia del eje de coordenadas *4* al *5*. *L5=60mm.*

Nota: *L4* es la suma del eslabón *4*, figura 6.4, más desplazamiento de la herramienta *(72+30)*. Y *L5* es la longitud de la herramienta.

DH 15.- Obtener la matriz de transformación entre la base y el extremo del robot utilizando la fórmula 6.8.

$$ T = {}^{0}A_1 \, {}^{1}A_2 \ldots {}^{n-1}A_n \quad (6.8) $$

$$ T = {}^{0}A_6 = {}^{0}A_1 \, {}^{1}A_2 \, {}^{2}A_3 \, {}^{3}A_4 \, {}^{4}A_5 \, {}^{5}A_6 \quad (6.8a) $$

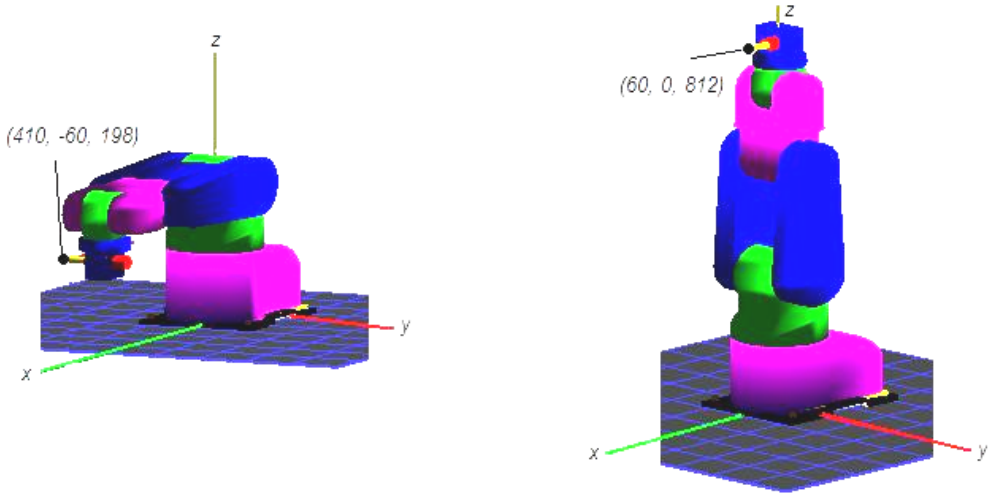
DH 16.- La matriz *T* define la orientación (submatriz de rotación) y posición (submatriz de traslación) del extremo referido a la base en función de las *n* coordenadas articulares.

Una vez que se obtiene la matriz de transformación *T*, se comprueba evaluando para las *qi* para valores reales, considerando las longitudes *Li* del robot manipulador. En la tabla 6.4, se representan los resultados que se obtienen de evaluar las matrices de transformación contra los resultados que en realidad se deben obtener.

Tabla 6.4. Evaluación de matrices de transformación (a) vs Resultados correctos (b).

Entrada (ángulos en radianes)					Salida (posición mm)			
q1	***q2***	***q3***	***q4***	***q5***	***x***	***y***	***z***	***Fig.***
0	0	0	0	0	410	-60	198	a
0	0	0	0	0	60	0	812	b

En la figura 6.5(a) se muestra el resultado de la posición del robot evaluando para estos ángulos de entrada y en la figura 6.5(b) se representa la posición que debería ser para los mismos ángulos.



(a) *(b)*

Figura 6.5. (a) Posición de evaluación, (b) Posición real correspondiente.

De la tabla 6.4 y las figura 6.5(a) y (b), se puede observar que no corresponden la posición real que debe tener el robot con la posición que se obtiene de evaluar la matriz de transformación *T* para una entrada dada. Este es un pequeño error de planteamiento, que se corrige fácilmente utilizando un ajuste (offset) de los ángulos de entrada. Como se muestra a continuación:

$$q1 = q1, \quad q2 = \pi/2 - q2, \quad q3 = -q3, \quad q4 = \pi/2 - q4, \quad q5 = q5 + \pi/2$$

los cuales se establecieron después de realizar algunas corridas en Matlab, considerando entradas y salidas conocidas. Finalmente en la tabla 6.5 se muestran algunas corridas con los valores modificados y se comprueba que esta vez si corresponden la posición real con la evaluación de la matriz de transformación *T*.

Tabla 6.5. Evaluación de cinemática directa.

Entrada (ángulo en radianes)					Salida (posición mm)		
q1	***q2***	***q3***	***q4***	***q5***	***x***	***y***	***z***
0	0	0	0	0	60	0	812
π/2	0	0	0	0	0	60	812
- π/2	0	0	0	0	0	-60	812
0	π/2	0	0	0	512	0	240
0	- π/2	0	0	0	-512	0	360
0	0	π/2	0	0	262	0	490
0	0	- π/2	0	0	-262	0	610
0	0	0	π/2	0	102	0	650
0	0	0	- π/2	0	-102	0	770
0	0	0	0	π/2	0	60	812
0	0	0	0	- π/2	0	-60	812

Los cálculos y algoritmos desarrollados para la cinemática directa son de vital interés para calcular la cinemática inversa, ya que sirven para comprobar los resultados, por lo que hay que tener cuidado durante todo el proceso de obtención de los resultados de la cinemática

directa y corroborar que efectivamente correspondan a lo correcto. Es por esto que se trabajó a la par del desarrollo de la cinemática directa con un software de simulación en 3-D. La simulación numérica, al igual que los cálculos se realizan en Matlab, que fue de gran ayuda para determinar estos resultados.

La simulación en 3-D se desarrolló en C++ utilizando librerías de OpenGL, utilizando los resultados que se obtuvieron en Matlab, en el anexo 1 se da una breve introducción de lo que es OpenGL y en el CD anexo a la tesis se presentan el diagrama de flujo y la función ***cinematica_directa*** en Matlab que resuelve la cinemática directa del manipulador, y se presenta la función ***RobConHta*** que realiza la simulación en 3D en OpenGL.

Una vez que ya se tiene la cinemática directa y la simulación del robot manipulador el siguiente punto es obtener la cinemática inversa, que a continuación se describe.

6.3 CINEMÁTICA INVERSA

Existen diferentes técnicas para calcular la cinemática inversa, donde destacan como las más conocidas, las matrices de transformación, el método geométrico y el desacople cinemático. Sin embargo, el cálculo de la cinemática inversa se puede llegar a complicar, debido a puntos de singularidad del manipulador ó a múltiples soluciones del sistema cinemático inverso, debido a que son sistemas no lineales que involucran funciones periódicas.

El cálculo de la cinemática inversa se desarrolló por el método de desacople cinemático, que no es más que una variante del método geométrico. Se utiliza esta técnica y no la de matrices de transformación homogénea, debido a que con las matrices de transformación homogénea los cálculos llegan a un punto donde las matrices se vuelven poco manejables.

Para poder desarrollar los cálculos de la cinemática inversa, siempre es conveniente el manejo de diagramas de cuerpo libre. En la figura 6.6 se muestra un diagrama de cuerpo libre asociado al robot manipulador.

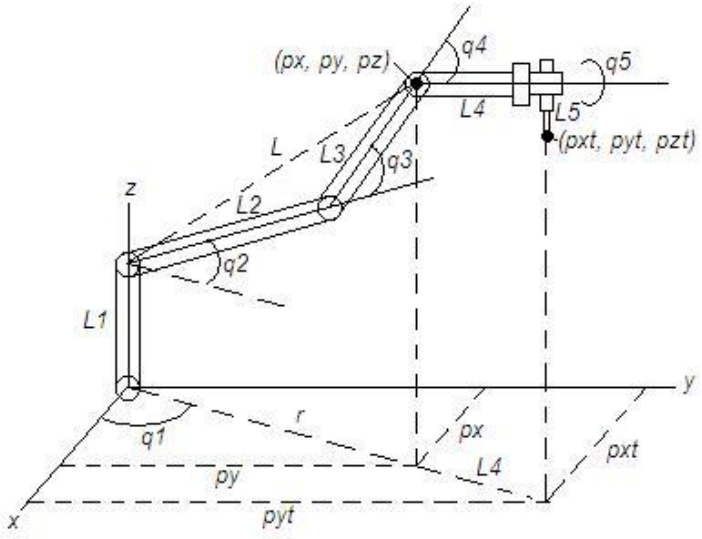


Figura 6.6. Diagrama de cuerpo libre.

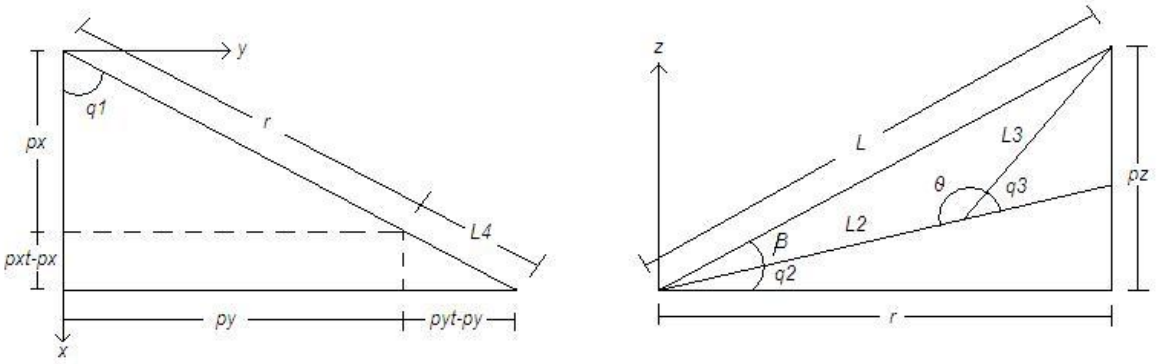
La cinemática inversa se resolvió para dos casos (a y b); en el caso (a) se considera que el robot trabaja en el plano *xy* manteniendo la orientación de la herramienta ortogonal a dicho plano, más específicamente, en dirección del eje *z* negativo. Y para el caso (b) el robot trabaja en el plano *yz,* pero esta vez la herramienta puede adoptar cualquier orientación en el plano *yz*. A continuación se desarrollan estos cálculos, primeramente para el caso (a) y posteriormente para el caso (b).

6.3.1 Cinemática Inversa caso (a)

Para poder resolver la cinemática inversa, antes que nada hay que considerar algunas restricciones correspondientes al caso (a), estas restricciones se muestran a continuación:

- El robot trabaja en el plano *xy* a una altura determinada *z*.
- La herramienta siempre está orientada en dirección del eje *z* negativo.
- El eslabón *L4* siempre es paralelo al plano *xy*.

Al considerar estas restricciones se simplifican los cálculos, ya que el ángulo *q5* es siempre constante (*q5=0*) y la suma de *q2+q3+q4=π/2*. En la figura 6.7 (a y b) se representa el diagrama de cuerpo libre por planos.



(a) *(b)*

Figura 6.7. Representación de cuerpo libre por planos (a y b).

El truco para calcular la cinemática inversa, es separar los primeros tres eslabones de los dos últimos, es por esto que se utilizan dos posiciones (*pxt, pyt, pzt*) y (*px, py, pz*). Donde (*pxt, pyt, pzt*) representa la posición final de la herramienta y (*px, py, pz*) representa la posición final de los primeros tres eslabones al igual que la posición inicial de los dos últimos eslabones. Entonces, lo único que hay que hacer es calcular la cinemática inversa primero de la posición final de la herramienta (*pxt, pyt, pzt*) a la posición final de los tres primeros eslabones (*px, py, pz*) y posteriormente de esta posición a la referencia (*0, 0, 0*). Para simplificar estos cálculos, lo que se hace es restarle a la posición (*pxt, pyt, pzt*) la posición (*px, py, pz*) quedando (*pxt-px, pyt-py, pzt-pz*).

Después de tener en cuenta estas consideraciones se desarrollan los cálculos, numerados a continuación:

1. A simple vista de la figura 6.7(a), se ve que el ángulo q_1 es el más sencillo de calcular y de hecho tiene varias opciones, pero sólo se maneja la siguiente, representada en la fórmula 6.9:

$$q_1 = \tan^{-1}\left(\frac{pyt}{pxt}\right) \tag{6.9}$$

2. Se traslada el eje de coordenadas de (*pxt, pyt, pzt*) a (*px, py, pz*), de la figura 6.7(a) se observa que,

$$L_4 \cos q_1 = pxt - px \qquad \sim \qquad px = pxt - L_4 \cos q_1 \tag{6.10}$$

$$L_4 \sin q_1 = pyt - py \qquad \sim \qquad py = pyt - L_4 \sin q_1 \tag{6.11}$$

y de la figura 6.6 se observa que,

$$pzt = L_1 + pz - L_5 \qquad \sim \qquad pz = pzt - L_1 + L_5 \tag{6.12}$$

3. Se obtiene q_3, que por simplicidad se calcula primero que q_2. de la figura 6.7(a) obtenemos el valor de *r* y de la figura 6.7(b) el valor de *L*,

$$r^2 = px^2 + py^2 \tag{6.13}$$

$$L^2 = r^2 + pz^2 \qquad \sim \qquad L^2 = px^2 + py^2 + pz^2 \tag{6.14}$$

posteriormente de la figura 6.7(b) se observa que con las longitudes *L*, *L2* y *L3* y el ángulo θ podemos utilizar la ley del coseno,

$$L^2 = L_2^2 + L_3^2 - 2L_2L_3 \cos\theta \tag{6.15}$$

considerando que,

$$\theta + q_3 = \pi \qquad \sim \qquad \theta = \pi - q_3 \tag{6.16}$$

y

$$\cos\theta = \cos(\pi - q_3) = -\cos q_3 \tag{6.17}$$

ver figura 6.7(b), sustituyendo la expresión 6.17 en 6.15 queda,

$$L^2 = L_2^2 + L_3^2 + 2L_2L_3\cos q_3 \tag{6.18}$$

y finalmente sustituyendo 6.14 en 6.18 y despejando para q_3 queda,

$$q_3 = \cos^{-1}\left(\frac{px^2 + py^2 + pz^2 - L_2^2 - L_3^2}{2L_2L_3}\right) \tag{6.19}$$

4. Para obtener q_2, que es un poco más complejo, primero de la figura 6.7(b) se hace que la suma de los ángulos β y q_2 sea igual a α,

$$\alpha = \beta + q_2, \qquad \sim \qquad q_2 = \alpha - \beta \tag{6.20}$$

de la misma figura 6.7(b) se observa que α,

$$\alpha = \tan^{-1}\left(\frac{pz}{\sqrt{px^2 + py^2}}\right) \tag{6.21}$$

utilizando la ley del seno para β,

$$\frac{\sin\theta}{L} = \frac{\sin\beta}{L_3} \tag{6.22}$$

sustituyendo el valor de L, expresión 6.14 y θ, expresión 6.16 en 6.22 y despejando para β se tiene,

$$\beta = \sin^{-1}\left(\frac{L_3\sin q_3}{\sqrt{px^2 + py^2 + pz^2}}\right) \tag{6.23}$$

finalmente sustituyendo 6.21 y 6.23 en 6.20, se tiene que,

$$q_2 = \tan^{-1}\left(\frac{pz}{\sqrt{px^2 + py^2}}\right) - \sin^{-1}\left(\frac{L_3 \sin q_3}{\sqrt{px^2 + py^2 + pz^2}}\right) \tag{6.24}$$

5. Para calcular q_4 y q_5 sólo hay que considerar las restricciones iniciales, que son, mantener la herramienta siempre en dirección del eje z negativo, teniendo que $q2+q3+q4=\pi/2$, y $q_5=0$,

$$q_4 = \frac{\pi}{2} - q_2 - q_3 \tag{6.25}$$

$$q_5 = 0 \tag{6.26}$$

A continuación se hace un resumen de las fórmulas, recordando que la entrada es la posición final de la herramienta determinada por el vector (*pxt, pyt, pzt*) y la orientación de la herramienta es siempre en dirección del eje z negativo. Y la salida es la orientación de cada articulación determinada por el vector (q_1, q_2, q_3, q_4, q_5).

$$q_1 = \tan^{-1}\left(\frac{pyt}{pxt}\right) \tag{6.27}$$
$$px = pxt - L_4 \cos q_1 \tag{6.28}$$
$$py = pyt - L_4 \sin q_1 \tag{6.29}$$
$$pz = pzt - L_1 + L_5 \tag{6.30}$$
$$q_2 = \tan^{-1}\left(\frac{pz}{\sqrt{px^2 + py^2}}\right) - \sin^{-1}\left(\frac{L_3 \sin q_3}{\sqrt{px^2 + py^2 + pz^2}}\right) \tag{6.31}$$

$$q_3 = \cos^{-1}(\frac{px^2 + py^2 + pz^2 - L_2^2 - L_3^2}{2L_2L_3})$$	(6.32)
$$q_4 = \frac{\pi}{2} - q_2 - q_3$$	(6.33)
$$q_5 = 0$$	(6.34)

Estas fórmulas representan la solución del problema de cinemática inversa tomando en cuenta las restricciones del caso (a). En la siguiente sección se desarrolla la cinemática inversa para el caso (b).

6.3.2 Cinemática Inversa caso (b)

Para desarrollar los cálculos asociados a la cinemática inversa para el caso (b), se consideran restricciones diferentes, las cuales se describen a continuación:

- El robot trabaja en el plano *yz* a una distancia determinada *x*.
- La herramienta puede adoptar cualquier orientación en el plano *yz*.
- El eslabón *L4* siempre es paralelo al plano *xy*.

En este punto del desarrollo se hace notar un caso interesante y es que al considerar estas restricciones y realizar algunas simulaciones utilizando únicamente la cinemática directa se llego a una contradicción, y es que el robot manipulador de 5 grados de libertad no puede satisfacer las restricciones iniciales. Para entender mejor a que se refiere esta contradicción se presenta el siguiente caso. Por ejemplo, si se quiere colocar la herramienta en diferentes puntos alrededor de un círculo, ubicado en el plano *yz* a una distancia determinada *x*, manteniendo la orientación de la herramienta en dirección normal al contorno del círculo, figura 6.8, se observa que esto no es posible, salvo para un par de puntos de los cuales uno se representa en la figura 6.9.

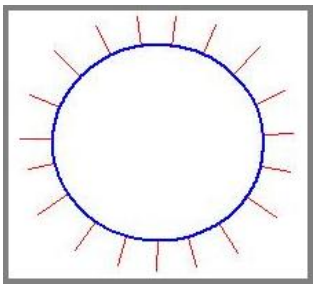


Figura 6.8. Posiciones y orientaciones de la herramienta.

En la figura 6.9 se representa una posición y orientación de la herramienta del manipulador sobre el círculo, con ayuda de la simulación, donde se ilustra que "si" es posible mantener la herramienta en la orientación y posición deseada, al menos para este punto.

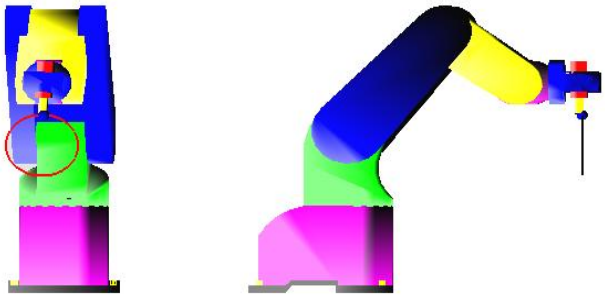


Figura 6.9. Herramienta en posición y orientación deseada.

Sin embargo, en la figura 6.10 se representa otra posición y orientación de la herramienta del manipulador sobre el mismo círculo, donde se ilustra que "no" es posible mantener la herramienta en la dirección y posición deseada.

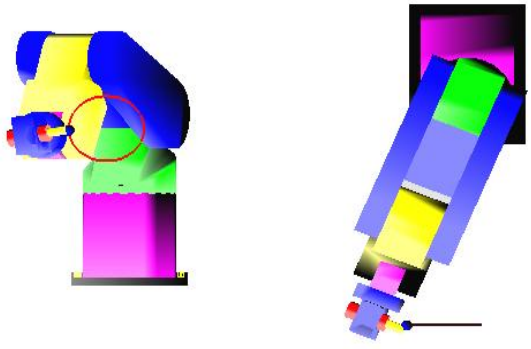


Figura 6.10. Herramienta en posición y orientación no deseada.

En general, si se analiza la orientación y posición de la herramienta para cada punto alrededor del círculo se puede entender que el robot prácticamente no puede satisfacer estas condiciones para todos los puntos, a excepción de las posiciones donde *y=0*, para este caso la posición superior e inferior del círculo (la superior se ilustra en la figura 6.9).

Este caso fue planteado inicialmente pensando que era posible (al sólo imaginarlo), sin embargo, es de buena experiencia, ya que se hace notar la importancia de la modelación y la simulación, ya que gracias a éstas se descubrió que se estaba en un error. Sin embargo, se desarrollarán los cálculos para este caso, haciendo algunas consideraciones para aproximar la posición y orientación de la herramienta del robot a las deseadas, estos cálculos se desarrollan a continuación.

Considerando las restricciones se simplifican los cálculos para la aproximación, ya que como para el caso (a) la suma de $q2+q3+q4=\pi/2$, (*L4* siempre paralelo al eje *xy*). La restricción que difiere del caso (a), es que, la orientación de *q5* es variable. Se vuelve a considerar el diagrama de cuerpo libre de la figura 6.6, para el desarrollo de los cálculos.

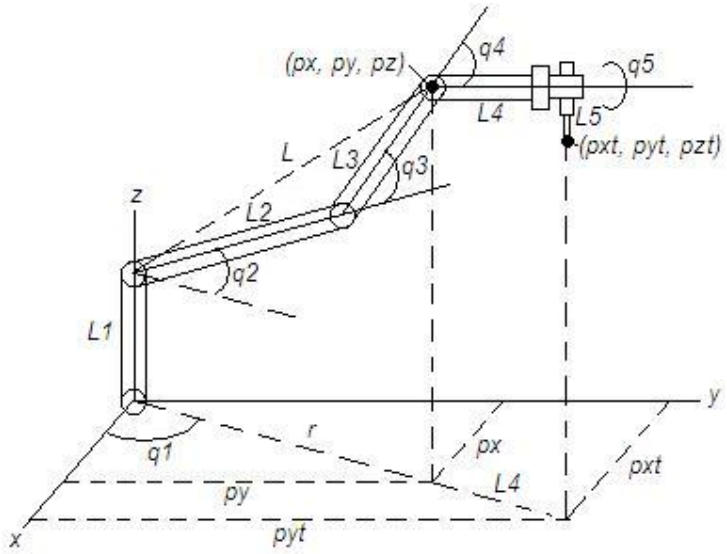
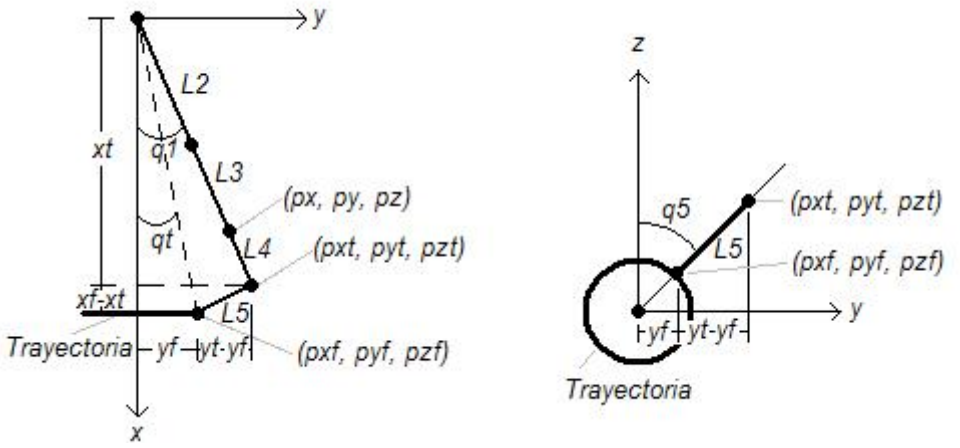


Figura 6.6. Diagrama de cuerpo libre.

Los cálculos son prácticamente los mismos, sólo se agregan un par de fórmulas a las ya conocidas del caso (a). Con base a un par de diagramas de cuerpo libre simplificados para los dos últimos grados de libertad, representados en la figura 6.11 (a) y (b), se realiza el desarrollo.



(a) *(b)*

Figura 6.11. Diagramas de cuerpo libre simplificado.

Tomando las consideraciones de que el robot trabaja en el plano *yz* pero manteniendo la posición en el eje *y* lo más cercano de su origen ($y \approx 0$) y observando la figura 6.11(a) se deduce que la distancia *xf-xt* se puede despreciar, esto se hace por simplicidad y nos conduce a la primera fórmula para el caso (b),

$$xf - xt \approx 0\,, \qquad \sim \qquad xt = xf \tag{6.35}$$

posteriormente, para calcular *yt* y *zt* se considera la figura 6.11(b) donde,

$$pyt = pyf + L_5 \sin q_5 \tag{6.36}$$

$$pzt = pzf - L_5 \cos q_5 \tag{6.37}$$

donde q_5 es el ángulo de orientación de la herramienta.

El resto de las fórmulas son las mismas que para el caso (a) a excepción de la 6.30 que se reemplaza por 6.38, éstas fórmulas se muestran nuevamente incluyendo el cambio:

$$q_1 = \tan^{-1}(\frac{pyt}{pxt})$$	(6.27)
$$px = pxt - L_4 \cos q_1$$	(6.28)
$$py = pyt - L_4 \sin q_1$$	(6.29)
$$pz = pzt - L_1$$	(6.38)
$$q_2 = \tan^{-1}(\frac{pz}{\sqrt{px^2 + py^2}}) - \sin^{-1}(\frac{L_3 \sin q_3}{\sqrt{px^2 + py^2 + pz^2}})$$	(6.31)
$$q_3 = \cos^{-1}(\frac{px^2 + py^2 + pz^2 - L_2^2 - L_3^2}{2L_2 L_3})$$	(6.32)
$$q_4 = \frac{\pi}{2} - q_2 - q_3$$	(6.33)

Donde se tienen como entradas el vector de posición en el espacio y la orientación de la herramienta (*pxf, pyf, pzf, q_5*). Y como la salidas la orientación de cada articulación determinada por el vector (q_1, q_2, q_3, q_4).

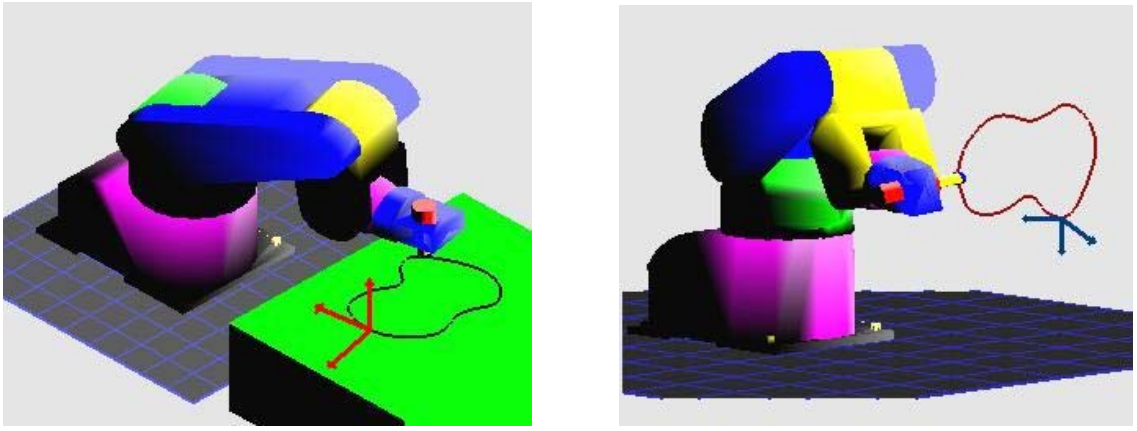
Una vez que se tienen resueltos los casos de cinemática directa e inversa, se retoma el procedimiento de generación de la trayectoria que ha de seguir el robot manipulador.

En el CD anexo a la tesis se presentan el diagrama de flujo y la función ***cinematica_inversa*** en Matlab que resuelve la cinemática inversa del robot manipulador para el caso (a), y se presenta la función ***Robot3DB*** que realiza la simulación en 3D de la cinemática inversa para el caso (a).

6.4 GENERACIÓN DE TRAYECTORIA

Como se mencionó al inicio del capítulo, los resultados obtenidos de la cinemática directa e inversa del robot manipulador, así como los resultados del procesamiento de imágenes y de la interpolación, permiten generar automáticamente la trayectoria del robot manipulador.

En esta sección se desarrolla el procedimiento para la generación de la trayectoria a partir de los resultados de la interpolación. Al igual que para la solución de la cinemática inversa se consideran dos casos de interés. El caso (a) donde la herramienta del manipulador se mantiene orientada constantemente en dirección ortonormal a la trayectoria, (el vector ortonormal es el producto cruz entre los vectores tangente y normal), figura 6.12(a). Para el caso (b) se pretende que la herramienta del manipulador se mantenga orientada aproximadamente en dirección normal a la trayectoria, figura 6.12(b).



(a) *(b)*

Figura 6.12. (a) Dirección ortonormal, (b) Dirección normal.

Dado que el proceso para generar la trayectoria es análogo tanto para la interpolación por splines como para la interpolación circular, sólo se desarrollará el proceso tomando en cuenta la interpolación por splines.

Definiendo el tipo de trayectoria que se va a manejar, que es el tipo de trayectoria continua, definida en la sección 4.2, y recordando de las limitaciones y alcances de la tesis, que sólo se manejan trayectorias formadas por curvas suaves, se procede a explicar cómo se genera la trayectoria.

Ya que la trayectoria está determinada por una serie de puntos que representan la orientación y la posición de la herramienta en el espacio de trabajo del robot manipulador. Una característica deseable de ésta serie de puntos es que haya continuidad entre ellos y que la trayectoria que formen sea suave.

Entonces, para generar la trayectoria para el robot manipulador, primero hay que conocer su posición y orientación en el espacio. Para conocer la posición basta con ubicar la trayectoria deseada en el espacio de trabajo del manipulador, en la figura 6.13 se muestra un ejemplo de ubicación de la trayectoria en el espacio de trabajo del manipulador, la generación del espacio de trabajo del manipulador se logra con ayuda de su modelo cinemático y se comprueba en la simulación.

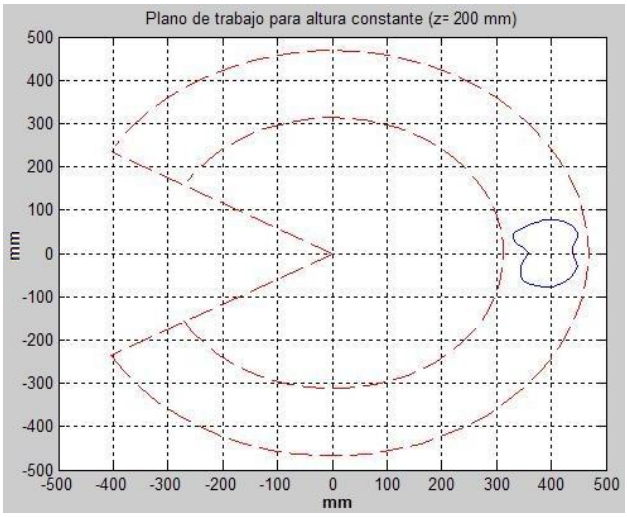


Figura 6.13. Ubicación de trayectoria en espacio de trabajo.

Por otro lado para determinar la orientación de la trayectoria se utilizan las ecuaciones obtenidas de la interpolación, determinando los vectores tangente, normal y ortonormal a la trayectoria, vectores que finalmente ayudan a representar la orientación de la herramienta del manipulador en el espacio. Este procedimiento se desarrolla a continuación:

Primero, de la interpolación se obtienen un par de ecuaciones paramétricas, representadas por dos polinomios de grado 3, determinados por las fórmulas 6.39 y 6.40,

$$P_{xi}(t) = d_i t^3 + c_i t^2 + b_i t + a_i \tag{6.39}$$

$$P_{yi}(t) = h_i t^3 + g_i t^2 + f_i t + e_i \tag{6.40}$$

por las propiedades de los polinomios, se sabe que son fácilmente diferenciables, teniendo así que las derivadas de los polinomios 6.39 y 6.40 están determinadas en las fórmulas 6.41 y 6.42.

$$P'_{xi}(t) = 3d_i t^2 + 2c_i t + b_i \tag{6.41}$$

$$P'_{yi}(t) = 3h_i t^2 + 2g_i t + f_i \tag{6.42}$$

Para determinar los vectores tangente y normal a partir de las ecuaciones anteriores, es suficiente aplicar las siguientes fórmulas a los polinomios,

Para el vector tangente de P_{xi}: $T_{xi} = P_{xi}(t_0) + P'_{xi}(t_0)$ (6.43)

Para el vector tangente de P_{yi}: $T_{yi} = P_{yi}(t_0) + P'_{yi}(t_0)$ (6.44)

Para el vector Normal de P_{xi}: $N_{xi} = P_{xi}(t_0) + P'_{yi}(t_0)$ (6.45)

Para el vector Normal de P_{yi}: $N_{yi} = P_{yi}(t_0) - P'_{xi}(t_0)$ (6.46)

Por lo que, si para graficar la posición de la trayectoria bastaba evaluar para ($P_x(t)$, $P_y(t)$), para el vector tangente hay que evaluar para ($T_x(t)$, $T_y(t)$) y para el vector normal hay que evaluar para ($N_x(t)$, $N_y(t)$).

Por otro lado, si se desea conocer la orientación del vector tangente o el vector normal, la forma más sencilla es obtener el ángulo que represente su orientación. Este ángulo está determinado por las fórmulas 6.47 para el vector tangente y 6.48 para el vector normal,

$$\theta_T(t_n) = \arctan\left(\frac{T_y(t_n) - T_y(t_{n-1})}{T_x(t_n) - T_x(t_{n-1})}\right) \tag{6.47}$$

$$\theta_N(t_n) = \arctan\left(\frac{N_y(t_n) - N_y(t_{n-1})}{N_x(t_n) - N_x(t_{n-1})}\right) \tag{6.48}$$

cualquiera de los últimos ángulos puede representar la orientación de la herramienta en el espacio de trabajo del manipulador, uno en dirección tangente y otro en dirección normal.

El conocer la posición y orientación de la trayectoria es de vital importancia, ya que si se recuerda, de los cálculos de la cinemática inversa del robot manipulador, es necesario conocer los parámetros de entrada, que corresponden a la posición y a la orientación del elemento final del robot en el espacio de trabajo. Así calculando la cinemática inversa para cada punto de la trayectoria, se están generando automáticamente las posiciones angulares de cada articulación que satisfacen el seguimiento de la trayectoria especifica para el robot manipulador.

Una vez que se tienen estos cálculos se limita la generación de la trayectoria del robot, para cada uno de los casos ya mencionados.

6.4.1 Generación de trayectoria caso (a)

El criterio de seguimiento de trayectoria (a) se ilustra en la figura 6.14 (donde se muestran los vectores tangente, normal y ortonormal para diferentes puntos de la trayectoria), siendo el

vector ortonormal a la trayectoria, el que represente la orientación de la herramienta del robot para este caso.

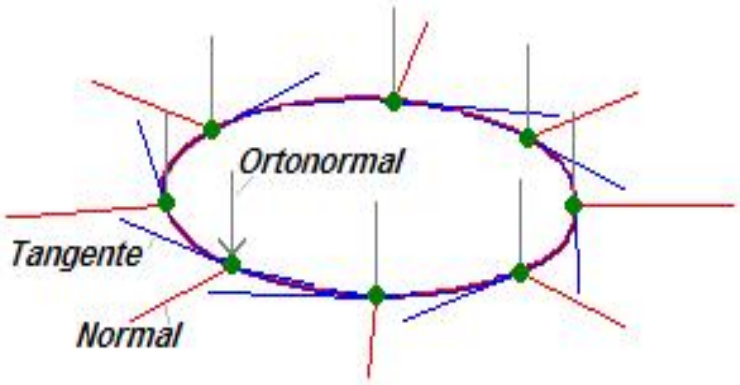


Figura 6.14. Vectores en diferentes puntos de la trayectoria.

Para entender más el caso (a), se agrega la figura 6.15, donde se representa al robot manipulador pasando por un punto de la trayectoria, manteniendo siempre la herramienta en dirección ortonormal a la misma.

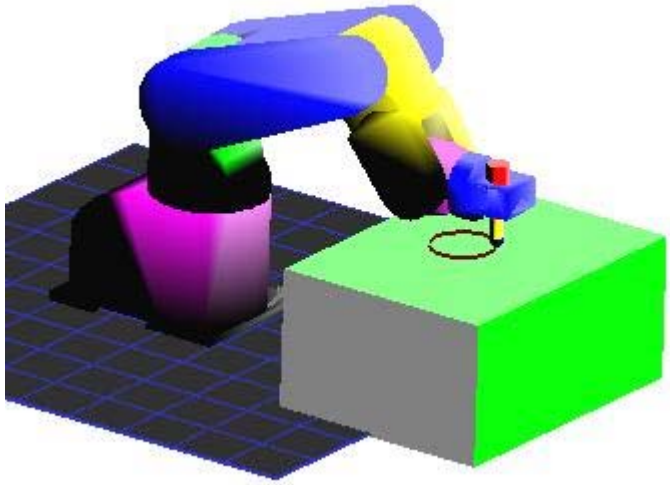


Figura 6.15. Robot con herramienta ortonormal a trayectoria.

El vector ortonormal, es el resultado del producto cruz entre el vector tangente y el vector normal, para el caso (a) no hay gran problema ya que el vector ortonormal permanece en la misma dirección para toda la trayectoria, simplificándose así los cálculos de la cinemática

inversa como ya se observo en la sección 6.3.1 caso (a), sin embargo, se analizará también el caso (b), que es un poco más complicado, donde se pretende que la herramienta siga a la trayectoria en dirección normal a esta. En el CD anexo se presentan los diagramas de flujo generales y las funciones que se utilizan para generar la trayectoria para el caso (a) que son: ***trayectoria_splines***, ***trayectoria_circular*** en Matlab y ***Robot3DCvistas*** en OpenGL.

6.4.2 Generación de trayectoria caso (b)

Para el caso (b) el criterio de seguimiento de la trayectoria, como se había mencionado, es ahora en dirección del vector normal a la trayectoria, vector ilustrado en la figura 6.14. Esto es, la herramienta sigue la trayectoria en orientación normal a la misma. En la figura 6.16 se ilustra un instante de seguimiento de la trayectoria con el robot manipulador.

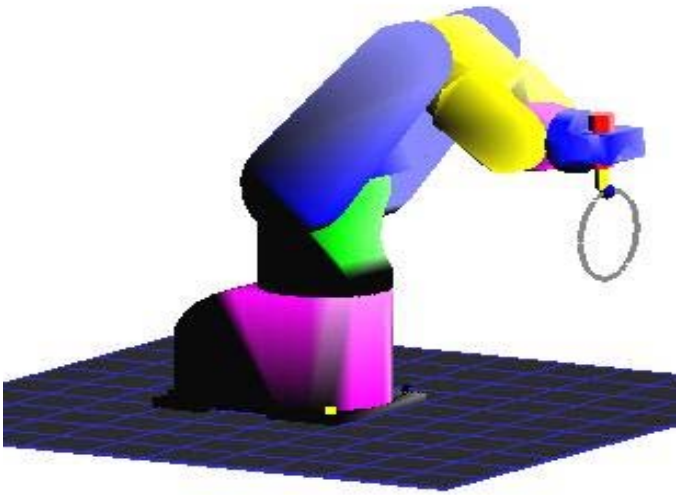


Figura 6.16. Robot con herramienta normal a trayectoria.

Recordando de la sección 6.3.2, donde se resolvió el problema de cinemática inversa para este caso, que no es posible para el robot seguir este tipo trayectorias como tal, sólo se realizará una aproximación del seguimiento de la trayectoria del robot a la trayectoria deseada.

Este caso, a diferencia del anterior, es un poco más complicado, ya que al querer mantener la herramienta en dirección normal a la trayectoria implica que ahora el quinto grado de libertad

del robot q_5 sea variable. Es decir, q_5 tendrá ahora un valor, el cual está representado por la orientación del ángulo del vector normal θ_N, ángulo representado en la fórmula 6.48.

Cabe señalar, que al generar cualquier tipo de trayectoria hay que tener cuidado en no sobrepasar las velocidades y aceleraciones máximas permitidas por el robot, por lo que hay que generar la trayectoria y evaluar para los vectores dq_i y d^2q_i, que representan las velocidades y aceleraciones de las articulaciones del manipulador respectivamente, en la figura 6.17 se muestra un ejemplo para la trayectoria que se ha venido manejando (figura 6.12(a)), donde se muestran las graficas de posiciones, velocidades y aceleraciones de las primeras 4 articulaciones, ya que se está considerando el caso (a), donde la herramienta está en orientación ortonormal y la ultima articulación no se mueve ($q_5=0$). De la misma figura hay que comparar las velocidades y aceleraciones máximas, con los valores más altos permitidos por el robot, representados en las tablas 6.1 y 6.2.

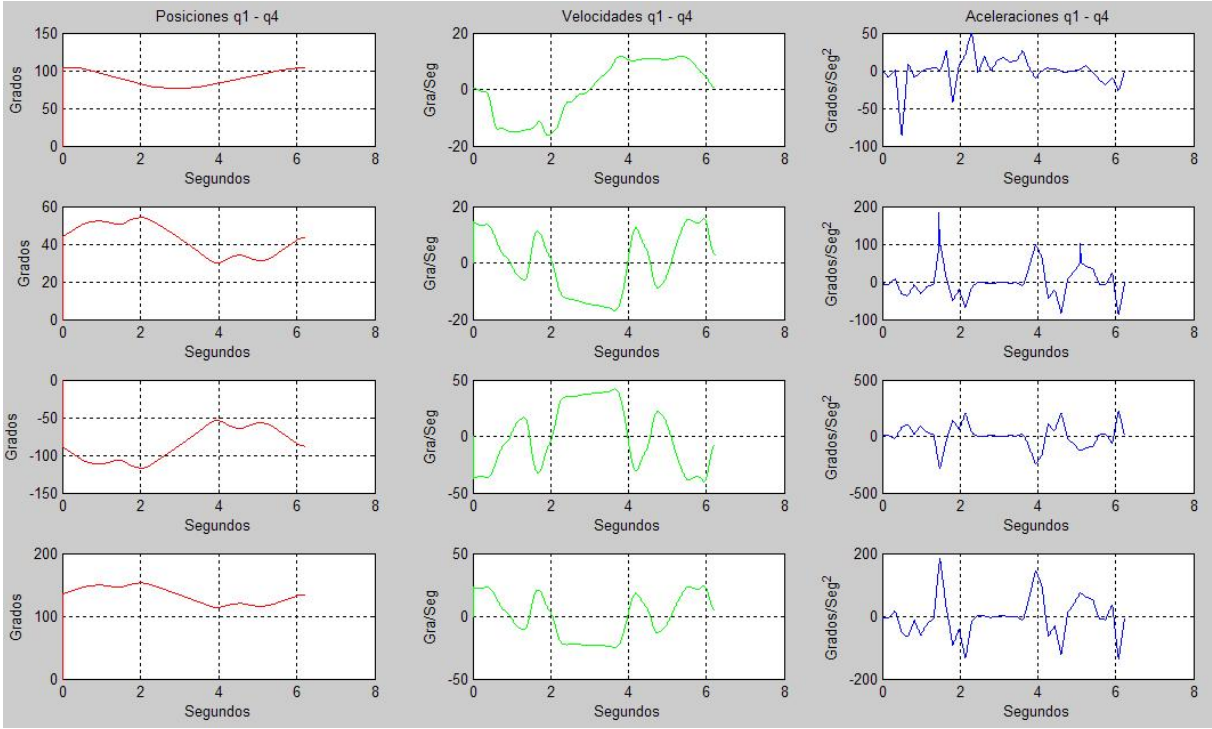


Figura 6.17. Posición, velocidad y aceleración de las articulaciones.

Una vez que ya se generaron todos estos cálculos y simulaciones con ayuda Matlab, se exportan los datos a C++ [11], donde con ayuda de librerías de OpenGL se simula la

trayectoria y se comprueba que efectivamente el robot manipulador la puede seguir de forma adecuada. En el CD anexo a la tesis se presentan las funciones que se utilizan para generar la trayectoria para el caso (b) que son: ***trayectoria_casob***, en Matlab y ***Robot3DDvistas*** en C++.

Por último, si la simulación representa correctamente el seguimiento de la trayectoria deseada, se generan automáticamente los programas necesarios para el robot manipulador, uno que representa las instrucciones y otro que representa los puntos involucrados en la trayectoria, para finalmente cargar este par de programas al robot, ejecutarlos y corroborar cómo el robot sigue la trayectoria que se había simulado. El programa para generar el código del robot, llamado ***codigo***, se presenta en el CD anexo a esta tesis.

En este capítulo se representó todo lo referente a la cinemática del robot manipulador, generando la trayectoria asociada a la imagen digital, además se explicó la importancia de la simulación y finalmente se llego a la programación real del robot. En el siguiente capítulo, se muestran diferentes resultados y se calculan algunos errores involucrados a lo largo del procedimiento hasta aquí desarrollado.

CAPÍTULOS 7-8
RESULTADOS Y CONCLUSIONES

CAPÍTULO 7

RESULTADOS EXPERIMENTALES

En este capítulo se muestran algunos resultados obtenidos a lo largo del desarrollo del proyecto, que son principalmente el cálculo de algunos errores involucrados. Como primer resultado del procedimiento, se explicará y analizará la obtención del error asociado a la adquisición de la imagen, comparando la figura original con la imagen adquirida, posteriormente se obtiene el error asociado a la interpolación, comparando el contorno de la imagen con el contorno de la interpolación y finalmente se obtiene el error asociado a la ejecución del robot, comparando la figura original con la trayectoria que finalmente reproduce el robot.

Para el análisis de estos errores, los cálculos se fundamentan en el uso de diferentes figuras, pero hay una en especial, el círculo, ya que es una figura geométrica conocida, fácil de parametrizar y cumple con las condiciones iniciales del proyecto. También se muestran resultados meramente ilustrativos para diferentes formas, debido al mencionado problema de parametrización.

7.1 ERROR DE ADQUISICIÓN

Uno de los primeros problemas que se presentó al desarrollar este trabajo, fue el conocer el error que se genera al adquirir la imagen.

Como se ha venido manejado en el proyecto, el punto de interés del objeto (o figura) es su contorno asociado, por lo que la obtención del error se enfoca en la diferencia que existe entre el contorno de la figura real y el contorno obtenido de la figura de la imagen digital, específicamente la imagen binarizada. Para obtener este error se define el siguiente procedimiento:

- Se tiene como figura real a un círculo con radio constante *r*, como lo ilustra la figura 7.1.
- Se adquiere la imagen de la figura real, conociendo los parámetros de adquisición (principalmente la escala).

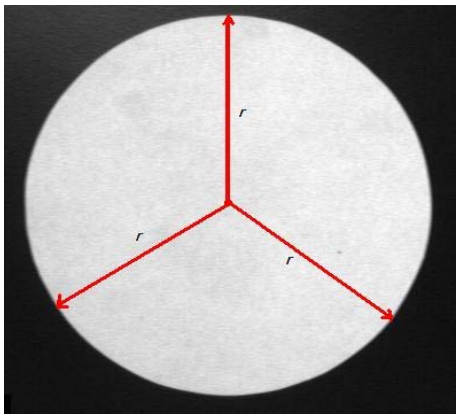
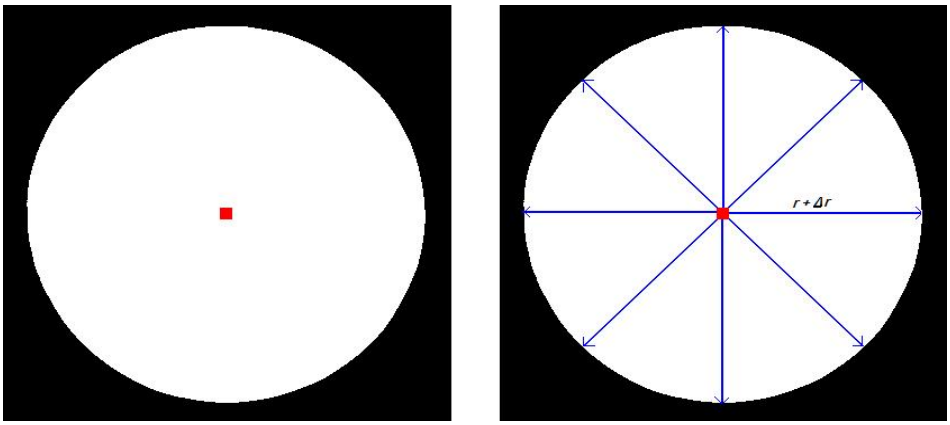


Figura 7.1. Representación de círculo real con radio constante.

- Se binariza la imagen y se obtiene el centroide correspondiente al círculo, figura 7.2(a).
- Del centroide del círculo se trazan *n* vectores normales para diferentes ángulos, apuntando hacia el contorno del círculo, guardando la magnitud de cada vector, que finalmente representa la magnitud del radio para el ángulo correspondiente, figura 7.2(b).



(a) *(b)*

Figura 7.2. (a) Centroide, (b) Vectores normales.

- Finalmente se compara cada uno de éstos vectores con el radio original, para determinar, el error máximo, la media y su desviación.

De esta manera queda determinado el error de adquisición como, $eA=r-(r-\Delta r)=\Delta r$. En la tabla 7.1, se muestran los resultados del error de adquisición para tres diferentes círculos con radios conocidos, considerando 128 vectores normales.

Tabla 7.1. Error de adquisición.

Radio Real (mm)	Error máximo (mm)	Media (mm)	Desviación (mm)
30.0	0.6469	0.6273	0.0527
50.0	0.3327	0.3113	0.0439
65.0	0.5500	0.5221	0.0599

En la figura 7.3 se ilustra la imagen binaria, así como los 128 vectores normales usados para calcular el error del círculo con radio de 65mm.

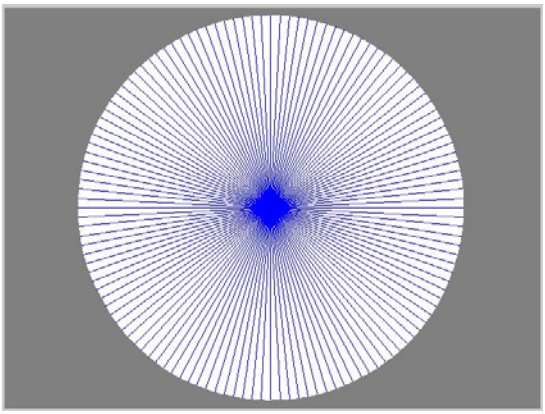


Figura 7.3. Imagen binaria y vectores normales.

De los resultados de la tabla 7.1 se puede concluir que el error de adquisición más notable es debido a cuestiones de dimensionamiento o escala, ya que la desviación del error es mínima y que el error máximo es casi igual al error medio. Esto implica que existe un error casi despreciable en cuanto a forma del contorno y que existe un error de dimensionamiento mínimo.

Por otro lado, las diferentes desviaciones del error asociadas a las diferentes magnitudes de radio de los círculos no dicen nada en realidad, ya que las desviaciones no presentan ningún patrón de proporcionalidad.

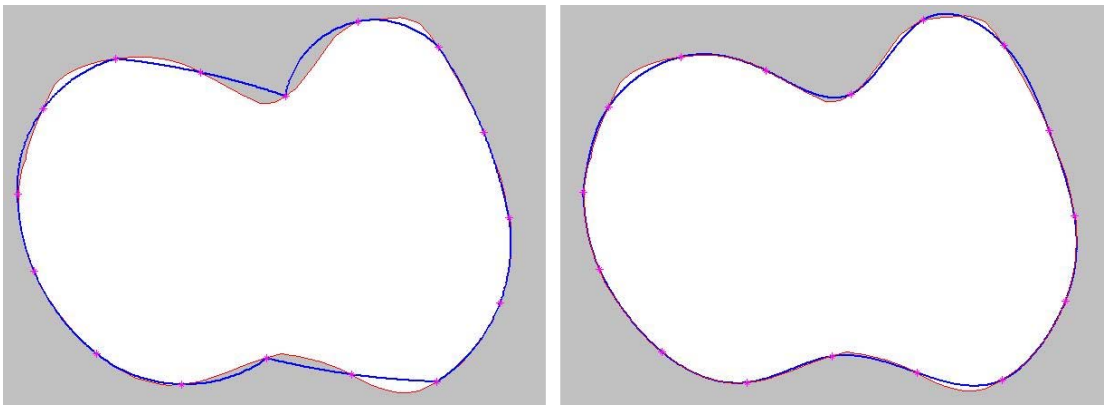
Cabe señalar que este error sólo se calculó para círculos con diferentes radios y no para otras figuras, ya que como se menciona al principio del capítulo, el círculo es una figura conocida paramétricamente, condición necesaria para calcular este tipo de error, y que además es fácilmente representado por curvas suaves. En cuanto a alguna otra figura, sería difícil de parametrizar, sin embargo, se sabe que cualquier figura representada por curvas suaves, puede ser formada a base de una serie de círculos, por lo que el error obtenido con círculos es suficiente para tener una idea del error de adquisición para cualquier tipo de figura representada por curvas suaves.

7.2 ERROR DE INTERPOLACIÓN

Posteriormente de calcular el error de adquisición, el siguiente error que se presento fue el de interpolación, tanto para la interpolación circular como para la interpolación por splines.

Al igual que para el error de adquisición, la obtención del error se enfoca al contorno de la figura, específicamente en la diferencia que existe entre el contorno obtenido de la figura de la imagen binaria y el contorno que se genera con las curvas paramétricas de la interpolación. Para obtener el error de interpolación se definen dos casos, el error asociado a la interpolación circular y el error asociado a la interpolación por splines, sin embargo, el procedimiento para calcular cualquiera de éstos es el mismo y a continuación se define:

- Se genera la figura con las ecuaciones de la interpolación y se gráfica sobre el contorno de la imagen, figura 7.4(a) Circular y 7.4(b) Splines (ej. para 16 puntos).



(a) *(b)*

Figura 7.4. (a) Contorno vs Circular, (b) Contorno vs Splines.

- Se obtienen los vectores normales a las curvas de interpolación, los suficientes para representar adecuadamente al contorno. En la figura 7.5 se muestran sólo algunos de los vectores normales al contorno especificado.

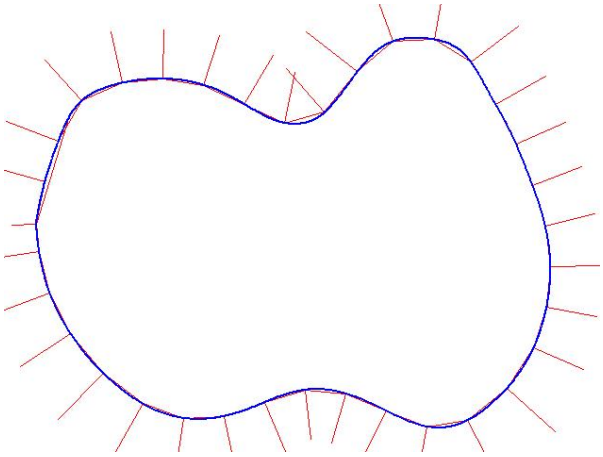


Figura 7.5. Vectores normales a las curvas de interpolación.

- A partir de las curvas de interpolación, en ambos sentidos y en dirección de los vectores normales se busca la intersección con el contorno de la imagen, guardando la distancia en un vector que represente el error de interpolación, en la figura 7.6 se representa este procedimiento para las splines.

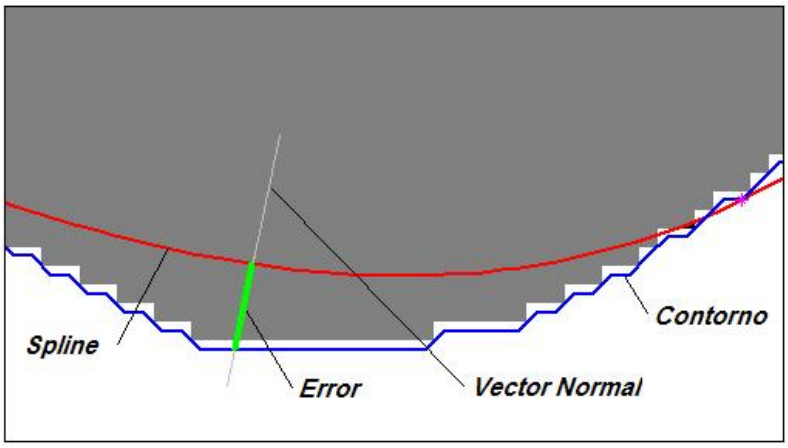


Figura 7.6. Intersección vector normal.

- Finalmente se obtienen el error medio, el error máximo y su desviación.

Un caso evidente, es que el error de interpolación depende del numero de puntos que se seleccionen para la interpolación a partir del contorno, sin la necesidad de que a mayor numero de puntos considerados menor error de interpolación, ya que se llega a un punto donde no necesariamente al aumentar el numero de puntos disminuye el error. Para ejemplificar los resultados del error de interpolación, se presentan diferentes casos;

(a) Para el primer caso se utiliza el contorno de un círculo con radio de 65mm, considerando diferentes cantidades de puntos para la interpolación, en la tabla 7.2 se muestran los resultados del error asociados a la interpolación:

Tabla 7.2. Resultados del error de interpolación, caso (a).

Puntos Considerados	Error Máximo (mm)		Error Medio (mm)		Desviación (mm)	
	Circular	Splines	Circular	Splines	Circular	Splines
8	0.5044	2.5678	0.0664	0.3449	0.1576	0.5695
16	0.4047	0.6555	0.0376	0.0543	0.1169	0.1512
32	0.3933	0.3990	0.0260	0.0307	0.1186	0.1171
64	0.4702	0.5016	0.0364	0.0307	0.1282	0.1326

De la tabla 7.2, se pueden concluir dos cosas. Primero, que el error no es proporcional al numero de puntos, tal como se puede observar en la tabla, cuando se escalan los puntos de 32 a 64 el error no disminuye por el contrario aumenta. Y segundo, que la interpolación circular es mejor que la interpolación por splines, pero sólo para trayectorias muy suaves, para comprobar esto en los siguientes ejemplos se realizan los cálculos de error para contornos más irregulares.

(b) El segundo caso es para el contorno representado en la figura 7.7, de igual forma se consideran diferentes cantidades de puntos para la interpolación, en la tabla 7.3 se muestran los resultados del error asociados a la interpolación:

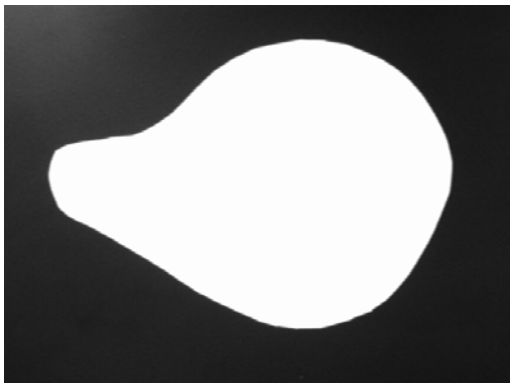


Figura 7.7. Prueba del error de interpolación (b).

Tabla 7.3. Resultados del error de interpolación, caso (b).

Puntos Considerados	Error Máximo (mm)		Error Medio (mm)		Desviación (mm)	
	Circular	Splines	Circular	Splines	Circular	Splines
8	6.7688	7.2732	1.3624	1.3197	1.8410	1.8671
16	4.1154	3.9358	0.5187	0.2784	0.8131	0.6011
32	0.9433	0.8977	0.0874	0.0618	0.1860	0.1649
40	0.7125	0.6013	0.0598	0.0372	0.1540	0.1339

De la tabla 7.3, se dedujeron un par de conclusiones. Primero se observa que para este caso no se consideraron los 64 puntos como en el caso anterior, debido a una de las desventajas que

presenta la interpolación circular, y es que los tres puntos considerados para interpolar el arco del círculo no deben estar alineados, de lo contrario no es posible interpolar. El segundo es que la interpolación por splines es mejor que la circular para contornos más irregulares, como ya se había mencionado.

(c) El tercer y último caso, consiste en otro contorno no tan suave, como se muestra en la figura 7.8. y los resultados asociados al error de interpolación se muestra en la tabla 7.4.

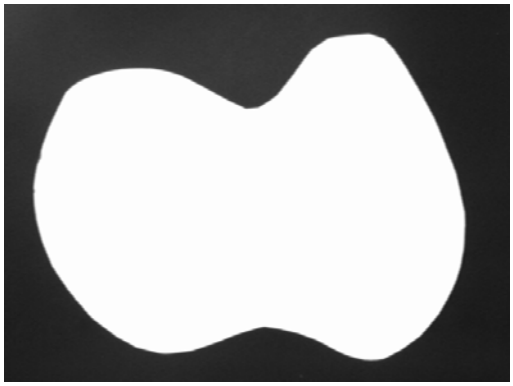


Figura 7.8. Prueba del error de interpolación (c).

Tabla 7.4. Resultados del error de interpolación, caso (c).

Puntos Considerados	Error Máximo (mm)		Error Medio (mm)		Desviación (mm)	
	Circular	Splines	Circular	Splines	Circular	Splines
8	16.5300	11.5454	3.1107	2.4220	3.9932	2.6812
16	4.5913	2.1631	0.9709	0.3316	1.2120	0.4716
32	1.6074	1.2397	0.1605	0.1045	0.2965	0.2245
50	0.7153	0.5985	0.0918	0.0634	0.1670	0.1560

De la tabla 7.4 y considerando los resultados anteriores, se corroboran las conclusiones que se han manejado. Que la interpolación circular es mejor para contornos muy suaves y regulares, y que la interpolación por splines es mejor para contornos no tan suaves aceptando algunas

irregularidad y por ultimo que el error no es proporcional al numero de puntos considerados, sino que depende además de su ubicación en el contorno.

Otro punto que se debe tomar en cuenta es la selección del numero de puntos, ya que como se había mencionado, se limito a cierto número debido a que entre más puntos se consideren es más fácil encontrar tres puntos alineados. Esto ocasiona que no se pueda realizar la interpolación circular y por consiguiente no se pueda tener comparación con la interpolación por splines. Sin embargo, cabe señalar que la interpolación por spline si puede admitir tantos puntos como se requiera, siempre y cuando sean menos que los puntos que definen al contorno.

Por otro lado, debido a que el error de interpolación depende en gran medida de la cantidad y la ubicación de los puntos de interpolación sobre el contorno, hay que tomar en cuenta cómo se seleccionan los puntos. Recomendando ubicar más puntos para regiones con curvas no tan suaves y menos para curvas más suaves. Un ejemplo de la selección de puntos más optima se muestra en la figura 7.9.

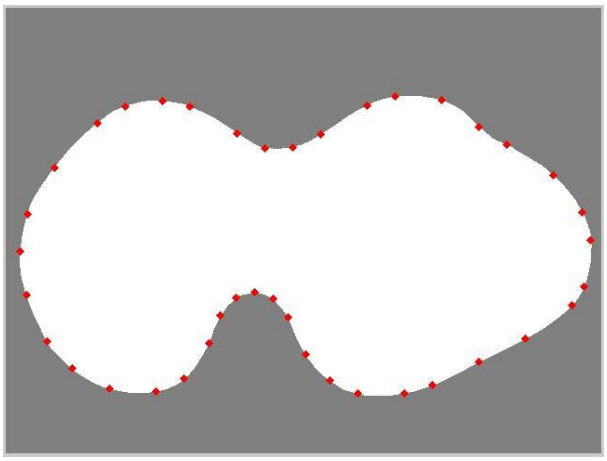
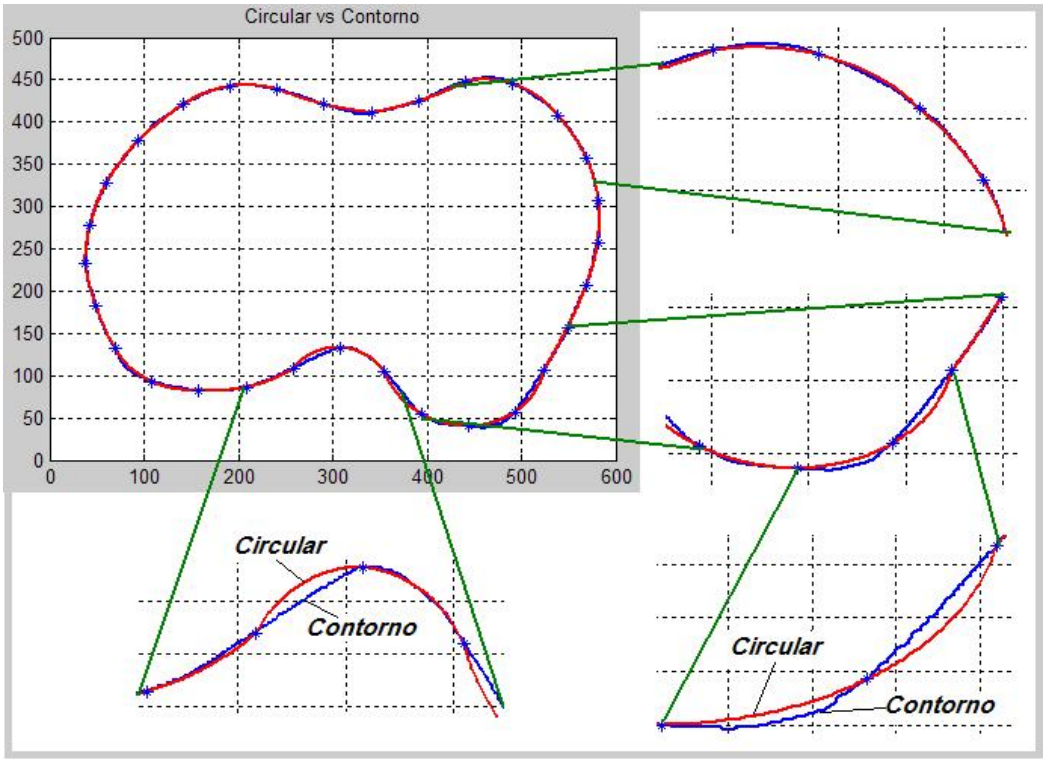
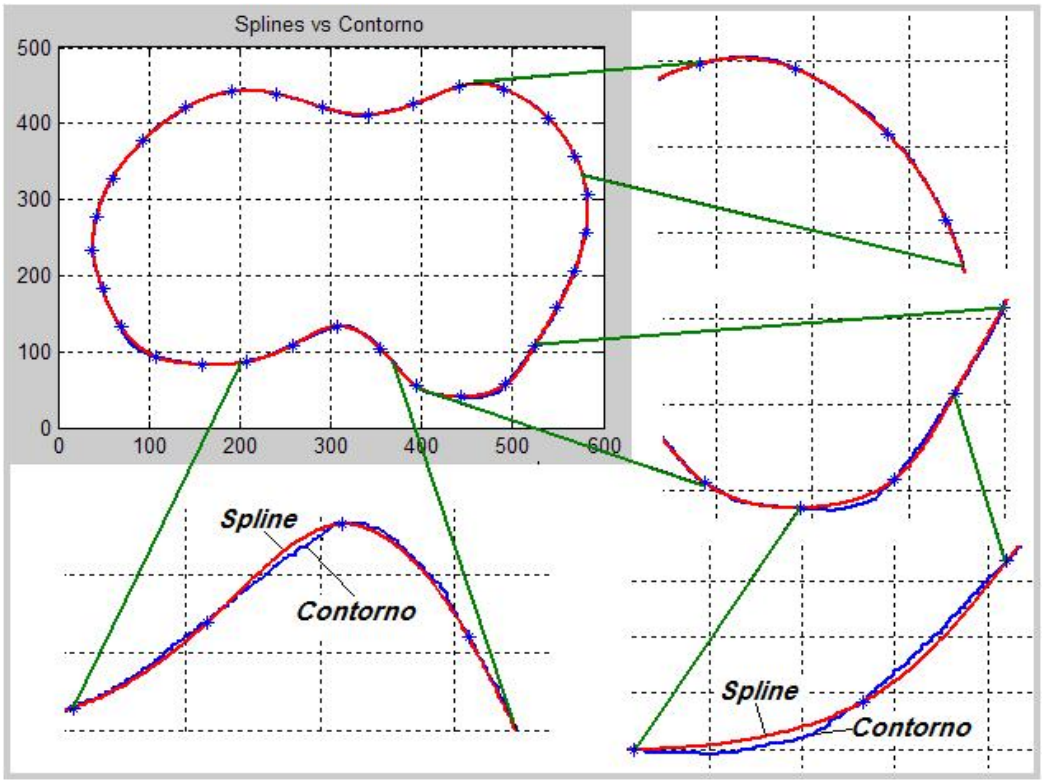


Figura 7.9. Criterio de selección de puntos por usuario.

Por otro lado, en la figura 7.10, se muestra la comparación entre la interpolación circular y el contorno, además de algunas ampliaciones en regiones críticas del contorno, considerando 30 puntos de interpolación. Y en la figura 7.11, se muestra el mismo análisis, pero ahora considerando la interpolación por splines.

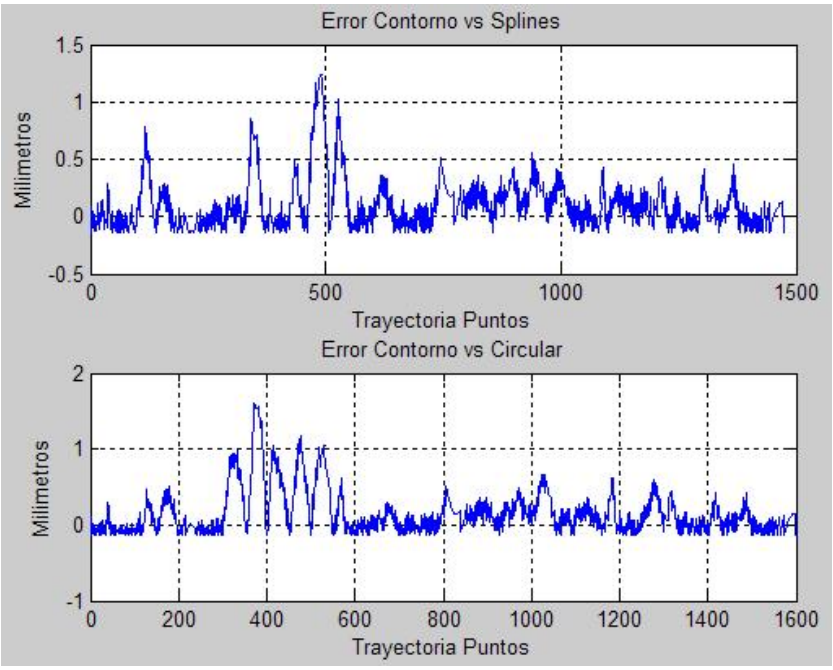


7.10. Interpolación circular vs contorno y ampliaciones.



7.11. Interpolación splines vs contorno y ampliaciones.

Finalmente en la figura 7.12 se muestran las gráficas del error de interpolación circular e interpolación por splines, para la figura 7.8, considerando los 32 puntos que se representan en la tabla 7.4.



7.12. Errores de interpolación contorno vs spline y circular.

Donde se observa que la interpolación por splines es mejor que la interpolación circular, ya que tanto el error máximo como el error medio son menores para la interpolación por splines.

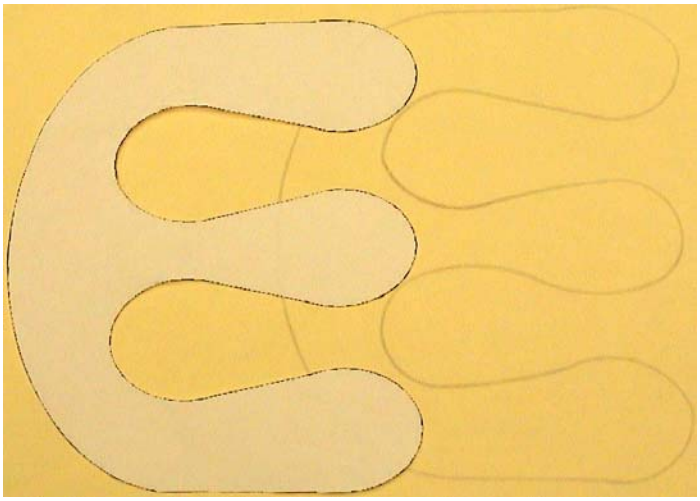
Hasta el momento se tiene el error de adquisición, así como el error de interpolación, donde se obtuvieron resultados interesantes. Finalmente el último error que se presenta, es el error de ejecución del robot manipulador, provocado principalmente por errores o desviaciones mecánicas asociadas a éste. La determinación del error de ejecución se muestra en seguida.

7.3 ERROR DEL MANIPULADOR

El error del manipulador, se refiere al error que se presenta al comparar la ejecución del robot que consiste en dibujar el contorno sobre una plantilla y compararlo con el objeto inicial. Esta

comparación es posible realizarla por una simple inspección visual para figuras geométricamente desconocidas o hacer algunas mediciones para figuras geométricamente conocidas.

En la figura 7.13, se muestra una comparación entre un contorno dibujado por el robot y el objeto inicial de una figura geométricamente desconocida.



7.13 Foto dibujo del robot vs objeto real.

Para este caso es difícil parametrizar ambas figuras, la original y la reproducida por el robot manipulador, simple y sencillamente se hace un análisis visual. Sin embargo, hay un factor que es determinante, y es que el robot usado sólo realiza curvas a base de la interpolación circular. De esta manera el resultado del dibujo del robot sólo es comparable con la interpolación circular y no para la interpolación por splines.

El caso donde la figura es un círculo (geométricamente conocido), es posible realizar una mejor comparación ya que el círculo inicial presenta un radio constante conocido y es posible compararlo para el círculo reproducido por el robot, de la siguiente manera:

Primero se tiene el radio constante del círculo original, el cual es la referencia, posteriormente del círculo dibujado por el robot se toman diferentes medidas del radio, teniendo así que el error es la diferencia de las medidas del radio original con el radio medido. Se realizan 16

medidas equidistantes alrededor del círculo usando un vernier. En la tabla 7.5, se muestran los resultados para un círculo de radio 65mm.

Tabla 7.5. Error del manipulador 1.

No.	Radio (mm)	No.	Radio (mm)	No.	Diámetro (mm)	No.	Error Radio	No.	Error Radio	No.	Error Diámetro
1	65.35	9	65.25	1	130.60	1	0.35	9	0.25	1	0.60
2	65.20	10	64.90	2	130.10	2	0.20	10	0.10	2	0.10
3	65.20	11	64.90	3	130.10	3	0.20	11	0.10	3	0.10
4	65.00	12	64.80	4	129.80	4	0.00	12	0.20	4	0.20
5	65.15	13	64.95	5	130.10	5	0.15	13	0.05	5	0.10
6	65.60	14	65.10	6	130.70	6	0.60	14	0.10	6	0.70
7	65.50	15	65.40	7	130.90	7	0.50	15	0.40	7	0.90
8	65.50	16	65.40	8	130.90	8	0.50	16	0.40	8	0.90

Finalmente en la tabla 7.6 se muestra el error máximo, el error medio y su desviación:

Tabla 7.6. Error del manipulador 2.

Radio (mm)	Error Max. (mm)	Error Med. (mm)	Desviación (mm)
65.000	0.600	0.256	0.180

En las tablas anteriores se muestra un resultado, que se considera como el error final, ya que se compara la figura inicial (círculo con radio constante y conocido) y la figura final (círculo que dibuja el robot) obteniendo un error absoluto entre ambas. Sin embargo, el error final tiene implícitos el error de adquisición, el error de interpolación circular, errores de medición y finalmente un error de calibración del robot.

Algo claro, es que si el robot está bien calibrado y la interpolación que realiza es de tipo circular, entonces el dibujo que realice el robot del contorno y el dibujo realizado por la interpolación circular del contorno deben ser iguales. Sin embargo, al ejecutar las primeras pruebas existía una gran diferencia entre el dibujo del robot y el dibujo de la interpolación circular, debido a una descalibración mecánica que presenta el robot y que fue necesario corregirla por software, en el anexo 2 se explica el procedimiento que se realizó para la

calibración del robot, una calibración no muy adecuada pero que ayudo a que el error final se redujera considerablemente.

Finalmente debido al ajuste de calibración que fue necesario aplicarle al robot, no se puede dar un resultado contundente de cómo cada uno de los errores implícitos (de adquisición, de interpolación circular y del mismo robot), repercuten en los resultados finales. Sin embargo, algo que es claro es que si el robot estuviera bien calibrado y su funcionamiento fuera el adecuado, se podría discriminar el error del robot y el error final solo sería debido a los errores de adquisición e interpolación.

CAPÍTULO 8
CONCLUSIONES

En este último capítulo se analizan los resultados obtenidos a lo largo del desarrollo del proyecto de tesis, donde se incluyen los resultados para las diferentes etapas que se manejaron así como las perspectivas de trabajo a futuro del proyecto.

8.1 PROCESAMIENTO DE IMÁGENES

En la sección referente al procesamiento digital de imágenes, se manejaron las etapas de adquisición, binarización y generación del contorno, resumiendo los resultados de la siguiente manera:

Para parametrizar la efectividad de la etapa de adquisición se calculó un error de adquisición, el cual representará específicamente la diferencia entre el contorno real y el contorno de la imagen adquirida. Pudiendo concluir que a pesar de que no se puso especial énfasis en la etapa de adquisición, los resultados de los errores son mínimos.

Por otro lado, el algoritmo de binarización trabajó adecuadamente, por lo que no fue necesario buscar alguna variante ni corrección. Esto debido principalmente a que las características de los objetos de los cuales se obtuvieron las imágenes son casi ideales, de forma que la calidad de las imágenes adquiridas es buena y por consiguiente el preprocesamiento es mínimo y muy sencillo.

En cuanto al algoritmo de generación del contorno, se probó con diferentes tipos de figuras, desde las más sencillas como lo son los círculos, hasta algunas más complicadas como la representada en la figura 8.1, y el algoritmo nunca falló. Una recomendación para ver la potencialidad de este algoritmo, seria probarlo con imágenes de menor calidad. Cabe señalar también que el algoritmo sólo genera el contorno del primer objeto o figura que se encuentra

en la imagen, omitiendo todos los objetos o formas adjuntas y que si el contorno no está representado por una figura cerrada o existe ruido en la imagen binaria, el algoritmo se detiene especificando el punto de irregularidad del contorno.

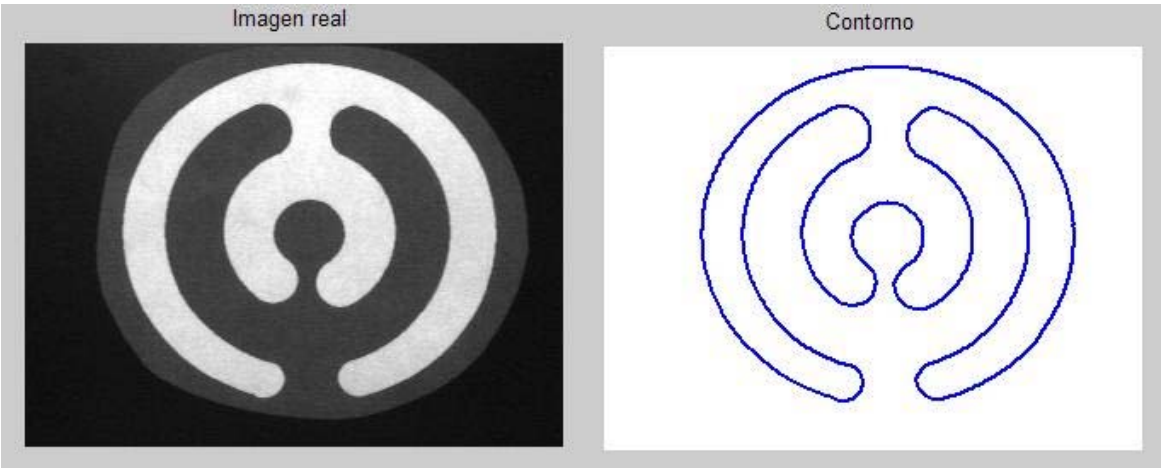


Figura 8.1. Representación de imagen real y su contorno asociado.

Finalmente, del contorno generado se discriminaron algunos puntos, considerando el índice de los datos. Cabe recordar que un mayor numero de puntos no necesariamente es la mejor opción para obtener una representación adecuada del contorno. Se proponen alternativas, como considerar el criterio de la derivada o un método manual donde el usuario seleccione los puntos que él considere que representan mejor al contorno.

8.2 INTERPOLACIÓN Y CURVAS PARAMÉTRICAS

En cuestiones de interpolación los resultados son muy buenos, ya que al utilizar la interpolación se reduce considerablemente el número de puntos que representan al contorno, teniendo además una muy buena aproximación al contorno real.

Las conclusiones para esta sección son las que se han venido manejando: Que la interpolación por splines se considera una muy buena opción para aplicaciones donde el contorno esté representado por curvas no tan suaves, mientras que la interpolación circular es mejor para

curvas muy suaves; y que el error presente en la interpolación depende del numero de puntos seleccionados pero que no es proporcional, es decir, no necesariamente a mayor número de puntos el error es menor, además de considerar que para la interpolación circular entre más puntos se consideren, es más probable tener tres puntos alineados, cuestión no deseable ya que no se puede generar un circulo a partir de los tres puntos alineados.

Finalmente la interpolación por curvas paramétricas de algún contorno no sólo sirve para representar el contorno, sino que gracias a sus propiedades matemáticas sirven para resolver la cinemática de algunas máquinas o robots, principalmente para generar trayectorias. Es decir, la representación paramétrica y su interpolación son la base para el diseño del control cinemático de diferentes máquinas y robots.

8.3 CINEMÁTICA DEL ROBOT

En la sección de la cinemática del robot manipulador se desarrollaron todos los cálculos correspondientes a la solución del problema cinemático, involucrando los resultados obtenidos de la interpolación. A partir de las ecuaciones resultantes de la interpolación se obtienen los vectores tangente, normal y ortonormal a la trayectoria, vectores que son de gran ayuda para representar la orientación en el espacio de la herramienta del robot, dato necesario para resolver la cinemática inversa y para generar la trayectoria a seguir para el robot.

Concluyendo que la interpolación por curvas paramétricas es una herramienta poderosa para resolver la cinemática inversa asociada a robots manipuladores. Esto quiere decir que el conocimiento de los parámetros que representan matemáticamente a cualquier trayectoria son fundamentales para poder solucionar la cinemática inversa asociada a la misma trayectoria, para cualquier máquina o robot, más allá de los mismos robots manipuladores.

8.4 MODELACIÓN Y SIMULACIÓN

Un resultado interesante del desarrollo de este trabajo, fue la generación del modelo y la simulación del sistema en 3-D, desarrollada en C++ utilizando librerías de OpenGL.

La simulación del sistema resultó ser una herramienta muy poderosa para la evaluación de resultados, ya que el robot está modelado a escala, lo que implica que la simulación es muy aproximada al mundo real.

Primero fue de gran ayuda para comprobar los resultados de la cinemática directa, representando la posición y orientación en el espacio de la herramienta correspondiente a los ángulos de entrada dados. En la figura 8.2 se representa esta simulación.

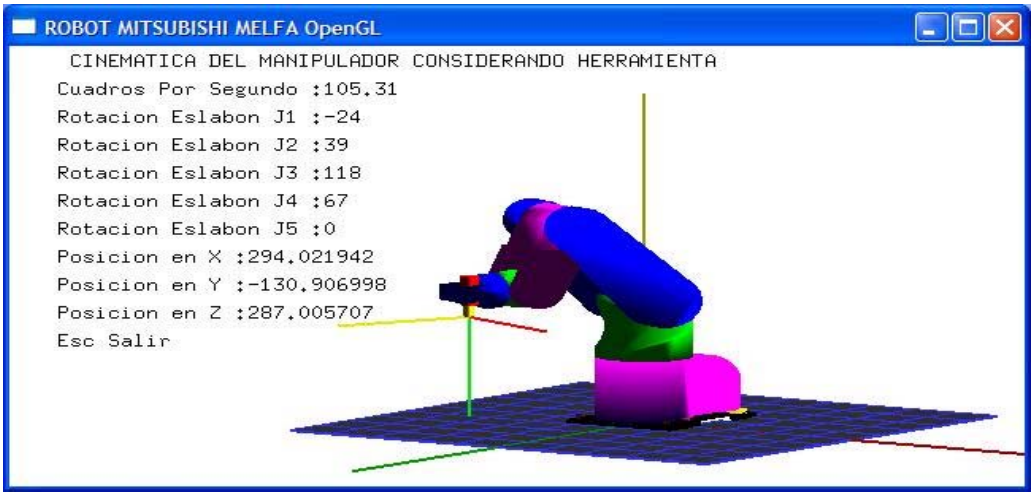


Figura 8.2. Simulación del robot para la cinemática directa.

Posteriormente para resolver la cinemática inversa, y finalmente para la generación de la trayectoria, en la figura 8.3 se representa un instante de la simulación del robot manipulador siguiendo la trayectoria representada en la figura 8.1.

La simulación del robot nos permitió comprobar los resultados de los cálculos realizados de la cinemática del manipulador y también validar el sistema completo. Esto debido a que al modelo de la simulación se le agregaron las restricciones asociadas al robot real, por lo que sí

existe alguna trayectoria que el robot no pueda realizar, ésta se observará previamente en la simulación. Con esto se tiene la certeza de que si la simulación se ejecutó sin ningún problema se puede asegurar que el robot real no tendrá ningún problema en la ejecución de la trayectoria, ahorrando tiempo y esfuerzo.

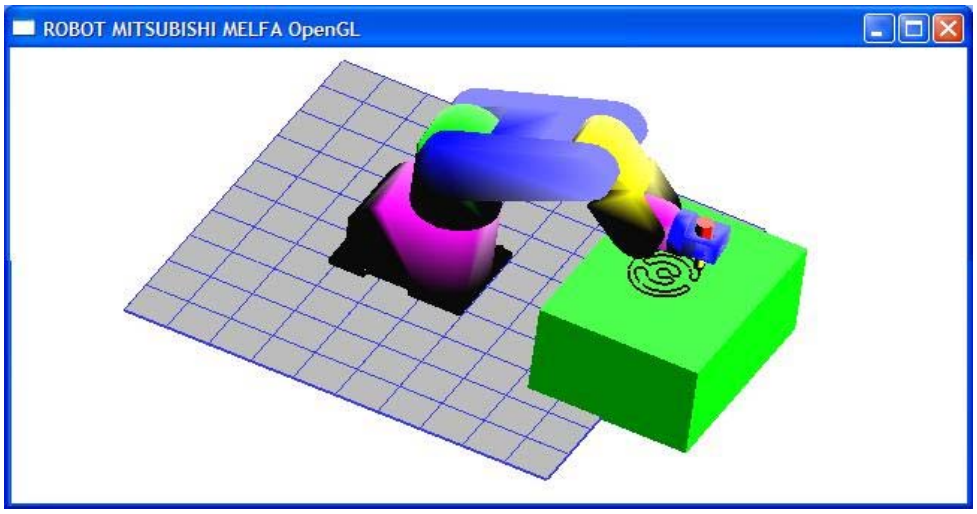


Figura 8.3. Simulación de seguimiento de trayectoria.

Por otro lado la interfaz entre Matlab y C++ que se utilizó no fue más que un archivo de texto *.txt, ya que no era necesario pasar información en tiempo real entre ambos programas [11] y el archivo de texto es la forma más sencilla de pasar información. En Matlab se desarrollaron los cálculos necesarios para posteriormente exportar los resultados a C++ para su simulación, en la figura 8.4 se muestra un diagrama que ejemplifica la interfase entre Matlab y C++.

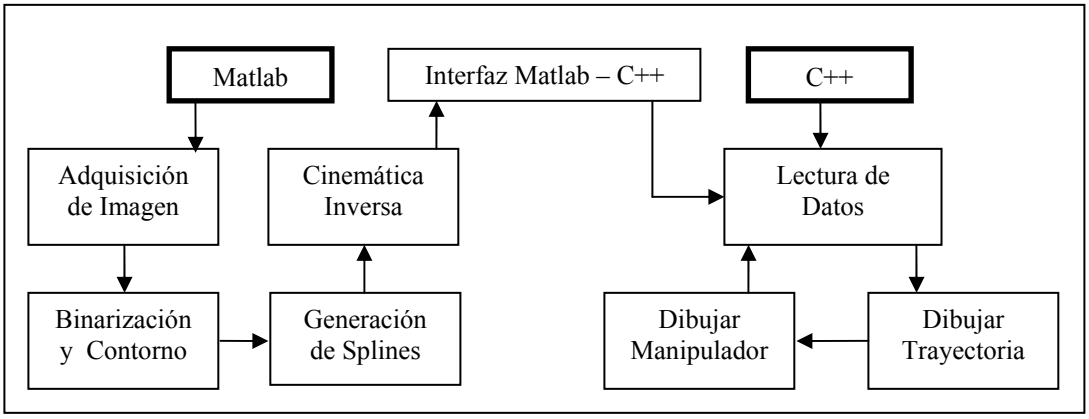


Figura 8.4. Diagrama a bloques, interfaz Matlab C++.

8.5 PROGRAMACIÓN DEL ROBOT

Existen dos formas básicas de programar la trayectoria en un robot manipulador, la programación Explícita y la programación Implícita.

La programación Explícita a su vez se divide en programación Gestual y programación Textual. En la programación Gestual se le enseña al robot, llevándolo a cada una de las posiciones de interés, guardándolas, definiendo la velocidad, la aceleración y generando manualmente los comandos asociados a estos puntos. A este tipo de programación también se le conoce como programación en línea, ya que muchas veces es necesario detener el proceso en el que actúa el robot manipulador para poder programarlo. Por otro lado para la programación Textual se realizan cuidadosamente los cálculos de la cinemática para generar la trayectoria, considerando las posiciones, las velocidades, las aceleraciones y el espacio de trabajo, para finalmente escribir los programas necesarios para el robot y ejecutarlos, esperando que no se haya cometido ningún error y el robot realice correctamente su tarea. A este tipo de programación también es conocido como programación fuera de línea ya que no es necesario detener el proceso en el que actuará el manipulador.

La programación Implícita, es cuando el robot cuenta con tareas predefinidas y con ayuda de un sistema de sensores y programación adecuados, hace parecer inteligencia en el robot. Esta programación cae dentro de la inteligencia artificial y es lo más avanzado en cuanto a programación de manipuladores se refiere.

El tipo de programación que se está manejando para nuestro caso cae dentro de la programación textual explícita, ya que se realizan todos los cálculos fuera de línea, es decir no es necesario disponer del robot físicamente para programar su tarea, con la ventaja de que el operador no tiene que realizar ningún cálculo ni escribir ninguna línea de código para programarlo. Esto es, todos los cálculos de cinemática se generan automáticamente a partir de la imagen adquirida, comprobándolos en la simulación en 3-D para tener la certeza de que el robot realizará correctamente su tarea.

Una vez que se tienen los resultados adecuados, y se comprueban con la simulación, se generan automáticamente los programas que representan la tarea para el robot manipulador, uno que representa el código (instrucciones y comandos) y otro que representa a los puntos que corresponden a las posiciones de interés para la tarea, en este caso los puntos por donde pasa la trayectoria. Los programas generados se describen a continuación:

- ***Trayectoria.mrl*** que representa el código [12], [13]:

```
10 SP 20          ' Ajusta velocidad al 66.67% (30~100%)
20 MO 1           ' Movimiento libre a la posición inicial del robot (1)
30 SP 15          ' Ajusta velocidad al 50%
40 MO 2           ' Movimiento libre al punto inicial de la trayectoria (2)
50 MR 2,3,4       ' Interpola circularmente los puntos (2, 3 y 4)
60 MR 4,5,6       ' ...
70 MR 6,7,8       ' ...
80 MR 8,9,2       ' ...
90 SP 20          ' Ajusta velocidad al 66.67%
100 MO 1          ' Movimiento libre a la posición inicial del robot (1)
110 ED            ' Representa el fin del programa
```

Donde los comandos MO y MR representan movimientos. MO lleva al robot manipulador de su posición actual a la posición indicada por el número asociado al comando, siguiendo una interpolación de tipo Joint y MR lleva al robot de su posición actual a la posición indicada por el primer número asociado al comando utilizando interpolación lineal, una vez que está en posición el robot genera un arco que pasa por los tres puntos asociados al comando (MR 2,3,4), utilizando una interpolación circular. Los números asociados a los comandos representan posiciones en el espacio del robot y están determinados por el siguiente archivo;

- ***Trayectoria.pos*** que representa los puntos asociados al código [12], [13]:

No.	*Posición*	*Orientación*
PD 1	232.00, 0.00,550.00	0.00,90.00,R,A
PD 2	387.58,-28.51,200.00	0.00,90.00,R,A
PD 3	365.92,-17.79,200.00	0.00,90.00,R,A
PD 4	359.36, 2.65,200.00	0.00,90.00,R,A
PD 5	371.33, 22.85,200.00	0.00,90.00,R,A
PD 6	393.28, 28.53,200.00	0.00,90.00,R,A
PD 7	414.37, 16.85,200.00	0.00,90.00,R,A
PD 8	420.07,- 3.75,200.00	0.00,90.00,R,A
PD 9	407.24,-23.51,200.00	0.00,90.00,R,A

Donde PD 1 representa la posición inicial del robot y los siguiente PD N representan puntos por donde pasa la trayectoria. De la lista de posiciones se pueden observar principalmente dos cosas, primero que la trayectoria que sigue el robot es siempre a una altura constante determinada por z=200.00 mm y que la orientación de la herramienta no cambia en ningún momento durante el recorrido de la trayectoria.

8.6 PUBLICACIONES

Se han escrito tres artículos, de los cuales dos ya están publicados y el otro ya fue aceptado. Dos son a nivel internacional y uno nacional, los datos de estos artículos se muestran a continuación:

- Soto Cajiga Jorge Alberto, Vargas Soto José Emilio, Pedraza Ortega Jesús Carlos, ***"Generación de Trayectorias por Visión para un Robot Manipulador de 5 grados de libertad"*** , Cuarto Congreso Nacional de Mecatrónica, Noviembre del 2005 Ramos Arizpe Coahuila, ISBN 970-9702-01-7.

- J. A. Soto Cajiga, J. E. Vargas Soto, J. C. Pedraza Ortega, ***"Generación de Trayectorias para un Robot Manipulador Utilizando Procesamiento de Imágenes y Splines"***, Segundo Congreso Internacional de Ingeniería, Marzo del 2006 Querétaro Qro. ISBN 968-845-296-3.

- J. A. Soto, J. E. Vargas, J. C. Pedraza, ***"A New Trajectory Generation Method Using Vision for a Robot Manipulator"***, International Conference on Dynamics, Instrumentation and Control CDIC'06, Agosto del 2006 Querétaro Qro., (Aceptado).

8.7 TRABAJO FUTURO

Hasta el momento se han presentado resultados interesantes, sin embargo, como se presentó en las limitaciones, existen diferentes puntos que se pueden mejorar. A continuación se describen algunos de estos puntos.

Primero, para el desarrollo del proyecto se tomó la consideración de que sólo se pueden generar trayectorias compuestas por curvas suaves, lo que implica que el algoritmo no acepte el manejo de líneas rectas y mucho menos de aristas sobre el contorno, por lo que un punto interesante para aumentar el alcance del trabajo seria mejorar los algoritmos tomando en cuenta estas restricciones.

Otro punto que se puede mejorar es la discriminación de puntos que se está utilizando, algo referente al tema se mencionó en las secciones 5.3 y 7.2, donde se propuso utilizar alguna otra técnica, como puede ser la selección manual de puntos a consideración del usuario o la selección de puntos de acuerdo a algún parámetro que represente la complejidad del contorno. Por ejemplo, considerar la derivada del contorno de manera que si existen regiones donde los cambios de derivada no sean tan suaves considerar más puntos y donde los cambios sean muy suaves considerar pocos, procurando no perder la esencia del contorno.

Por otro lado, algo que sería interesante analizar en el proyecto, es la implementación del método a alguna aplicación real, considerando las restricciones que se tienen, principalmente las referentes a los resultados de los errores. Para así poder validar si la aplicación está dentro de las posibilidades del sistema, y de no ser así, buscar la forma de mejorar o adecuar el método en las etapas que sea posible, considerando desde la etapa de adquisición, pasando por todo el procesamiento y cálculos, y hasta la misma calibración del robot manipulador.

Por último, el desarrollo de este trabajo sirvió para analizar el diseño de simulaciones en 3-D enfocadas a la robótica, utilizando librerías de OpenGL. Librerías que son poco conocidas y utilizadas en México. Obteniendo un resultado muy bueno, ya que se comprobó cómo estas librerías son una herramienta muy poderosa para la implementación de simulaciones en 3-D, utilizando pocos recursos físicos de la computadora, facilitando la programación y sobre todo ahorrando tiempo de diseño. Estos resultados obtenidos de la simulación abren camino al diseño de software de simulación de robótica, específicamente al diseño de robots manipuladores, pretendiendo desarrollar software de programación y simulación de robots manipuladores para el entrenamiento y la enseñanza, especialmente en instituciones de nivel medio y superior que no tienen las posibilidades de adquirir un robot manipulador industrial.

En la figura 8.5, se muestran los avances en el software de simulación, el cual se desarrollo a la par con la tesis.

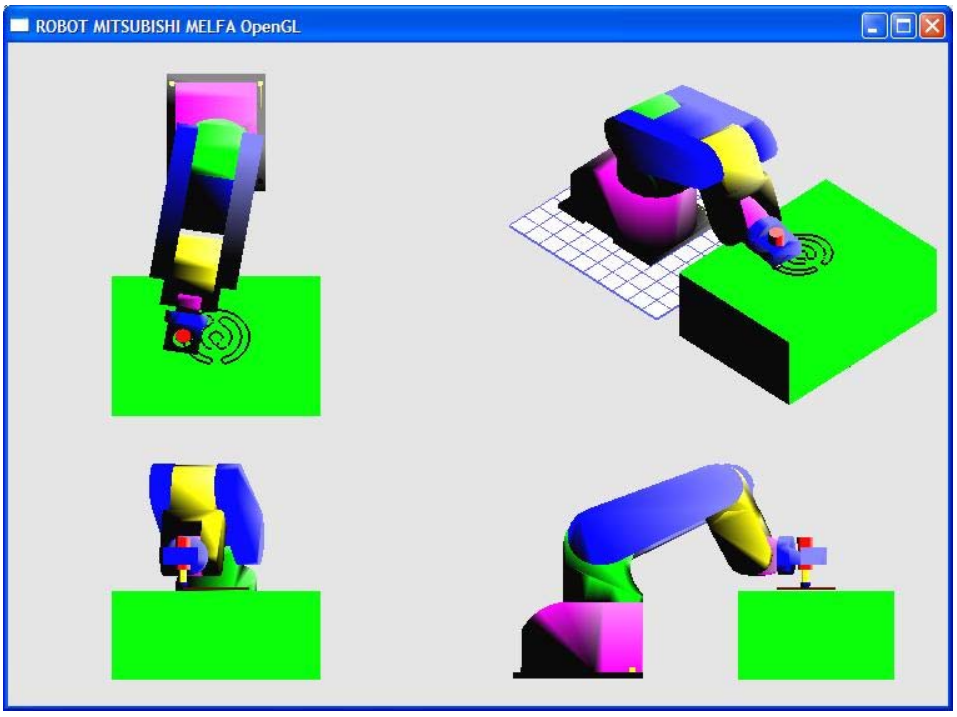


Figura 8.5.Software de simulación del robot utilizando librerías de OpenGL.

ANEXOS Y REFERENCIAS

ANEXO 1

INTRODUCCIÓN A OpenGL

OpenGL es una Interfase de programación de aplicaciones (API por sus siglas en inglés) estándar desarrollada por Silicon Graphics en 1992 [19].

En principio Silicon Graphics desarrollo una librería para sus estaciones graficas Iris llamada "Iris GL". Estas máquinas disponen de un hardware especialmente optimizado para visualización de gráficos, transformadas de matrices, soporte para Z-Buffer, etc. Con el uso de esta librería conseguían la independencia del hardware entre sus distintas estaciones Iris.

Fue en 1992 cuando Silicon Graphics presentó una librería llamada OpenGL, evolución de la antigua Iris GL. En su desarrollo pusieron especial énfasis en su portabilidad, posibilidades de expansión y por supuesto su rendimiento.

Al tratarse de una tecnología abierta, su especificación no debe estar controlada por un solo fabricante, sino que esta dirigida por un consorcio independiente, la plataforma de revisión de la arquitectura OpenGL (OpenGL Architecture Review Board) cuyos miembros fundadores son SGI, Digital Equipament Corporation, IBM, Intel y Microsoft. Actualmente 3Dfx, 3DLabs, ATI, Evans & Sutherland, Hewlett-Packard, NVidia y Sun también son miembros.

¿Porqué Utilizar OpenGL?

Durante años OpenGL se ha consolidado como la librería por excelencia para desarrollar aplicaciones 2D y 3D con independencia de la plataforma o el hardware gráfico.

Una de las principales ventajas que aporta OpenGL es que se trata de un estándar industrial. Gracias a la OpenGL ARB es realmente una tecnología abierta, lo que supone una ventaja inestimable frente a otras tecnologías.

Por otra parte cuenta con más de 13 años de vida, por lo que existe una extensa base de conocimientos a su alrededor. Durante todo este tiempo se han producido cambios en la librería, pero OpenGL siempre asegura una compatibilidad "marcha atrás". De esta forma, una aplicación que se desarrolló usando la primera implementación de la librería, compilaría y funcionaría con la última versión de la misma.

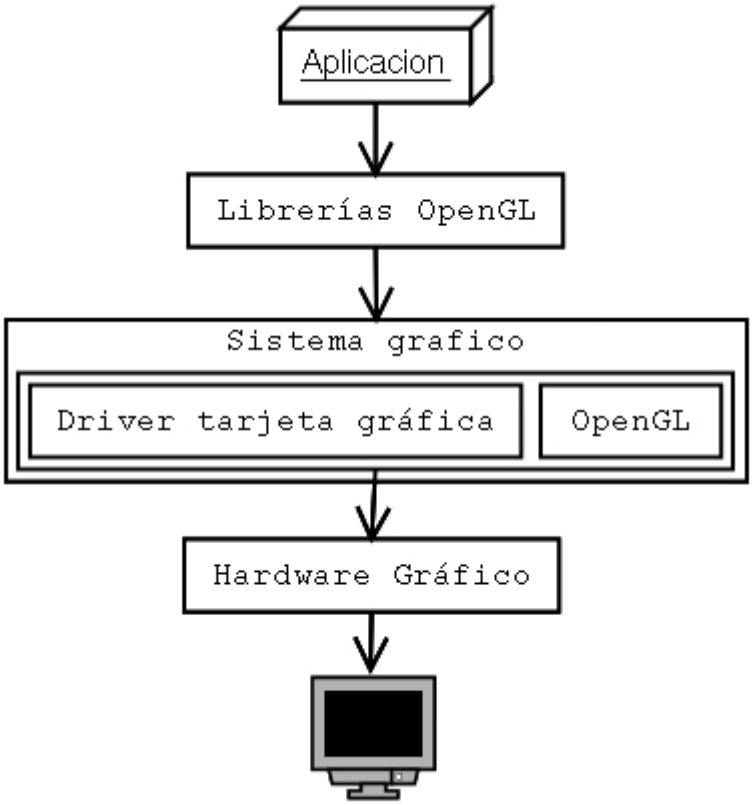


Figura A1.1, Sistema con OpenGL.

Gracias a la portabilidad de OpenGL nuestras aplicaciones podrán ejecutarse en una amplia variedad de arquitecturas y de soportes gráficos, sin que el resultado se vuelva inconsistente. Por ejemplo, nuestro código fuente va a ser el mismo para una máquina Sparc corriendo Linux, que para un PC corriendo Windows y usando una aceleradora gráfica. Este punto dota a nuestras aplicaciones de una gran escalabilidad. Actualmente OpenGL esta disponible para una gran variedad de sistemas operativos, tales como Unix, Windows, Mac OS, etc [1], [2].

Debido a los cambios en el hardware gráfico, y a que OpenGL es básicamente una interfase de abstracción del hardware, parece muy probable que se quede anticuado pronto, es decir, podríamos suponer que en cuanto aparezca una nueva prestación (por ejemplo en los aceleradores) OpenGL no sería capaz de proporcionar unas funciones al programador para que este pudiese hacer uso de estas nuevas características en sus programas. Pues bien, OpenGL se diseñó desde el principio para ser capaz de hacer frente a este problema, y gracias a sus mecanismos de extensión, las nuevas funcionalidades se pueden ir introduciendo sin problemas mientras que se respeta la compatibilidad con las versiones anteriores.

ANEXO 2

CALIBRACIÓN DEL ROBOT

En el transcurso del trabajo, específicamente en la etapa final, al momento de ejecutar la trayectoria con el robot manipulador se descubrió que el robot tiene un problema de calibración.

El problema consiste en que el robot realiza la trayectoria programada fuera de dimensiones, por ejemplo, se le programaron directamente con la computadora un par de figuras conocidas; un cuadrado de ancho 130mm y un circulo de diámetro 130mm, y el resultado fue una distorsión en dimensiones, sobre el eje *Y* principalmente. En la figura A2.1 se ilustra este ejemplo de distorsión.

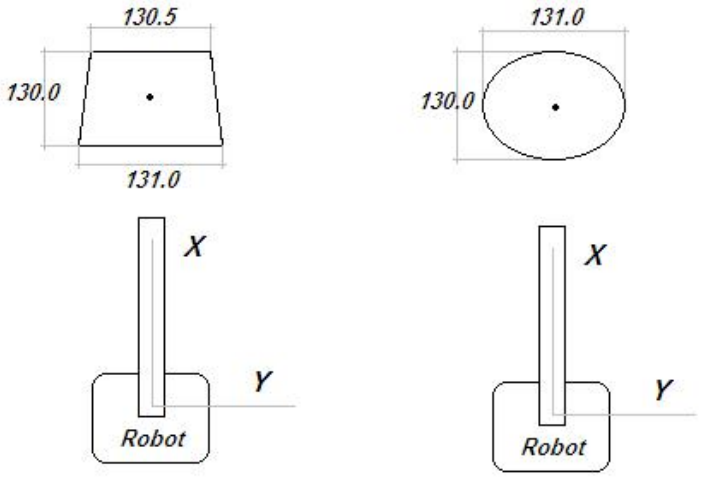


Figura A2.1. Ejemplo de distorsión de trayectorias.

Cabe señalar que el robot se calibró de dos diferentes formas según los manuales, esperando resolver el problema, pero ninguno de estos métodos de calibración funcionó, por otro lado el único método de calibración que no se pudo realizar fue la mecánica por cuestiones de integridad del robot.

Por lo que para resolver el problema se optó por implementar una técnica de calibración por software, que consiste en realizar una linealización en función de las desviaciones de los ejes *X* e *Y*. El procedimiento de esta calibración se muestra a continuación:

- Primero se le programa al robot una trayectoria paramétricamente conocida, procurando utilizar la mayoría del espacio de trabajo donde el robot ejecutará su tarea, en la figura A2.2 se ilustra la trayectoria que en teoría se le programó al robot.

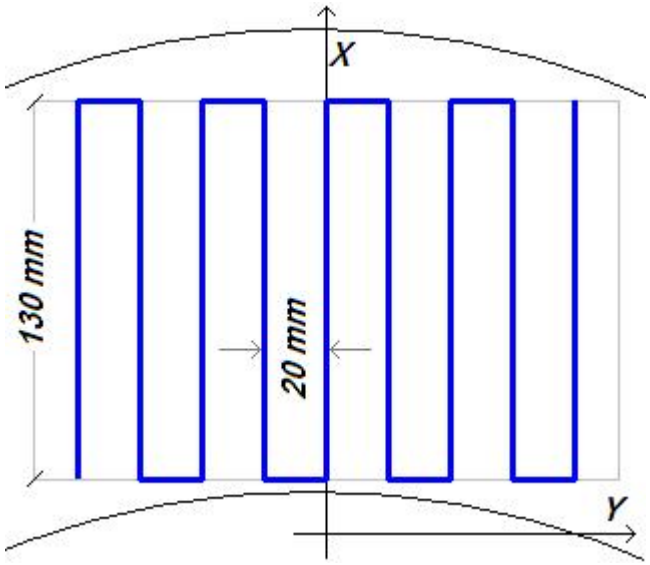


Figura A2.2. Trayectoria patrón para calibración.

- El siguiente punto es ejecutar el programa en el robot y capturar la trayectoria real que ejecuta el robot para parametrizarla, en la figura A2.3 se ilustra el resultado de la ejecución del robot.

- Posteriormente de parametrizar la ejecución del robot, se hacen un par de suposiciones. En base a la figura A2.3 se observa que cuando x=450 mm se tienen valores de separación de líneas que van de 20.9 a 21.4 mm sin ningún patrón, entonces la primera suposición es que la separación de líneas es constante y viene dada por el promedio de todos los valores. Esto es $xs(450)$=21.125 mm. Lo mismo se hace para cuando x=320 mm obteniendo una $xs(320)$=21.5 mm.

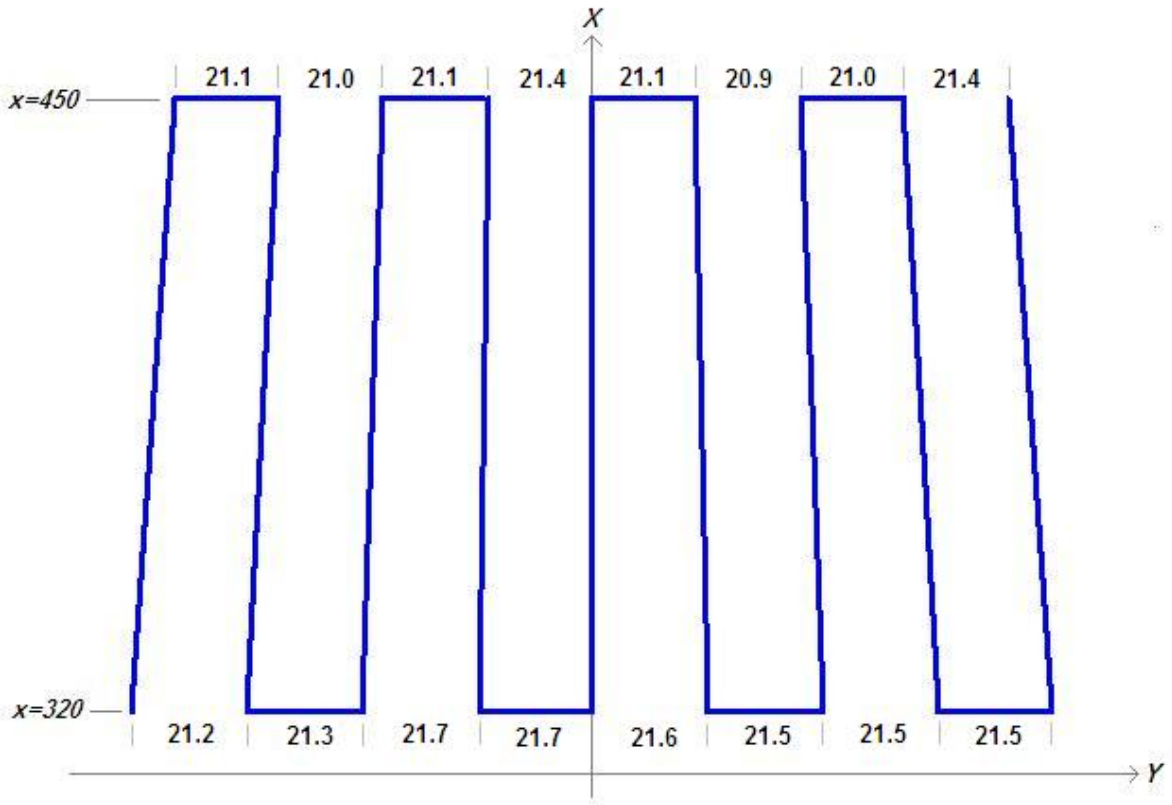


Figura A2.3.Representación gráfica de la ejecución del robot.

- El siguiente paso es linealizar. Primero de la figura A2.3 se observa que para los valores de x no hay ningún problema, el problema se presenta en el eje y, es decir, cuando la trayectoria se aleja de y=0, ya que ésta presenta una especie de escalamiento que es diferente para cada valor de x. De estas consideraciones se define el sistema de ecuaciones que se debe resolver para ajustar las desviaciones:

$$x = x \tag{A2.1}$$

$$y = y - (fac)(y) \tag{A2.2}$$

Donde fac es un factor de corrección en función de x y viene dado por la ecuación de la recta,

$$fac = mx + b \tag{A2.3}$$

Donde,
$$m = \frac{fac_2 - fac_1}{x_2 - x_1} \tag{A2.4}$$

Y fac_1 y fac_2 corresponden a los factores de corrección para x_1=320 y x_2=450 respectivamente, que corresponden a la relación entre las medidas de diseño (figura A2.2) y las medidas reales (figura A2.3),

$$fac_1 = \frac{172 - 160}{160} = 0.075$$

$$fac_2 = \frac{169 - 160}{160} = 0.05625$$

y finalmente *b* se obtiene de,

$$b = fac_1 - mx_1 \quad \text{ó} \quad b = fac_2 - mx_2 \tag{A2.5}$$

quedando por último las siguientes formulas,

$$fac = \frac{0.05625 - 0.075}{450 - 320}x + 0.075 - \frac{0.05625 - 0.075}{450 - 320}(320) \tag{A2.6}$$

$$y = y - (fac)(y) \tag{A2.7}$$

$$x = x \tag{A2.8}$$

Evaluando estas tres últimas formulas se obtienen los valores corregidos para *x* e *y*, y como se observó, esto es sólo una aproximación y no es una calibración adecuada, que sin embargo, fue necesaria implementar debido a las circunstancias ya mencionadas y que trabajó de una buena manera teniendo resultados de ejecución muy buenos, como se mostró en la sección 7.3.

REFERENCIAS

Bibliográficas

1. Neider J. y Davis T., *"ReedBook"*, Addison Wesley, Segunda Edición, USA, 1997.

2. Wright R. y Sweet M., "*OpenGL Super Bible"*, Waite Group Press, Segunda Edición, USA, 1999.

3. Pajares G. y Cruz J., *"Visión por Computadora"*, Alfaomega, España, 2002.

4. Gonzalez R. y Woods R., *"Digital Image Processing"*, Addison-Wesley, USA, 1992.

5. Otsu N., *"A Threshold Selection Method for Gray Level Histograms"*. IEEE Transactions on System, Man and Cybernetics. January, 1979.

6. Ferman H., *"Boundary encoding and processing*, in *Picture Processing and Psychopictorics"*, B.S. Lipkin and A. Rosenfeld, Editors. Academic Press: New York. p. 241-266, 1970.

7. Burden R. y Faires J., *"Análisis Numérico"* Thomson Learning, Séptima Edición, USA, 2001.

8. Barrientos A. y Balaguer C., "*Fundamentos de Robótica"*, Mc Graw Hill, Primera Edición, España, 1997.

9. Myler H. y Weeks A., *"The Pocket Handbook of Image Processing Algorithms in C*", Prentice Hall, 1993.

10. Pedraza J., *"Image Processing for Real World Representation Using Depth From Focus Criteria"*, Universidad de Tsukuba, Japón, 2002.

11. Bai Y., *"Applications Interface Programming Using Multiple Languages"*, Prentice Hall, USA, 2003.

12. Mitsubishi Industrial Robot, CR1/CR2/CR2A Controller INSTRUCTION MANUAL, Explanations of MOVEMASTER COMMANDS.

13. COSIROP 2.0, First Steps, Programming Software for Mitsubishi Industrial Robots.

14. Vargas E., *"Robótica, Un Reto para la Ciencia y la Convivencia"* , Revista CIENCIA Y DESARROLLO, Consejo Nacional de Ciencia y Tecnología – CONACYT, Volumen 30, p.p. 32 - 35, Julio 2005, México. ISBN: 0185-0008.

15. Christophe Collewet, *"A contour approach for image-based control on objects with complex shape"*, Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems.

16. Patrick Lim, *"Application of Shared Telerobotic Control for Contour Following Processes"*, Australian Conference on Robotics and Automation, Auckland November2002.

17. M. Charlebois, *"Shape Description of General, Curved Surface Using Tactile Sensing an Surface Normal Information"*, Experimental Robotics Lab. Simon Fraser University School of Engineering Science, Burnaby, B.C. V5A LS6, Canada.

18. Johan Baeten, *"Hybrid Vision/Force Control at Corners in Planar Robotic-Contour Following"*, IEEE/ASME Transactions on Mechatronics, Vol. 7, No. 2, June 2002.

Internet

19. http://www.opengl.org/, OpenGL, marca registrada de Silicon Graphics, Inc, Noviembre del 2005.

20. http://mathworld.wolfram.com/Circle.html, Weisstein, Eric W. "Circle." From MathWorld, A Wolfram Web Resource, Enero del 2006.

21. http://www.yorobot.com/, Ediciones Nowtilus, Febrero del 2006.