Centro de Investigación en Matemáticas, A.C.

Simulación de Jugadores de Air-hockey con Control Inteligente

T E S I S

que para obtener el grado de

Maestro en Ciencias con Especialidad en Computación y Matemáticas Industriales

presenta

Dan-El Neil Vila Rosado

Director de Tesis

Dr. Rogelio Hasimoto Beltrán
Dr. Axel Domínguez López

Guanajuato, Gto. *Agosto de 2007*

Simulación de jugadores de air-hockey con control inteligente

Dan-El Neil Vila Rosado
Centro de Investigación en Matemáticas. CIMAT, A.C.
dnvr30@cimat.mx

Agosto de 2007

Dedicatoria

A mis padres, hermana e hija pues son mi inspiración.

Agradecimientos

Al CONACYT por su apoyo económico

y al CIMAT por la formación que me han dado

Resumen

Jugar air-hockey es entretenido, pero simular el comportamiento de 2 jugadores en la computadora lo es más. Este es el propósito de esta tesis; simular el comportamiento de un portero y de un brazo robótico para que jueguen air-hockey. Se trabaja el diseño de un robot manipulador con un gripper cuyo comportamiento esta modelado por un sistema difuso y optimizamos los manipuladores en base a diversas características físicas y de trabajo. Para un mejor desempeño aportamos la información captada y procesada por un sistema de visión al sistema de control de los robots, quien a través de un modelo difuso toma las decisiones de defensa y ataque; es de mencionar que se compara contra las decisiones de un modelo neurodifuso. Para todo esto, se da como aportación una nueva herramienta en MATLAB para simulación de robots manipuladores que ya incluye sistemas de control difusos para el gripper (jerárquico) entre otras características únicas, además de que se combina los robots simulados con una nueva estrategia de juego de air-hockey basado en un sistema neurodifuso y otro difuso. Como resultados pudimos observar lo fácil que es modelar un manipulador con nuestra herramienta de MATLAB, además de que mostramos mejores resultados al utilizar un control difuso jerárquico contra un control difuso. Los resultados de la optimización nos arrojan que tenemos 2 modelos que son óptimos para nuestra tarea, ambos planares, de 2 y 3 DOF. En cuanto al comportamiento de los jugadores tenemos mejores comportamientos para lo sistemas difusos implementados, teniendo en cuenta que el neurodifuso presenta mas ventajas al ser adaptativo.

Índice general

Índice de figuras

Capítulo 1

Introducción

La historia de la automatización tiene poco tiempo, pero esta caracterizada por cambios bruscos y acelerados. Esta evolución tan rápida podemos atribuirla a que los humanos siembre buscamos entendernos y el autoconocimiento a nivel físico, emocional y espiritual siempre tendrá estudiantes y alumnos de aquí que conocernos es algo básico para poder vivir y es precisamente la robótica la ciencia que agrupa todas las temáticas que tratan de imitar los aspectos de la función humana mediante mecanismos, sensores, algoritmos, computadoras, etc.

Esta tesis se realiza para tratar de imitar el comportamiento de un jugador de air hockey. La motivación de este trabajo fue a partir del trabajo realizado para desarrollar un robot jugador de tennis [**?**] en 1989.

Crear este simulador es la base para armarlo físicamente y posiblemente incluirlo en alguna competencia de robots jugadores de air-hockey, pero su creación también incrementa los conocimientos dentro del campo de la robótica pues aportamos un nuevo sistema de control, así como nuevas herramientas de diseño y optimización de robots manipuladores.

1.1. Objetivo

El juego de air hockey es un juego de mesa de lo mas interesante en el que 2 jugadores tienen una paleta y con ella tratan de pegarle a un puck que se mueve en una mesa de aire, con tal de que entre en la portería del oponente.

Nuestro objetivo dentro de esta tesis los podemos dividir en partes:

1 : Desarrollar todas las herramientas necesarias para la simulación correcta de un robot manipulador.

2 : Optimizar el modelo de robot manipulador en base a ciertas características deseadas de un jugador de air hockey.

3 : Capturar la información real del movimiento de un puck en una mesa de air-hockey y predecir su comportamiento a través de visión por computadora.

4 : En base a la información recibida del movimiento del puck, tomar decisiones de juego con tal de que nuestro robot manipulador (brazo robótico) pueda defender, atacar o esperar al momento oportuno y en el caso de nuestro portero, que este pueda tomar la posición adecuada de defensa.

En esta tesis se diseña un robot manipulador en base a ciertas características físicas, dinámicas, de configuración o de restricciones y se simula su comportamiento ante información real de un juego de air-hockey con tal de que funcione como el brazo de un jugador y trate de anotar un gol en la portería contraria; así también, se simula el control de un robot portero cuya función es evitar un gol en su portería.

El proceso de todo este sistema se basa en que una cámara CCD captura las imágenes de un puck en una mesa de air-hockey y a través de técnicas de visión artificial, una PC procesa las imágenes para identificar la posición del puck y estimar su velocidad y trayectoria con tal de que en nuestra simulación el brazo robótico golpee el puck. Simultaneamente a la situación anterior, un robot portero simulado tratará de evitar un gol en su portería basado en la predicción del movimiento del puck.

Los movimientos del brazo robot en cuanto al golpeo del puck son controlados por técnicas de lógica difusa y se compara con técnicas de sistemas neurodifusos, mientras que las del portero tiene una heurística.

Como experimentos finales mostramos las respuestas de ambos robots ante distintas situaciones de velocidad del puck y de restricciones de diseño de los mismos robots.

1.2. Organización de la tesis

Para llevar a cabo nuestro objetivo se desarrollaron los siguientes capítulos:

El segundo capítulo está enfocado a dar una introducción a la **Robótica**; nos da los primeros aspectos a saber de esta ciencia, desde el origen del concepto de robot,

las clasificaciones de los robots y a grandes rasgos el funcionamiento general de un robot.

Como tercer capítulo tenemos la información necesaria para poder realizar **Diseño de robots manipuladores**, pues durante este capítulo construimos el modelo y pasamos a evaluarlo en términos de cinemática directa, la viabilidad de su cinemática inversa, generamos diversos tipos de trayectoria viables en los cuales evaluar el comportamiento de nuestro robot y finalmente simulamos un elemento de agarre para nuestro robot manipulador con 2 técnicas de control: un modelo difuso y un control jerárquico difuso; ambos sistemas están enfocados en mantener a un objeto sostenido por el gripper de nuestro robot sin resbalarse y sin que lo destruyamos. Con esto se verificará el funcionamiento adecuado de nuestro brazo robótico que fungirá como jugador de air-hockey.

La simulación nos permite verificar el comportamiento de nuestros sistemas antes de elaborarlos físicamente, pero dado un modelo siempre es necesario buscar su optimización; es por eso que en el capítulo cuatro realizamos **Optimización de diseño para robots manipuladores**. Dentro de los métodos de optimización trabajados están la de síntesis de mecanismos donde buscamos configuraciones de robots viables (morfología y posición) para seguir una trayectoria entre obstáculos. Una mejora para los modelos de robots manipuladores es reducir el número de grados de libertad (DOF) por lo que también optimizamos basándonos en esta característica. Finalmente optimizamos teniendo en cuenta una tarea a realizar con restricciones de cinemática y estructura (longitud de eslabones).

Teniendo un robot manipulador adecuado, tenemos que proporcionarle la información de su ambiente de trabajo; en este caso, la tabla sobre la que se juega air-hockey y todo lo que acontece ahí. Es por esta situación que incluimos el capítulo cinco de **Sistema de Visión**, donde describimos como se lleva a cabo la captura de video, su procesamiento y la estimación de movimiento, información que es enviada al sistema de control de nuestro robot para que tome decisiones de juego.

Como habíamos mencionado, el sistema de control toma las decisiones en base a la información proporcionada por el sistema de visión. Para términos de esta tesis se recurre a un **Control inteligente**; desarrollado en el capítulo seis e implementado en una PC, comparamos 2 métodos: uno difuso y otro neurodifuso que fue quien presento mejores resultados al ser un tipo de control adaptativo a diferencia del difuso.

En el capítulo siete, con todas las herramientas desarrolladas en los capítulos anteriores armamos el **Simulador**. Mostramos la estructura del hardware, el área de trabajo del manipulador y la implementación de los 2 robots (el brazo y el portero) así como las **Conclusiones** de las pruebas del simulador, mientras que el **Trabajo a futuro** es presentado en el capítulo 8.

1.3. Contribuciones

Los robots tienen muchas aplicaciones y ya hay herramientas que permiten realizar los análisis respectivos; sin embargo, la mayoría son incompletos, tienen costos altos o no permiten una participación activa del usuario. El toolbox de MATLAB para robótica desarrollado como objetivo de esta tesis es versátil e implementa lo necesario para un diseño eficiente, así como también es una herramienta para evaluar el desempeño del modelo. La publicación de este toolbox es prueba de su aceptación.

* : D. N. Vila Rosado and J. A. Dominguez Lopez. *A MATLAB toolbox for robotic manipulators* Computer Science, 2005. ENC '05. Sixth Mexican International Conference on. pp. 256-263. September 2005. Puebla, México.

Otra contribución que se aportó con este toolbox de MATLAB fue el incorporar técnicas de optimización evolutiva con tal de mejorar los diseños hechos.

* : D. N. Vila Rosado and J. A. Dominguez Lopez. *A MATLAB toolbox for the optimal design of robot manipulators using evolutionary techniques* 10th International Conference in Mechatronics Technology. November 2006. Mexico City, Mexico.

El sistema de control jerárquico difuso del gripper fue otra de las contribuciones aceptadas como artículo.

* : J. A. Dominguez Lopez and D. N. Vila Rosado *Hierarchical fuzzy control to ensure stable grasping* Computer Science, 2006. ENC '06. Seventh Mexican International Conference on. pp. 37-43. September 2006. San Luis Potosi, México.

Otra contribución fue el sistema neurodifuso empleado en el sistema de control para tomar la decisión de juego del robot manipulador en un sistema de air-hockey, pues hasta la actualidad para esta actividad solo se han trabajado modelos de ecuaciones diferenciales, PID y sistemas difusos. Cabe mencionar que el sistema neurodifuso presentó varias ventajas sobre el que se considera hasta ahora el mejor, el sistema difuso.

Capítulo 2

Robótica

En la robótica convergen distintas y muy diversas áreas de estudio; entender a grandes rasgos lo que es un robot y como funciona es el objetivo de este capítulo. En la sección 2.1 veremos las distintas definiciones de robot y como surgió este concepto. En la sección 2.2 daremos un pequeño panorama de esta área de la ciencia. Continuamos con la sección 2.3 donde daremos una breve explicación de como se pueden clasificar los robots actuales para finalmente en la sección 2.4 describir los componentes de un robot manipulador y como funcionan.

2.1. Definición de Robot

El término robot procede de la palabra checa "robota", que significa "servidumbre o esclavitud". Cuando Karel Capek (1890-1938) utilizó la ciencia ficción para filosofar acerca del futuro de la humanidad en la tierra, también introdujo la palabra "robot". Efectivamente, es en 1917 cuando en su libro R.U.R (Robots Universales Rossum) hace referencia a un concepto que aun hoy en día esta en discusión, sobretodo cuando se habla de lo que es un robot.

Es de mencionar que la robótica (ciencia o rama de la tecnología, que estudia el diseño y construcción de robots) tal y como se conoce en la actualidad, surge muchos años después de la obra de Capek y no se reduce a entender el concepto de robot como un humanoide automatizado sino que el concepto de robot es mucho más amplio.

La Enciclopedia Británica dice: *"Un dispositivo robot es un mecanismo instrumentado que se usa en la ciencia e industria para sustituir al ser humano. No tiene por qué asemejarse físicamente a un ser humano ni tiene por qué realizar sus tareas de un modo humano"*, por otra parte el Diccionario de la Lengua Española de la Real

Academia define robot como *"Ingenio electrónico que puede ejecutar automáticamente operaciones o movimientos muy varios"*.

Con el tipo de definiciones anteriores, es difícil distinguir entre lo que es un dispositivo robot y lo que es una simple maquinaria automatizada pues gran parte de las máquinas actuales se pueden entender como robots sin serlo. Por ejemplo, cualquier maquina doméstica (lavadora, podadora, etc.) puede entenderse como robot ya que sustituye al ser humano en muchas tareas cotidianas (lavar, podar, etc.).

Una definición más amplia del concepto de robot extraída del Instituto Norteamericano del robot y de la ISO 8373, y que limitaría esta ambigüedad entre máquina y robot es:

Definición 2.1.1 *Un robot es un manipulador reprogramable, multifuncional, controlado automáticamente, que puede estar fijo en un sitio o moverse, y que esta diseñado para mover materiales, piezas, herramientas o dispositivos especiales, por medio de movimientos variables programados para la realización de diversas tareas o trabajos.*

En la definición anterior encontramos los conceptos manipulador, reprogramable y multifuncional, los cuales pueden entenderse de la siguiente manera:

Manipulador : Mecanismo que consiste en un conjunto de segmentos y uniones para mover objetos normalmente en varios grados de libertad.

Reprogramable : Concepto que se refiere a que los movimientos programados o funciones auxiliares pueden modificarse sin que se realicen alteraciones en la estructura mecánica o en el sistema de control, excepto aquellas que suponen cambios de programas y memorias.

Multifuncional : Mecanismo que es posible de adaptar a diferentes aplicaciones con alteraciones en la estructura mecánica o en el sistema de control.

Poniendo un poco de atención en la definición 2.1.1 podemos notar que no se considera vehículos mobiles (terrestres, marinos o aéreos) reprogramables que bien podrían ser considerados como robots, ante esto, otra definición ampliamente aceptada en el mundo de la robótica es:

Definición 2.1.2 *Un robot es una máquina versátil y polivalente constituida por un sistema mecánico articulado y dotado de un sistema electrónico - informático programable, que incluye una gran variedad de dispositivos y sensores.*

2.2. La Robótica

La robótica hace uso de una gran cantidad de conocimientos (gráficos por computadora, cinemática, dinámica, control, mecánica, programación, etc.) de ahí que sea vista como "una ciencia multidisciplinar que estudia los robots".

El gran desarrollo de la robótica se debe a los avances y éxitos conseguidos en otras muchas ciencias distintas entre sí, muchas de ellas se pueden agrupar dentro de las áreas de conocimiento conocidas como informática y automática.

* La informática se encarga de lo concerniente al tratamiento de la información. Es decir, es la herramienta que utiliza la automática para tratar la información del sistema, tomar decisiones o comunicar la información a operadores humano o materiales.

* La automática tiene como objetivo asegurar el funcionamiento automático del sistema. Es decir, es la ciencia que realiza el control del sistema.

La robótica se puede ver como la ciencia que trabajando originalmente a partir de un mecanismo, incluye un automatismo y un equipo informático.

Si bien es cierto que un robot no puede sustituir a un ser humano en cualquier ámbito ya que las conductas y actividades humanas pueden tener un nivel de complejidad muy superior al sistema informático de cualquier robot; también es cierto que un robot puede trabajar ininterrumpidamente y además es capaz de realizar operaciones que un ser humano no puede realizar, ejemplo de esto son los trabajos que realizan los robots en condiciones adversas como manipulación de objetos en hornos industriales, explosivos, reparaciones de maquinas en el espacio, exploraciones y perforaciones submarinas o trabajos en ambientes tóxicos o radiactivos.

En resumidas cuentas, las ventajas de un robot esta en la capacidad de realizar de modo automático gran cantidad de actividades y en la gran productividad de la que es capaz.

2.3. Clasificación de robots

$$\text{Clasificación por actuadores} \begin{cases} \text{Robots manipuladores} \\ \text{Robots móviles} \begin{cases} \text{Terrestres} \\ \text{Acuáticos} \\ \text{Aéreos} \end{cases} \end{cases} \tag{2.3.1}$$

Existe una gran diversidad de formas para clasificar a los robots existentes, dentro de las más conocidas están la clasificación por actuadores (Diagrama 2.3.1), por capacidad de tarea ó área de aplicación (Diagrama 2.3.2) y una de las más intuitivas que podemos encontrar es la siguiente:

Humanoide : Robot con apariencia física humana que busca imitar el comportamiento de este.

Robot móvil : Robot montado sobre una plataforma móvil.

Robot industrial : Robot manipulador diseñado para mover materiales, herramientas o dispositivos especializados mediante movimientos variables programados para el desarrollo de diferentes tareas.

Robot inteligente : Robot capaz de trabajar y moverse en un entorno no estructurado y con eventos impredecibles. Este tipo de robots pretende hacer uso de la información procedente de sensores; es capaz de interactuar con un operario y dispone de capacidad de aprendizaje.

Robot de servicios : Robot que opera con total o parcial autonomía para desarrollar servicios útiles, excluyendo aquel que realiza operaciones de fabricación.

$$\text{Clasificación por tarea} \begin{cases} \text{Industriales} \\ \text{De prótesis} \\ \text{Didácticos} \\ \text{Caseros} \\ \text{De juguete} \\ \text{Científicos} \\ \text{Vehículos de control remoto} \end{cases} \tag{2.3.2}$$

Cuando hablamos de *robos industriales* estamos hablando de robots manipuladores, es decir, aquellos robots que poseen la estructura antropomórfica de un brazo

humano. Esta tésis se basa en el estudio y simulación de un robot manipulador en una mesa de air hockey, de aquí que ahondaremos en el tema de *robots manipuladores*. La *Federación Internacional de Robótica (IFR)* los clasifican según 3 parámetros: número de ejes, estructura mecánica y tipo de control.

1.- Clasificación según el numero de ejes o grados de libertad (DoF):

* Robots con 3 ejes.

* Robots con 4 ejes.

* Robots con 5 ejes o más.

2.- Clasificación por estructura mecánica:

* Cartesianos.

* SCARA.

* Antropomórficos.

* Paralelos.

* Esféricos y cilíndricos.

3.- Clasificación según el tipo de control:

* Secuencia controlada: Robot con sistema de control en el que los movimientos se llevan a cabo en un orden determinado.

* Trayectoria operada/continua: Robot que de acuerdo a especificaciones de trayectoria dadas y para alcanzar una posición en un área de trabajo, controla mediante algún procedimiento sus ejes mecánicos o grados de libertad.

* Adaptativos: Robot con un control cuyos parámetros de funcionamiento son ajustados a partir de las condiciones detectadas durante un proceso. Se dice que es adaptativo sensorial si los ajustes los lleva a cabo en base a la información que aportan sensores externos y se llama adaptativo mediante aprendizaje, cuando la experiencia obtenida se usa para cambiar los parámetros de control y/o algoritmos.

* Teleoperados: Robot que es operado de forma remota por un humano.

2.4. Componentes de un robot manipulador

Los componentes y subsistemas de un robot manipulador tienen funcionalidades similares a cada una de las partes que forman un brazo humano. Así como un brazo tiene una estructura (huesos, tendones, nervios y músculos), relaciones entre las diferentes partes (articulaciones), una mano y además cuenta con los sentidos (como el tacto) los cuales llevan la información del ambiente al cerebro a través de señales nerviosas; también un robot manipulador consta de una estructura mecánica compuesta por eslabones ("hueso"), accionadores ("músculo"), transmisiones ("tendones") y los cables de señal ("nervios"). En general, podemos decir que un robot manipulador esta conformado por 3 sistemas (Diagrama 2.4.3): mecánico, sensorial y el sistema de control.

$$\text{Subsistemas de robot manipulador} \begin{cases} \text{Mecánico} \\ \text{Sensorial} \\ \text{De control} \end{cases} \tag{2.4.3}$$

En un robot manipulador, los puntos de unión entre eslabones se llaman nodos y el elemento que permite la unión y el movimiento relativo entre ellos es la articulación (Figura 2.1). La capacidad de carga depende del tamaño y características estructurales de los eslabones, sistemas de transmisión y accionadores; todas estas características se pueden catalogar dentro del *sistema mecánico*. Dentro de este rubro también se encuentran los elementos terminales (pinzas, herramientas, ventosas, etc.) cuya funcionalidad es adaptarse de forma específica a la aplicación a realizar por el robot (Figura 2.1). Dicho elemento terminal se encuentra montado sobre la "muñeca" del robot, la cual permite orientar de forma adecuada en el espacio de trabajo el elemento terminal (Figura 2.1).

El *sistema sensorial* de un robot manipulador recoge la información necesaria, tanto para posicionar y orientar el robot en un espacio de trabajo, como para reconocer y poder actuar en estas circunstancias. Cabe mencionar que los sensores pueden ser internos o externos al robot.

La información recopilada por los sensores del robot es procesada y analizada por el *sistema de control* , este sistema hace la función de "cerebro" del robot y genera las ordenes necesarias y adecuadas que debe ejecutar el sistema mecánico para realizar una determinada tarea, en base a la información de los sensores.

A continuación veremos con más detalle las funcionalidades y características de cada sistema de un robot manipulador.

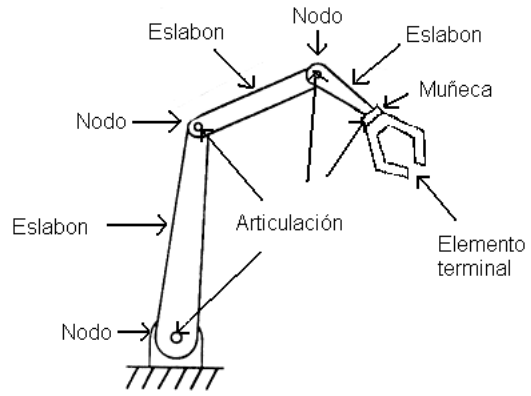


Figura 2.1: Diagrama de manipulador

2.4.1. Sistema mecánico

Para realizar movimientos en el espacio tridimensional un robot manipulador hace uso de la combinación de movimientos de traslación y rotación.

Un cuerpo libre en el espacio puede moverse en 3 direcciones independientes y perpendiculares entre sí, así como rotar alrededor de estas mismas direcciones; de aquí que se diga que tiene 6 grados de libertad (DoF).

Definición 2.4.1 *Los grados de libertad para un robot manipulador (DoF por sus siglas en inglés) son el número de componentes de movimiento simple (traslacional o rotacional) que se requieren para realizar movimientos más complejos.*

Si una articulación solo permite movimiento a lo largo de una linea recta o giro sobre 1 eje se dice que posee un grado de libertad, mientras que se dice que cuenta con 2 grados de libertad si puede moverse en un plano y con 3 si se mueve en el espacio 3-dimensional (Figura 2.2).

Es importante saber los grados de libertad necesarios para que nuestro robot manipulador realice determinada tarea(posición y orientación), pero lo es más, determinar el mínimo número necesario pues con eso tendríamos menos problemas de diseño y de consumo de energía (ver Capítulo 3).

Cuando se tiene más grados de libertad de los realmente necesarios para realizar una tarea se habla de un robot redundante. Se utiliza este tipo de robots en situaciones donde se complican las acciones a realizar como evitar un posible obstáculo (Figura 2.3) o aumentar su espacio de trabajo (Figura 2.4).

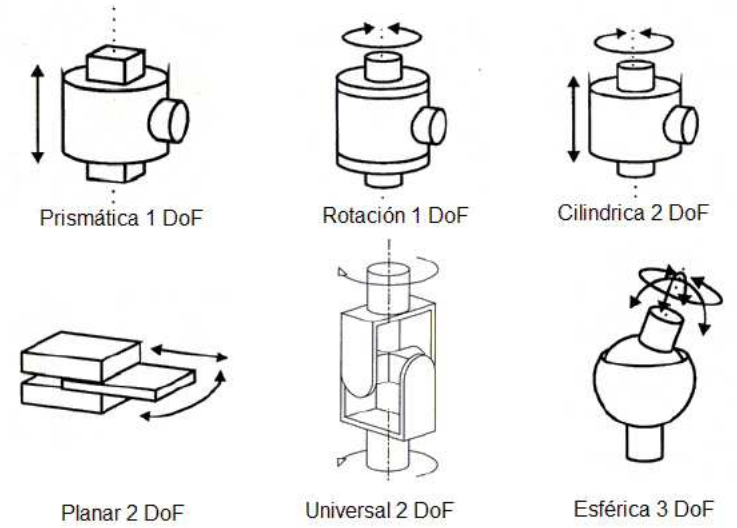


Figura 2.2: Articulaciones y sus grados de libertad (DoF)

Definición 2.4.2 *El espacio o volumen de trabajo es el espacio tridimensional que se es capaz de alcanzar por el extremo del robot.*

Nuestro primer objetivo en cuanto a la estructura del robot esta en poder situar el extremo del robot en una posición deseada; ante esto, utilizamos lo que a continuación describimos.

Primero buscamos que el robot se puede mover en el espacio de trabajo con bajo consumo de energía, alta velocidad de operación y precisión; de aquí que la elección adecuada de los materiales con los que hacemos al robot sea parte esencial del diseño y elaboración (ver Capítulo 3). Así, el sistema mecánico puede ser de tipo rígido o flexible; mientras que los de tipo rígido tienen gran resistencia y gran control de posición, los flexibles presentan bajo consumo de energía, alta velocidad de operación y alta taza de carga-peso.

Definición 2.4.3 *Un eslabón (o vínculo) en una cadena cinemática es un segmento rígido o flexible que une 2 nodos sucesivos o articulaciones.*

Los materiales utilizados para la construcción de los eslabones es muy diversa: madera, aleaciones metálicas, plásticos, gomas, etc. En general el eslabón debe ser lo más ligero posible además de tener características adicionales dependiendo de la actividad a realizar (resistente a la corrosión, no flamable, etc.).

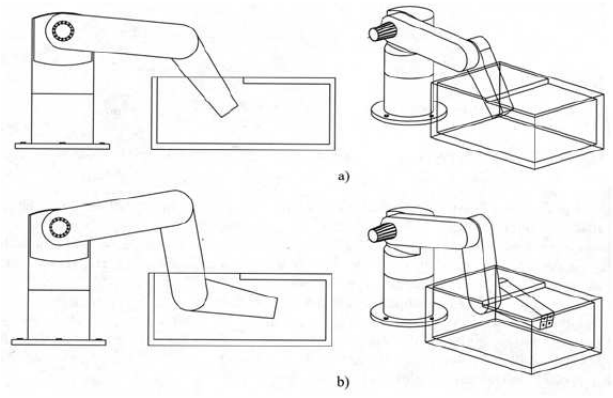


Figura 2.3: a) Robot con problema de alcance dentro de una caja. b) Robot redundante para evitar el obstáculo y tener máximo alcance en la caja

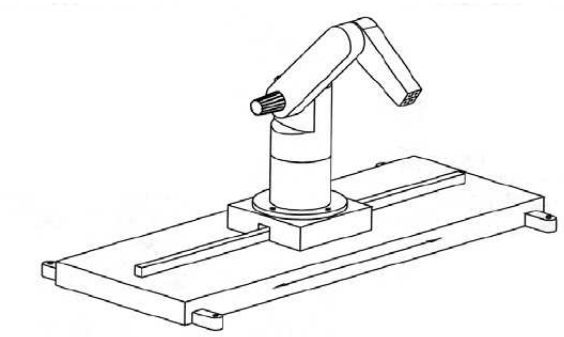
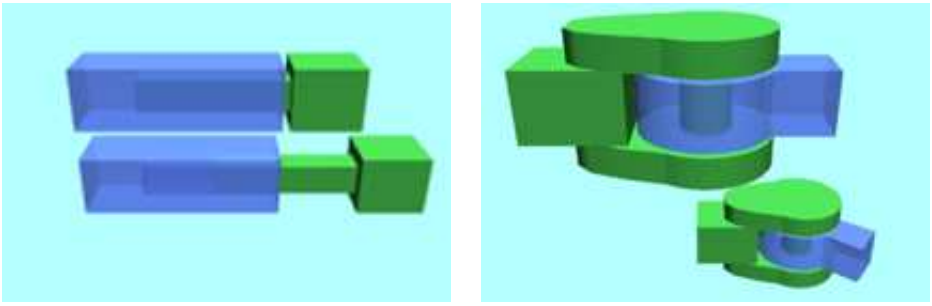


Figura 2.4: Robot redundante para amplificar el espacio de trabajo con ayuda de 1 DoF de traslación

Definición 2.4.4 *Una articulación es un elemento físico que une 2 eslabones entre sí, permitiendo un cierto movimiento relativo entre ellos.*

Los grados de libertad de una articulación son correspondientes al número de grados de libertad que permita el movimiento. Así, solo es posible 2 tipos de articulación de 1 DoF: los de traslación o de tipo prismático (desplazamiento de un eslabón con respecto a otro) mostrado en la figura 2.5(a) y los de rotación o de tipo rotacional (permiten un giro en torno a un eje de un eslabón con respeto a otro) como el de la figura 2.5(b).

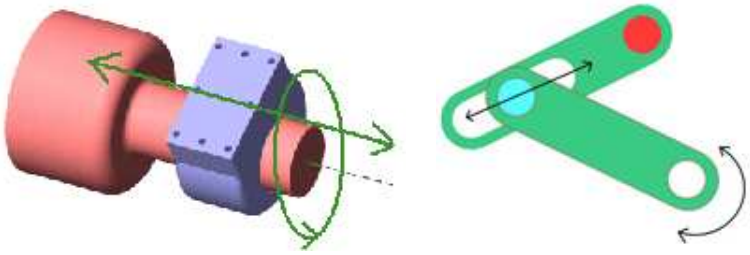
Dentro de las articulaciones que tienen 2 grados de libertad, las más conocidas son: la cilíndrica (Figura 2.6(a)) y la planar (Figura 2.6(b)). La junta universal o rotula es un caso especial de articulación con 2 DoF pues resulta de la combinación de



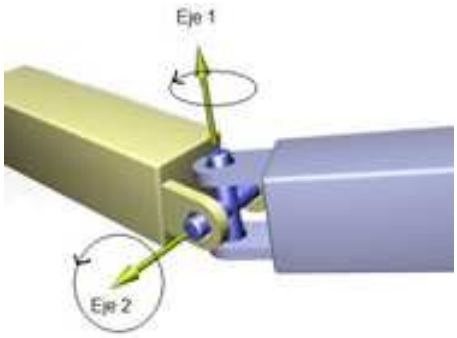
(a) Articulación prismática. (b) Articulación rotacional.

Figura 2.5: Articulaciones (a) prismática y (b) rotacional.

2 articulaciones rotacionales con ejes de giro perpendiculares de 1 DoF que no están en un único elemento físico (Figura 2.6(c)).



(a) Articulación cilindrica. (b) Articulación planar.



(c) Articulación universal.

Figura 2.6: Articulaciones (a) cilíndrica, (b) rotacional y c) universal

La articulación que en un solo elemento físico permite 3 giros independientes es la de tipo esférico (Figura 2.7). Los ejes de giro de esta articulación se cortan en un mismo punto, lo cual ofrece la ventaja de reducir cálculos cinemáticos.

Nuestro robot funciona como una cadena cinemática (secuencia de eslabones y articulaciones), de forma que los eslabones tienen movimientos relativos entre sí,

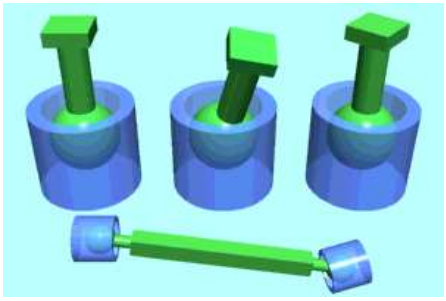


Figura 2.7: Articulación esférica

llevados a cabo por medio de las articulaciones que los unen (Figura 2.8). Con una misma cadena cinemática es posible obtener diferentes configuraciones de robots. La elección de la cadena cinemática y la configuración depende básicamente de la actividad a realizar y de la estética que se le quiera dar.

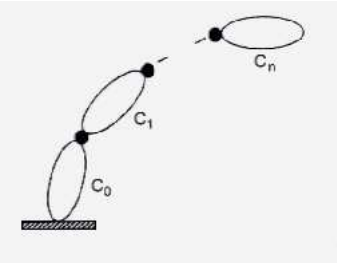


Figura 2.8: Cadena cinemática

Muchas cadenas cinemáticas de los robots manipuladores cuentan con 2 subcadenas; la primera está compuesta por articulaciones de 1 DoF y se emplea para que el extremo del robot llegue a la posición deseada del espacio de trabajo (brazo del robot). La segunda subcadena nos permite obtener la orientación deseada, ante esto, por lo general se recurre a una articulación esférica (muñeca del robot).

Las distintas combinaciones de configuración de cadenas cinemáticas nos da una clasificación de las mismas:

Robot cartesiano : Formado por 2 o 3 articulaciones prismáticas cuyos ejes son ortonormales entre sí. Este tipo de cadena aporta una buena precisión y velocidad constante en el espacio de trabajo. Se ocupa en actividades que ocupan movimientos lineales y grandes longitudes (Figura 2.9).

Robot cilíndrico : Llamado así por el tipo de espacio de trabajo que presenta, posee como base una articulación rotacional con eje paralelo a la segunda articulación prismática. La tercera articulación prismática tiene el elemento terminal (Figura 2.10).

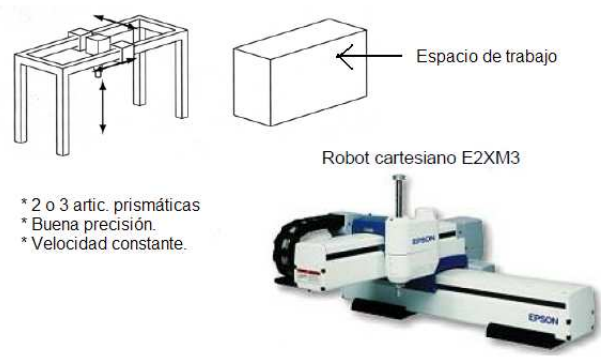


Figura 2.9: Robot cartesiano

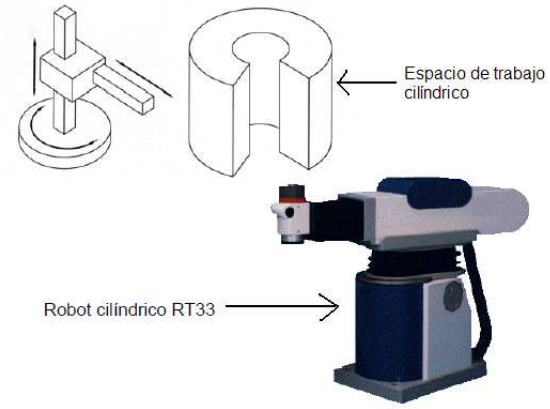


Figura 2.10: Robot cilíndrico

Robot esférico o polar : Variante del cilíndrico, en el que la segunda articulación es rotacional con eje perpendicular a la primera y el eje del prismático es perpendicular a las 2 rotacionales. El espacio de trabajo generado es un casquete esférico (Figura 2.11).

Robot SCARA : Tiene la misma secuencia de articulaciones que el esférico pero los 3 ejes de las articulaciones son paralelos entre sí; esto permite que el robot tenga alta rapidez y precisión. Son empleados por lo general como maquinas de ensamble (Figura 2.12).

Robot antropomórfico : También llamado angular. Es una variante del esférico en el que una articulación rotacional con eje paralelo a la intermedia ocupa el lugar de la última articulación, la prismática. Permite realizar trayectorias complejas, alta maniobrabilidad y accesibilidad a zonas con obstáculos (Figura 2.13).

Robot paralelo : Configuración de 6 grados de libertad donde la posición y orientación del efector final se controla a través del desplazamiento de 6 articula-

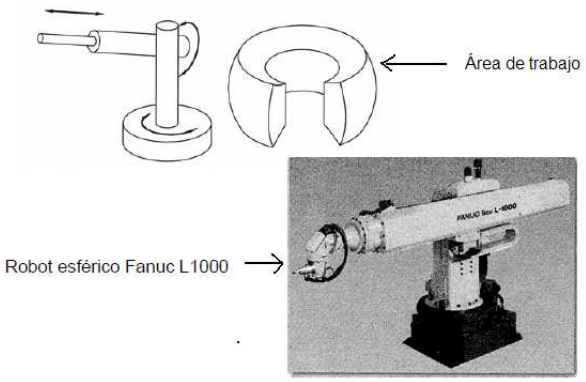


Figura 2.11: Robot esférico o polar

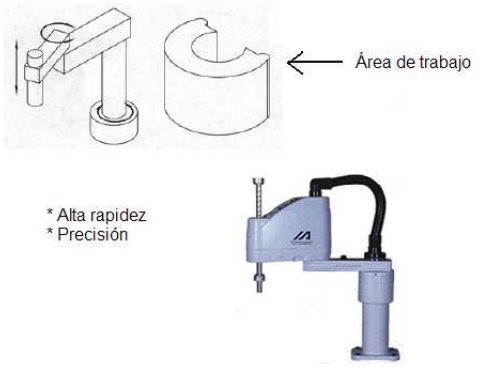


Figura 2.12: Robot SCARA

ciones prismáticas conectadas a la base del robot por una junta universal y al elemento terminal por una articulación esférica de 3 DoF (Figura 2.14).

Teniendo una configuración cinemática, eslabones y articulaciones elegidos, nos encontramos ahora con los accionadores quienes dan el movimiento a las articulaciones.

Definición 2.4.5 *Los accionadores son elementos encargados de traducir una señal de control (velocidad, posición, temperatura, viento, etc.) en acciones controladas de una máquina (movimiento, calentamiento, ensamblaje, embalaje, etc.).*

Los accionadores los podemos clasificar en función del tipo de energía que emplean: eléctrica, neumática o hidráulica. No considero que sea una muy buena idea utilizar un accionador eléctrico si estamos trabajando con gases explosivos por lo que suena

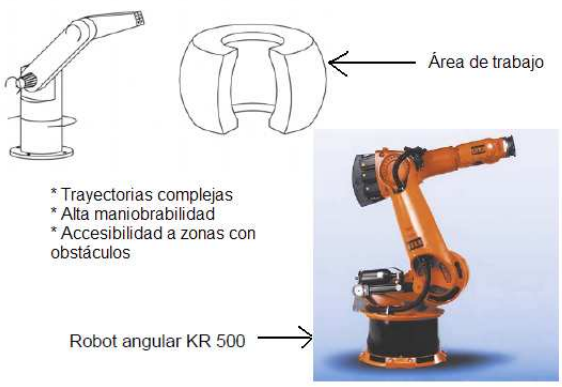


Figura 2.13: Robot antropomórfico o ángular

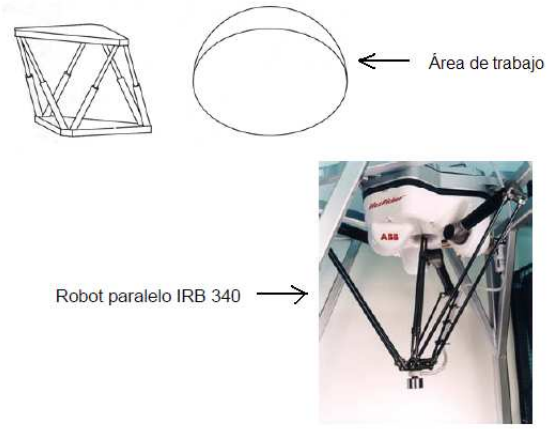


Figura 2.14: Robot paralelo

"más o menos lógico" elegir el tipo de accionador dependiendo de la actividad a realizar :) .

Accionadores eléctricos : Proporcionan mayor precision a comparación de los otros tipos de accionadores; así como también menor tamaño, peso, facilidad de conexión y bajo costo. Dentro de esta categoría encontramos conmutadores de tipo on-off (relé, diodo, transistor bipolar, etc.); solenoides o motores eléctricos (de corriente continua con/sin escobillas y motores paso a paso).

Accionadores neumáticos : Usan la energía del aire comprimido. Se utilizan cuando es necesaria gran potencia para efectuar la actividad. Son económicos, ofrecen altas velocidades de trabajo y no contaminan el área de trabajo con fluidos. Presentan gran complejidad en el control continuo y mala precisión. Se utilizan por lo general en los elementos terminales (pinzas, dedos, etc.).

Accionadores hidráulicos : Usan la energía de la presión de algún líquido. Funcionan suave a bajas velocidades, amplio rango de velocidades de funcionamiento sin sistemas de control adicionales, son autolubricados y operan en condición de paro sin sobrecalentamiento. Su defecto: Gran peligrosidad ante eventuales fugas de aceite o el líquido que este a presión, altos costos de instalación y problemas de miniaturización.

A pesar de los accionadores antes mencionados, existe la posibilidad de que el accionamiento directo no sea lo mejor, debido a las siguientes situaciones:

1 : Que la velocidad y el par no sean los adecuados para nuestros requerimientos, ante esto, es necesario adaptar el accionador mediante un dispositivo adicional, un reductor.

2 : Que el accionador sea pesado y/o voluminoso, lo cual aumenta la inercia del brazo del robot sobretodo en articulaciones más alejadas de la base. Esto se puede resolver poniendo la articulación cerca de la base y utilizar una transmisión que lleve el movimiento generado en el accionador hasta la articulación; esto ofrece la ventaja de realizar conversiones de movimiento (circular a lineal, rotacion en 2 planos, etc.).

Es posible utilizar al mismo tiempo una reducción en la articulación y despues transmitir el movimiento mediante una transmisión; incluso, el dispositivo de transmisión puede funcionar como reductor, el efecto negativo de esto esta en que aumenta la complejidad mecánica, se introduce fricciones, rozamientos, juegos mecánicos, impresiciones, errores de posicionado y velocidad.

Así, lo que buscamos en un reductor o en una transmisión es tener momento de inercia, peso y volumen reducidos, juegos mínimos, alta rigidez torsional, bajo mantenimiento y alta duración.

Las transmisiones más empleadas son las cadenas, cables, correas y enlaces rígidos. Si bien es cierto que los mecanismos de transmisión mencionados también permiten reducción, los reductores más empleados son los trenes de engranes y los reductores armónicos.

Ahora bien, al tener un accionador funcionando, si lo que queremos es reducir la velocidad de un elemento mecánico utilizamos los frenos; también se utilizan cuando queremos mantener una posición fija.

Todos los elementos anteriores corresponden al cuerpo principal del robot; es decir, aquel sistema mecánico que nos permite posicionar el extremo del mismo en el espacio;

ahora trabajaremos el sistema mecánico para la orientación y encargado de realizar la tarea; esto se consigue a través del elemento terminal, el cual esta sujeto al brazo del robot mediante una "muñeca" que da la orientación.

La elección de la muñeca viene dado por la precisión y potencia requerida, así como de las posibles orientaciones que se requieran alcanzar. Las características más deseables para estos dispositivos son el tamaño reducido, modelado matemático sencillo, potencia adecuada a la aplicación a realizar y que el elemento terminal este cerca de los ejes de la muñeca para mayor precisión.

En cuanto al elemento terminal, para cada aplicación en particular será necesario una herramienta en específico para acoplar al extremo de la muñeca, estos pueden ser garras o elementos que permiten sujetar y manipular objetos, herramientas que permiten realizar operaciones, etc.

Con todo la información anterior tenemos idea de como funciona físicamente un robot manipulador, seguiremos ahora con los sentidos del robot.

2.4.2. Sistema sensorial

Nuestro robot debe disponer de una serie de sensores que aporten la información necesaria del robot como posición, orientación, velocidad, fuerza, etc. Esto tiene como objetivo asegurarnos de que la tarea de posicionamiento y orientación se lleve a cabo de manera eficiente, pues facilita el control de movimiento a través de las posiciones y orientaciones que toma el robot en cada instante de tiempo en nuestra aplicación.

Por lo general estos dispositivos se encuentran dentro del robot mismo (sensores internos), pero si se requiere flexibilidad y autonomía de actuación los sensores pueden ser externos.

Dentro de los sensores más importantes están:

De posición : Su función es medir o detectar la posición de un determinado objeto en el espacio de trabajo. Dentro de esta categoría encontramos sonsores foto-electricos, magnéticos, cámaras de video, sensores de contacto, por ultrasonido, etc.

De esfuerzo : Se encargan de medir la fuerza que se aplica a un objeto, medir los pares, etc.

De movimiento : Nos permiten cuantificar los desplazamientos de objetos, la velocidad y aceleración de los mismos.

2.4.3. Sistema de control

Este sistema procesa la información recibida del sistema sensorial y controla los movimientos, tareas, formas de acción o inacción según la programación hecha.

La tarea de este sistema puede ser llevada a cabo por una PC, un FPGA, un microcontrolador o un procesador; la elección del dispositivo de control depende de la complejidad y numero de los sensores así como de la dificultad de la tarea a realizar.

Capítulo 3

Diseño de Robots manipuladores

Puesto que el objetivo de esta tesis es simular un robot manipulador y ver como reacciona ante la información de un juego de air-hockey, el primer paso es diseñar nuestro robot manipulador con tal de que cumpla con todas las características necesarias para la simulación correcta y eficiente.

Este capítulo nos ayuda a aprender como realizar un diseño de un robot manipulador para una actividad en específico. Como producto de este capítulo tenemos una herramienta en MATLAB [**?**] que nos permitirá diseñar una gran diversidad de robots manipuladores así como evaluar sus características mas importantes como alcance, cinemática, movimiento en trayectorias y desempeño al momento de agarrar objetos.

La sección 3.1 nos da una perspectiva del porque es importante realizar un diseño adecuado de robots manipuladores. La sección siguiente (3.2) nos da las herramientas básicas para el diseño de robots manipuladores además de que a la par, nos da ejemplos de la implementación de cada uno de estos pasos con el toolbox para MATLAB de robótica. Teniendo el modelo de nuestro robot manipulador en la sección 3.3 analizamos su capacidad para alcanzar puntos dentro del espacio de trabajo tomando como entrada movimientos de cada articulación, problema mejor conocido como *Cinemática Directa*. El problema de cinemática inversa en el cual dado un punto en el espacio de trabajo necesitamos encontrar cuales son los valores de las articulaciones del robot que nos darán esa posición es analizado en la sección 3.4.

La sección que sigue, la 3.6 nos permitirá generar diversas trayectorias que nuestro manipulador pueda realizar con diversas condiciones para el mismo. Finalmente, en la sección 3.7 analizaremos el diseño de un gripper de 2 dedos y su control difuso cuando transporta objetos a través de métodos difusos con tal de que no se le caiga el objeto pero que tampoco lo aplaste. Espero lo disfruten.

3.1. Importancia del diseño de robots manipuladores

El uso de robots manipuladores industriales cada vez es mas común en la industria, pues son capaces de mejorar su eficiencia en términos de exactitud, consumo de energía y alcance para el espacio de trabajo con tan solo pocas modificaciones.

Antes de emplear un robot en la industria se debe verificar que cumple todos los requerimientos para la tarea a realizar. Si se trabaja con prototipos en la etapa de prueba, estos resultan ser costosos por lo que si nos equivocamos habremos desperdiciado enorme cantidad de tiempo y dinero aparte de que regresaremos a la etapa de diseño de nuestro robot, ya que el modelado es la base de cualquier análisis de robot (cinemática directa e inversa, fuerzas dinámicas y estáticas, etc.) así como del proceso de construir el mecanismo.

En el campo de la robótica usar simulaciones evita el desgaste de tiempo y evita los altos costos de prototipos "a ciegas". Por otra parte, uno de los mejores sistemas para enseñar, diseñar y probar sistemas físicos, son las cajas de herramientas o "toolboxes" de MATLAB. Actualmente hay varios sistemas que nos ayudan a modelar robots; los mas famosos son caros y tienen la limitante de no ser tan interactivos con el usuario. El sistema de simulación de robots manipuladores mas viable presenta problemas de generación de trayectorias y no tiene simulación de elemento terminal [**?**]. Ante esta situación recurrimos a la ya reconocida técnica: "¡Hágalo usted mismo!".

A lo largo de este capítulo elaboramos nuestro propio sistema en MATLAB para el modelado y análisis de robots manipuladores. Ya que un robot manipulador es un manipulador (valga la redundancia) conformado por eslabones situados en una cadena y conectados por articulaciones, nuestro primer paso en el proceso de modelado es representar cada una de estar partes de manera adecuada. Después de esto crearemos en MATLAB un objeto robot con ciertas características y que agrupa un conjunto de objetos llamados eslabones dispuestos en cierto orden, lugar y orientación. Con esta nueva herramienta de MATLAB el usuario es capaz de visualizar el resultado de crear el robot a través de un modelo gráfico para empezar a estudiar la cinemática del robot, la generación de trayectorias y rutas de movimiento, así como la dinámica y comportamiento del elemento terminal.

El propósito del "toolbox" desarrollado en esta sección es mejorar el tiempo y efectividad de diseño. El usuario de esta herramienta encontrará al igual que yo (en mi caso, me di cuenta hasta que termine la herramienta) que es amigable para entender abierta y completamente los conceptos y funcionamiento del modelo de un robot manipulador. Los programas y/o comandos generados se mencionaran durante el desarrollo de este capítulo.

3.2. Construyendo el modelo

Construir el modelo de nuestro robot manipulador nos permitirá analizar posteriormente si cumple con las características físicas para realizar una actividad determinada (espacio de trabajo, evasión de obstáculos, alcances, etc.) así como obtener de manera eficiente las ecuaciones cinemáticas de nuestro mecanismo para verificar su desempeño en términos de su cinemática y dinámica.

En caso de que en la fase de análisis de nuestro manipulador no se cumpla con algún objetivo hemos de regresar a la fase de diseño y rectificar la característica necesaria para cumplir con el objetivo que faltó (algunos de los parámetros de Denavit-Hartenberg); por lo que los conocimientos que nos permitirán realizar de manera rápida y eficiente la etapa de diseño y en determinado caso rectificación, están dados en las secciones que a continuación se describen:

1.- Lo primero es lo primero y esto viene a ser el como describir la localización (posición y orientación) y movimientos de los diferentes componentes de nuestro robot manipulador entre sí (Seccion 3.2.1).

2.- Procederemos a continuación a numerar eslabones, articulaciones y localizar los ejes de articulación de nuestro robot manipulador (Sección 3.2.2).

3.- El siguiente paso consiste en establecer adecuadamente los sistemas de referencia para cada eslabón en base a los ejes de articulación correspondientes (Sección 3.2.3).

4.- Habiendo realizado los pasos anteriores debemos describir nuestro robot manipulador con tal de analizar su cinemática y dinámica; esto se puede llevar a cabo creando la tabla de parámetros de Denavit-Hartenberg (Sección 3.2.4).

5.- Teniendo la tabla de parámetros DH y la asignación de sistemas de referencia, buscamos obtener la posición y orientación de cada eslabón; combinando el conocimiento de las secciones 3.2.4 y 3.2.1 podemos realizar este objetivo. Todo esto se presenta en la sección 3.2.5.

6.- Finalmente, un ejemplo en MATLAB nos permitirá familiarizarnos con los pasos de las secciones 3.2.2 a 3.2.4, así como practicar los comandos empleados en nuestro toolbox de robótica y verificar los resultados de lo aprendido en esta sección al construir el modelo de un robot manipulador 3.2.6.

3.2.1. Descripción de la localización

Para realizar una determinada tarea un robot manipulador debe saber en todo momento su localización adecuada en el espacio, es decir, debe saber que posición y orientación tiene.

Un robot manipulador trabaja en el espacio tridimensional, de aquí que tenga que ser referenciado en este mismo espacio. Una **posición** se establece de forma unívoca mediante un vector de posición con tres componentes con respecto a un sistema de referencia M siendo este el origen y el extremo la posición. Para describir la **orientación** de un cuerpo adjuntamos un sistemas de coordenadas al cuerpo O (3 por ser tridimensional) y luego daremos una descripción de este sistema de coordenadas relativo al sistema de referencia M. Para una mejor visualización gráfica a veces se opta por representar ambos sistemas coincidentes en el origen.

Una forma de indicar la orientación de un sistema O con respecto a otro M, es hacerlo mediante las coordenadas en el sistema M de los vectores unitarios en la dirección de los ejes del sistema O; es decir las proyecciones de los vectores unitarios en O (cada una de sus coordenadas) sobre los ejes del sistema M. Teniendo esto en cuenta y los sistemas de referencia ortogonales y dextrógiros, con 3 parámetros (los vectores proyectados), la orientación de un sistema con respecto a otro queda determinada.

Sean x_O, y_O y z_O los 3 vectores unitarios del sistema O, al escribirlos en términos del sistema de coordenadas M, se llaman x_O^M, y_O^M y z_O^M. Si colocamos estos 3 vectores unitarios como columnas de una matriz de 3 x 3 en el orden x_O^M, y_O^M y z_O^M nos queda una forma eficiente de representación llamada **matriz de rotación** $\mathbf{Rot}_O^M$ (Ecuac. 3.2.1); esta matriz describe a O en forma relativa a M.

$$\mathbf{Rot}_O^M = [x_O^M y_O^M z_O^M] = \begin{bmatrix} x_O \cdot x_M & y_O \cdot x_M & z_O \cdot x_M \\ x_O \cdot y_M & y_O \cdot y_M & z_O \cdot y_M \\ x_O \cdot z_M & y_O \cdot z_M & z_O \cdot z_M \end{bmatrix} \tag{3.2.1}$$

Cada uno de los elementos de la matriz 3.2.1 corresponden a los cosenos de los ángulos formados entre los vectores correspondientes. Por otra parte, la orientación del sistema M respecto al O es la orientación inversa y se expresa a través de la ecuación 3.2.2 , de aquí que la matriz de rotación traspuesta es la matriz de rotación inversa $\mathbf{Rot}_O^M = \left(\mathbf{Rot}_M^O\right)^T = \left(\mathbf{Rot}_M^O\right)^{-1}$.

$$\mathbf{Rot}_M^O = [x_M^O y_M^O z_M^O] = \begin{bmatrix} x_M \cdot x_O & y_M \cdot x_O & z_M \cdot x_O \\ x_M \cdot y_O & y_M \cdot y_O & z_M \cdot y_O \\ x_M \cdot z_O & y_M \cdot z_O & z_M \cdot z_O \end{bmatrix} \tag{3.2.2}$$

Puesto que es ideal tener la información de posición y orientación conjuntamente, se hace uso de las *matrices de transformación homogénea* cuya dimensión es de 4 x 4 y esta compuesta por cuatro submatrices (Ecuac. 3.2.3).

$$\mathbf{T} = \left[\begin{array}{c|c} \textit{rotación} & \textit{traslación} \\ \hline \textit{perspectiva} & \textit{escalado} \end{array}\right] = \left[\begin{array}{ccc|c} & & & 3 \\ 3 & \times & 3 & \times \\ & & & 1 \\ \hline 1 & \times & 3 & 1 \times 1 \end{array}\right] \tag{3.2.3}$$

En robótica, el vector de perspectiva es el vector cero y la constante de escalado es igual a 1, las 3 primeras columnas columnas de la matriz homogénea representan las direcciones de los ejes principales de un sistema asociado a un objeto O referenciados a un sistema M, mientras que la cuarta columna representa la posición del sistema asociado al objeto O respecto al sistema de referencia M (3.2.4).

$$\mathbf{T} = \left[\begin{array}{c|c} \textit{rotación} & \textit{traslación} \\ \hline 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|c} & & & 3 \\ 3 & \times & 3 & \times \\ & & & 1 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{3.2.4}$$

En muchas ocasiones nos interesa conocer la localización de M respecto a O, lo que corresponde a la matriz de transformación inversa a 3.2.4. Haciendo uso de álgebra podemos observar que la fórmula mas sencilla para encontrar $\mathbf{T}^{-1}$ es 3.2.5 (Comando *hom_inv*).

$$\mathbf{T}^{-1} = \left[\begin{array}{c|c} \textit{rotación}^T & -\textit{rotación}^T \cdot \textit{traslación} \\ \hline 0 & 1 \end{array}\right] \tag{3.2.5}$$

Un movimiento de traslación $\mathbf{Tras}(\mathbf{p}) = \mathbf{Tras}(x, y, z)$ en coordenadas homogéneas se representa como en 3.2.6 pues la submatriz de rotación es la identidad ya que no ha producido rotación alguna (Comando implementado en *trasl2hom*).

$$\mathbf{Tras}(\mathbf{p}) = \mathbf{Tras}(x, y, z) = \left[\begin{array}{ccc|c} 1 & 0 & 0 & \mathrm{x} \\ 0 & 1 & 0 & \mathrm{y} \\ 0 & 0 & 1 & \mathrm{z} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{3.2.6}$$

Cuando consideramos traslaciones compuestas o totales, al tratarse de vectores, el orden en que se efectúan las operaciones básicas de traslación no afecta el resultado de la traslación total.

En cuanto a las rotaciones, hay 3 rotaciones básicas considerando giros sobre cada uno de los ejes coordenados: Giro respecto al eje x ($\mathbf{Rot}(x,\alpha)$ implementado en *rotx2hom* a través de la fórmula 3.2.7), al eje y ($\mathbf{Rot}(y,\beta)$ implementado en *roty2hom* a través de 3.2.8) y al eje z ($\mathbf{Rot}(z,\gamma)$ implementado en *rotz2hom* a través de 3.2.9).

$$\mathbf{Rot}(x,\alpha) = \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & \cos(90+\alpha) & 0 \\ 0 & \cos(90-\alpha) & \cos(\alpha) & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{3.2.7}$$

$$\mathbf{Rot}(x,\alpha) = \left[\begin{array}{ccc|c} \cos(\beta) & 0 & cos(90-\beta) & 0 \\ 0 & 1 & 0 & 0 \\ cos(90+\beta) & 0 & \cos(\beta) & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|c} \cos(\beta) & 0 & \sin(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{3.2.8}$$

$$\mathbf{Rot}(x,\alpha) = \left[\begin{array}{ccc|c} \cos(\gamma) & \cos(90+\gamma) & 0 & 0 \\ \cos(90-\gamma) & \cos(\gamma) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|c} \cos(\gamma) & \sin(\gamma) & 0 & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{3.2.9}$$

Al igual que en las traslaciones, resulta importante expresar un giro total completo en términos de las rotaciones básicas individuales, lo cual no es tan sencillo porque el producto de matrices no es conmutativo, esto se puede interpretar geométricamente como el hecho de que la localización final dependerá del orden con que se haya efectuado cada una de las rotaciones básicas.

En robótica, una transformación total se descompone en una serie de trasformaciones básicas de traslación y de rotación. Se hace uso de una traslación si lo que cambia es la posición del objeto con respecto a un sistema de referencia y/o de una rotación, si se produce un giro del objeto con respecto al sistema de referencia.

Puesto que aplicar una transformación implica una multiplicación de matrices, no solo es importante el orden en que se aplican, sino que también es necesario identificar en cada transformación con respecto a qué sistema se realiza. Esto es, que cuando se realiza una transformación, hay 2 maneras de referenciarlo:

Respecto al sistema móvil : Cuando se referencia con respecto al sistema resultante de la transformación anterior, es decir, con respecto a la última localización del sistema transformado, la nueva transformación (una matriz homogénea) se postmultiplica respecto a las aplicadas previamente.

Respecto al sistema fijo : Cuando se referencia con respecto al que fue de referencia para la última transformación. En este caso, la matriz de transformación homogénea de la nueva transformación se premultiplica sobre las transformaciones ya efectuadas.

Ya habíamos mencionado que un giro se puede descomponer en una combinación de 3 rotaciones básicas realizadas en cierto orden, a esto hay que añadir que es posible obtener mas de una agrupación de rotaciones básicas para un mismo giro general. Si nos ponemos a analizar con detalle, existen 24 combinaciones para hacer un giro general; estos giros se agrupan en 2 conjuntos:

En sistema fijo : Son 12 rotaciones que se obtienen mediante combinación de 3 rotaciones simples realizadas sobre los ejes principales del sistema fijo; dentro de estas la mas utilizada es el giro de combinación *X-Y-Z*, también conocida como giros de balance (roll,ϕ), inclinación (pitch,θ) y orientación (yaw,ψ).

En sistema móvil : 12 rotaciones mejor conocidas como *ángulos de Euler* se definen como combinación de 3 giros sobre ejes principales del sistema móvil.

En el comando *rpy2hom* se realiza la implementación de un giro en la combinación de roll-pitch-yaw (rpy)a través de la fórmula 3.2.10.

$$\mathbf{Rot}(X-Y-Zfijo) = \mathbf{Rot}(z,\phi)\cdot\mathbf{Rot}(y,\theta)\cdot\mathbf{Rot}(x,\psi) \tag{3.2.10}$$

Si $c\phi$ es la abreviación de $cos(\phi)$, $s\phi$ de $sen(\phi)$ y así sucesivamente y después de realizar la multiplicación de 3.2.10 obtenemos

$$\mathbf{Rot}(X-Y-Zfijo) = \left[\begin{array}{ccc|c} c\phi c\theta & c\phi s\theta s\psi - s\phi c\psi & c\phi s\theta c\psi + s\phi s\psi & 0 \\ s\phi c\theta & s\phi s\theta s\psi + c\phi c\psi & s\phi s\theta c\psi - c\phi s\psi & 0 \\ -s\theta & c\theta s\psi & c\theta c\psi & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{3.2.11}$$

El problema inverso, de obtener los ángulos $X-Y-Z$ equivalentes a partir de una matriz de rotación también es de suma importancia y se resuelve al tener en

cuenta la ecuación 3.2.11 y considerar ϕ, θ y ψ como variables. Viéndolo desde esta perspectiva, tenemos 9 ecuaciones y 3 incógnitas, pero hay 6 dependencias de aquí que nos queden solo 3 ecuaciones y 3 incógnitas.

$$\mathbf{Rot}(X-Y-Zfijo) = \left[\begin{array}{ccc|c} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \\ \hline r_{41} & r_{42} & r_{43} & r_{44} \end{array}\right] \tag{3.2.12}$$

Supongamos que tenemos 3.2.12. La forma mas eficiente de obtener los ángulos $X-Y-Z$ (rpy), siempre y cuando $\cos(\theta) \neq 0$ (incluyendo el caso numérico en la computadora) es partir de 3.2.13 implementada en el comando *hom2rpy*. Al sacar la raíz cuadrada de la suma de r_{11} y r_{21} obtenemos $\cos(\theta)$ y resolvemos para θ con el arco tangente de $-r31$ sobre el coseno calculado.

$$\begin{aligned} \theta &= Atan2\left(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2}\right) \\ \phi &= Atan2\left(\tfrac{r_{21}}{\cos(\theta)}, \tfrac{r_{11}}{\cos(\theta)}\right) \\ \psi &= Atan2\left(\tfrac{r_{32}}{\cos(\theta)}, \tfrac{r_{33}}{\cos(\theta)}\right) \end{aligned} \tag{3.2.13}$$

Existe una segunda solución también implementada en el comando *hom2rpy*, resultado de utilizar la raíz cuadrada negativa en la fórmula para θ.

En el caso de los ángulos de Euler, se usan con mayor frecuencia en la especificación de orientaciones y de las 12 combinaciones posibles sobre un sistema móvil, la mas utilizada es la sucesión de rotaciones $Z-Y-Z$, esta transformación puede ser evaluada por la fórmula 3.2.14 implementada en el comando *eul2hom*.

$$\mathbf{Rot}(Z-Y-Zmovil) = \mathbf{Rot}(z,\phi)\cdot\mathbf{Rot}(y,\theta)\cdot\mathbf{Rot}(z,\psi) \tag{3.2.14}$$

Despues de haber echo la multiplicación de matrices la ecuación 3.2.14 nos queda:

$$\mathbf{Rot}(Z-Y-Zmovil) = \left[\begin{array}{ccc|c} c\phi c\theta c\psi - s\phi s\psi & -c\phi c\theta s\psi - s\phi c\psi & c\phi s\theta & 0 \\ s\phi c\theta c\psi + c\phi s\psi & -s\phi c\theta s\psi + c\phi c\psi & s\phi s\theta & 0 \\ -s\theta c\psi & s\theta s\psi & c\theta & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{3.2.15}$$

Para el caso de determinar los ángulos de Euler dada una matriz de transformación homogénea como en 3.2.12 podemos determinarlos a través de las ecuaciones 3.2.16

implementadas en el comando *hom2eul*.

$$\begin{aligned} \phi &= Atan2(r_{23}, r_{13}) \\ \theta &= Atan2(\cos(\phi)r_{13} + \sin(\phi)r_{23}, r_{33}) \\ \psi &= Atan2(-\sin(\phi)r_{11} + \cos(\phi)r_{21}, -\sin(\phi)r_{12} + \cos(\phi)r_{22}) \end{aligned} \tag{3.2.16}$$

En el caso de los ángulos de Euler también se cuenta con una segunda solución si tomamos el caso en que $\phi = \phi + \pi$.

Un punto mas a considerar en describir la localización de todo componente de un robot es el orden en que aplican las transformaciones pues puede haber una gran diferencia en la cantidad de computo requerido para calcular la misma cantidad, por lo que en el proceso de realizar multiples rotaciones de un vector es mas eficiente si vamos multiplicando vector por matriz en vez de multiplicar primero las matrices, salvo en el caso de que las matrices de rotación sean constantes.

3.2.2. Numeración de las partes de un manipulador

Consideremos un robot genérico, compuesto por n+1 eslabones conectados consecutivamente por n articulaciones de un grado de libertad. Los eslabones se numeran iniciando con 0 en la base y las articulaciones comenzando con 1 en la que conecta la base con el eslabón 1 (Figura 3.1). El eje de cada articulación es el eje respecto al que se produce el movimiento de la articulación, el de giro en una articulación rotacional y el de la dirección de desplazamiento en una articulación prismática. A partir de los conceptos anteriores podemos definir las 4 cantidades de Denavit-Hartenberg.

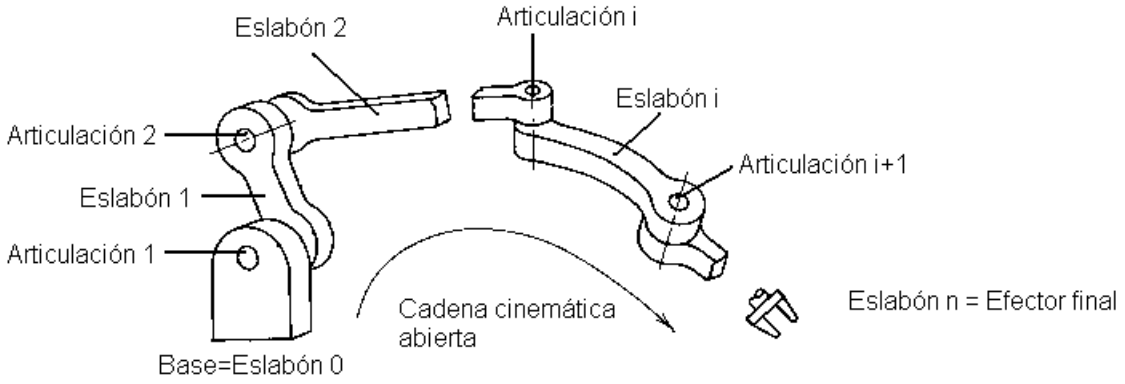


Figura 3.1: Numeración de eslabones y articulaciones

3.2.3. Asignación de sistemas de referencia

Cualquier robot puede describirse en forma cinemática proporcionando los valores de Denavit-Hartenberg para cada eslabón, pero para poder describir la ubicación de cada eslabón relativa a sus eslabones adyacentes es necesario asignar un sistema de referencia a cada eslabón.

Es importante destacar que hay varias maneras de asignar un sistema de referencia a un mismo eslabón, con el propósito de que algunos de estos parámetros para determinados eslabones resulten nulos y así las operaciones de cinemática y/o dinámica sean más sencillas.

Dos diferentes asignaciones son realizadas por Paul [**?**] en 1981 y Craig [**?**] en 1986. Ambos sistemas son utilizados en la actualidad por su fácil comprensión y también debido a que reducen la complejidad de las operaciones cinemáticas y dinámicas para la mayoría de los modelos de robots manipuladores. En estos sistemas de asignación varían las matrices de transformación intermedias, pero resulta idéntica la transformación final.

Tomando en cuanta la numeración de eslabones y articulaciones de la sección anterior y mostrados en la figura 3.1, la asignación de sistemas de referencia se puede realizar de la siguiente manera:

Asignación de Paul : Localiza el sistema de referencia del eslabón en el eje de la articulación que lo enlaza con el siguiente eslabón (Figura 3.2). En el caso de la base del robot o eslabón 0, su eje Z_0 esta alineado con el eje de la articulación 1 y se toman los ejes X_0 e Y_0 para que el sistema sea dextrógiro; cabe mencionar que este sistema es fijo y se coloca en la base del robot.

En cuanto a los ejes de las articulaciones 1 a n-1, el eje Z_i del sistema de referencia del eslabón i se alinea con el eje de la articulación i+1.

El sistema de referencia del último eslabón se sitúa al final de este mismo (extremo del robot) y ya que no existe la articulación n+1, su eje Z_n coincide en dirección con el eje Z del sistema asociado al eslabón n-1.

Asignación de Craig : Cuando asignamos sistema de referencia a la base del robot o eslabón 0, este no se mueve y se elige Z_0 sobre el eje 1 de manera que coincidan el sistema de referencia 0 y 1 cuando la variable de articulación 1 es cero. Con esto, siempre tendremos que $a_0 = 0$, $\alpha_0 = 0$, y que $d_1 = 0$ si la articulación 1 es angular o $\theta_1 = 0$ si es prismática la articulación.

Para el caso de los eslabones intermedios (1 a n-1) el sistema de referencia i se asigna al eslabón i y su eje Z es coincidente con el eje de la articulación i (Figura 3.3).

En el caso de la articulación n, si es angular, la dirección X_n se ubica de tal manera que se alinee con X_{n-1} cuando $\theta_n = 0$ y el origen se selecciona con tal de que $d_n = 0$; para el caso en el que la articulación es prismática X_n se localiza con tal de que $\theta_n = 0$ y el origen en la intersección de X_{n-1} y el eje de articulación n cuando $d_n = 0$.

Hemos de mencionar que se añade un sistema de referencia extra en el extremo de cualquier herramienta que el robot este sujetando; en el caso de una mano o pinza, el sistema se ubica con su origen entre las puntas de los dedos.

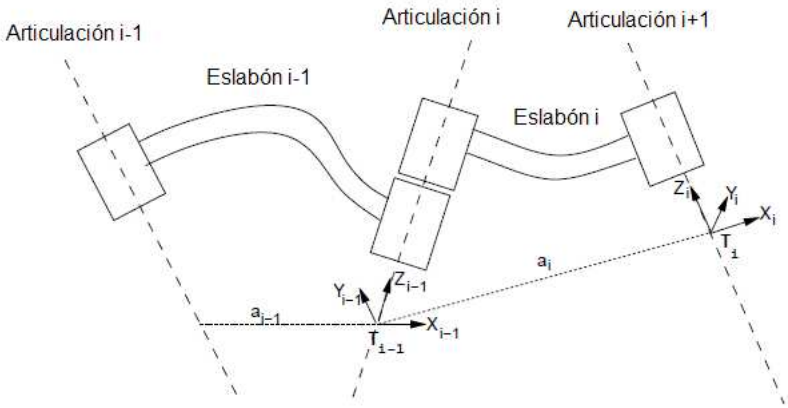


Figura 3.2: Asignación de sistemas de referencia usado por Paul

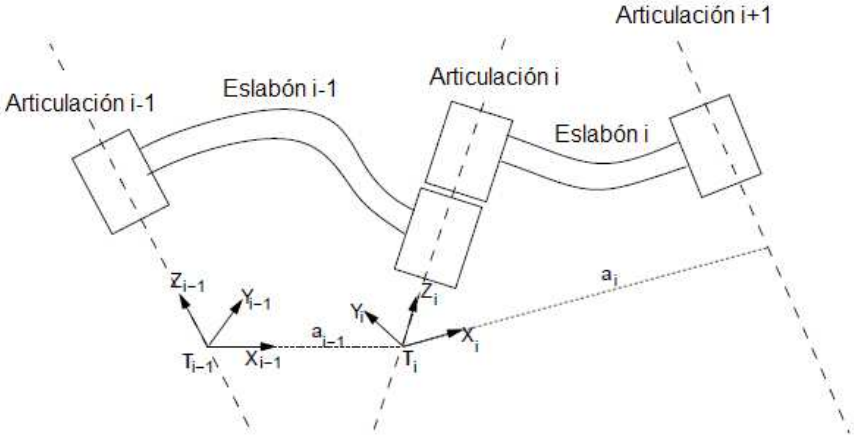


Figura 3.3: Asignación de sistemas de referencia usado por Craig

En ambos tipos de asignación (Paul y Craig) el eje X_i de este sistema se alinea con la normal común entre las articulaciones i e i+1, mientras que el eje Y_i se establece para que el sistema sea dextrógiro. Así también, cuando la línea normal común no es única , el origen del sistema queda indefinido, pero por convenio se toma el origen en

la articulación i+1. En el caso en el que los ejes se cortan, el origen del sistema se localiza en el punto de corte, en este caso X_i está según la dirección perpendicular al plano que forman el eje Z_i y Z_{i-1}, tomando el sentido de forma arbitraria.

Dado que hay diferencia en la asignación de sistemas de referencia, la tabla de parámetros de Denavit-Hartenberg difiere dependiendo del tipo de asignación de sistemas.

3.2.4. Parámetros de Denavit - Hartenberg

Recordando que un robot manipulador es un conjunto de cuerpos conectados en una cadena mediante articulaciones (prismáticas o rotacionales), llamados eslabones (o vínculos) y que gracias a la sección anterior podemos obtener la posición y orientación de los vínculos de los manipuladores en situaciones estáticas, solo nos falta describir las relaciones existentes entre estos vínculos. El objetivo de esta sección es describir como obtener una descripción de las relaciones importantes entre articulaciones y eslabones con tal de que posteriormente podamos estudiar la cinemática y dinámica del manipulador.

Es bastante claro que un solo eslabón tiene muchos atributos a considerar durante la etapa de diseño (tipo de material empleado, fuerza y rigidez del vínculo, ubicación de la articulación, su forma, inercia, peso, etc.), así también en el caso de las articulaciones y su interconexión (solidez, lubricación, engranaje, etc.); sin embargo cuando el objetivo es encontrar las ecuaciones cinemáticas del mecanismo, *un eslabón se considera solamente como un cuerpo rígido que define la relación entre 2 ejes de articulaciones adyacentes de un manipulador*, ante esta situación hay varios métodos disponibles para describir nuestro robot manipulador y la convención mas utilizada es la de los *parámetros de Denavit-Hartenberg* [**?**], también llamados parámetros DH.

Los parámetros de Denavit-Hartenberg son 4 cantidades para cada vínculo (a partir del eslabón 1 en el caso de la asignación de sistemas de referencia de Paul y del eslabón 0 en el de Craig); 2 números describen el eslabón en sí (longitud de eslabón y torsión de eslabón) y los otros 2 describen la conexión del eslabón con un eslabón adyacente (desplazamiento de eslabón y ángulo de articulación). Para esta notación, los ejes de articulación son la base para obtener estas 4 cantidades, de aquí la importancia de que la sección 3.2.3 haya quedado super bien entendida :).

Longitud de eslabón : Representado por **a**. Define la longitud del eslabón. Es la distancia entre 2 ejes de articulaciones adyacentes (las cuales delimitan al eslabón) a lo largo de la normal común.

Torsión de eslabón : Representado por α. Este parámetro mide de cierta manera

la forma del eslabón. Es un parámetro que permite definir la ubicación relativa de los 2 ejes entre los cuales se encuentra el eslabón y se define como el ángulo medido entre los ejes proyectados sobre el plano cuya normal sea la línea mutuamente perpendicular a las 2 articulaciones adyacentes que delimitan al eslabón (la utilizada en la *longitud del eslabón*). El ángulo se mide desde el eje en que estamos hasta el eje siguiente en el sentido de la mano derecha.

Desplazamiento de eslabón : Representado por **d**. Parámetro que relaciona 2 eslabones y en cierto modo expresa la distancia entre los 2 eslabones, marcado por el tamaño y la forma de la articulación; también recibe el nombre de longitud articular. Para un eslabón i, **d** se define como la distancia entre las intersecciones de las normales comunes a la articulación i (dos, una respecto a la articulación anterior y la otra respecto a la siguiente), a lo largo del eje de la articulación.

Para las articulaciones 1 y n, al no existir las articulaciones 0 y n+1, no existen líneas normales comunes entre 0 y 1, ni entre n y n+1, por lo que d_1 por convenio es la distancia de la linea normal común de las articulaciones 1 y 2 hasta el sistema de la base y para d_n se toma la distancia entre la linea normal común de las articulaciones n-1 y n al sistema del extremo del robot.

Ángulo de articulación : Representado por θ. Expresa el ángulo que forman 2 eslabones, determinado por la forma de la articulación; también recibe el nombre de ángulo articular. Describe la cantidad de rotación sobre el eje de la articulación común entre 2 eslabones adyacentes. Dados 2 eslabones adyacentes y su articulación común, se define como el ángulo que existe entre las líneas normales comunes al eje de la articulación (dos, una respecto a la articulación anterior y la otra respecto a la siguiente).

Los parámetros **a** y α de Denavit-Hartenberg relativos a la forma y el tamaño del eslabón, al ser éste rígido y despreciadas todo tipo de deformaciones que se puedan introducir en el material, una vez determinados no sufren variación alguna, a diferencia de los parámetros **d** y θ correspondientes a la posición relativa entre eslabones ya que estos varían al estar enlazados por una articulación.

Cuando tenemos una articulación de tipo rotacional, se producirá una variación del parámetro θ, ya que es el ángulo relativo que forman los eslabones que enlaza y las otras 3 cantidades son cantidades de parámetros fijas. Por el contrario, si la articulación es prismática se realizará una variación en el parámetro **d** y las otras 3 se mantienen fijas.

En el caso de la asignación de Craig, recordemos que añadimos un sistema de referencia mas, este sistema de referencia se asigna a la herramienta con la que cuenta el manipulador por lo que al calcular sus parámetros DH si la herramienta no tiene movimiento θ y **d** tendrán el valor constante de 0.

3.2.5. De parámetros DH a transformaciones homogéneas

Bueno, ya tenemos los sistemas de referencia y los parámetros DH que nos permiten representar las características de cada eslabón de nuestro robot manipulador; si queremos saber cual es la matriz de transformación homogénea para pasar del sistema de referencia i-1 al i solo necesitamos 2 giros y 2 traslaciones en el orden adecuado dada la información de los parámetros DH para la articulación i.

La importancia de esta sección se debe a que al calcular estas matrices de transformación podemos saber la posición de nuestro elemento terminal o de cada eslabón con tan solo conocer los valores de movimiento de cada articulación (rotacional o traslacional); no olvidemos que estos valores son los de θ o $\mathbf{d}$ de nuestra tabla de parámetros DH los cuales son variables.

Si utilizamos la convención de Paul para asignar sistemas de referencia, la matriz de transformación homogénea para pasar de un sistema i-1 a i en términos de los giros y traslaciones necesarios esta dado por 3.2.17. Haciendo los respectivos cálculos según lo aprendido en la sección 3.2.1 nos queda la ecuación 3.2.18.

$$\mathbf{T}_i^{i-1} = \mathbf{Rot}(Z_{i-1}, \theta_i) \cdot \mathbf{Tras}(Z_{i-1}, d_i) \cdot \mathbf{Tras}(X_i, a_i) \cdot \mathbf{Rot}(X_i, \alpha_i) \tag{3.2.17}$$

$$\mathbf{T}_i^{i-1} = \left[\begin{array}{ccc|c} cos(\theta_i) & -cos(\alpha_i)sen(\theta_i) & sen(\alpha_i)sen(\theta_i) & a_i cos(\theta_i) \\ sen(\theta_i) & cos(\alpha_i)cos(\theta_i) & -sen(\alpha_i)cos(\theta_i) & a_i sen(\theta_i) \\ 0 & sen(\alpha_i) & cos(\alpha_i) & d_i \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{3.2.18}$$

Por otra parte, en dado caso que hayamos utilizado el criterio de Craig para asignar los sistemas de referencia, el orden de los giros y traslaciones para pasar de un sistema i-1 a i están dados por la ecuación 3.2.19, y realizando las operaciones nos queda la matriz 3.2.20.

$$\mathbf{T}_i^{i-1} = \mathbf{Rot}(X_{i-1}, \alpha_i) \cdot \mathbf{Tras}(X_{i-1}, a_i) \cdot \mathbf{Rot}(Z_i, \theta_i) \cdot \mathbf{Tras}(Z_i, d_i) \tag{3.2.19}$$

$$\mathbf{T}_i^{i-1} = \left[\begin{array}{ccc|c} cos(\theta_i) & -sen(\theta_i) & 0 & a_i \\ sen(\theta_i)cos(\alpha_i) & cos(\theta_i)cos(\alpha_i) & -sen(\alpha_i) & -sen(\alpha_i)d_i \\ sen(\theta_i)sen(\alpha_i) & cos(\theta_i)sen(\alpha_i) & cos(\alpha_i) & cos(\alpha_i)d_i \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{3.2.20}$$

Al tener estas matrices de transformación homogéneas utilizadas para ir de un sistema de referencia a otro, podemos dar por terminada nuestra fase de construcción del modelo de nuestro robot manipulador, lo que nos faltaría es analizar su desempeño en términos de su cinemática y dinámica, lo cual lo veremos en las secciones siguiente, solo que antes de eso, verificaremos lo aprendido con un ejemplo.

3.2.6. Ejemplo

Trabajaremos lo estudiado en las subsecciones anteriores con un brazo planar de 2 eslabones y 2 articulaciones rotacionales (mecanismo **RR**). La figura 3.4 es una representación esquemática del manipulador en cuestión.

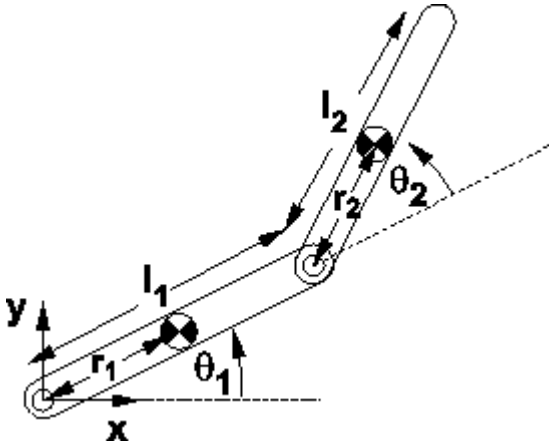


Figura 3.4: Robot planar

Procederemos en base a los pasos mencionados en las secciones 3.2.2 a 3.2.5.

1.- Numeración de eslabones y articulaciones (Véase sección 3.2.2 y figura 3.5).

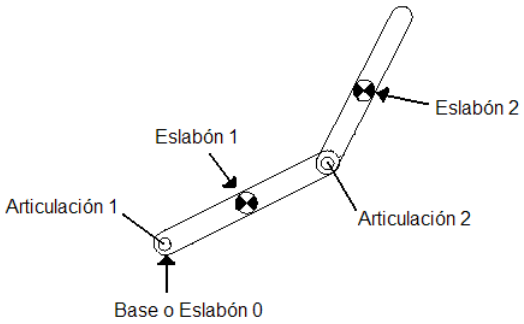


Figura 3.5: Ejemplo de numeración de articulaciones y eslabones en robot planar

2.- Asignación de sistemas de referencia (Véase sección 3.2.3, figura 3.6 y figura 3.7).

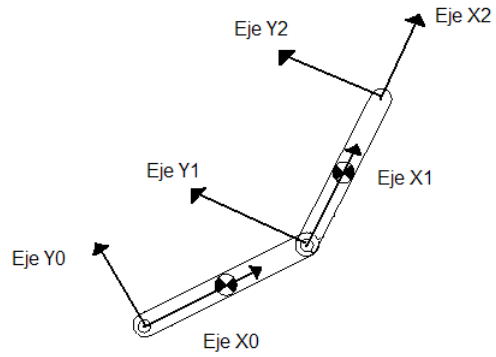


Figura 3.6: Ejemplo de asignación de sistemas de referencia de Paul

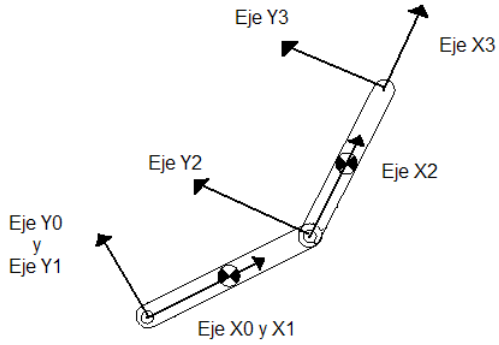


Figura 3.7: Ejemplo de asignación de sistemas de referencia de Craig

3.- Cálculo de tabla de parámetros DH.

Habiendo asignado los sistemas de referencia y recordando las definiciones de cada uno de los parámetros DH de la sección 3.2.4 obtenemos la siguiente información:

Teniendo en cuenta el caso de asignación de sistemas de referencia de Paul recordemos que solo calculamos los parámetros DH de los eslabones 1 y 2, por lo que la tabla queda de la siguiente manera:

Eslabón	α	**a**	θ	**d**
E1	0	l_1	0	0
E2	0	l_2	0	0

(3.2.21)

Para el caso de la asignación de sistemas de referencia de Craig calculamos los parámetros DH de los eslabones 0, 1 y 2, teniendo en cuenta el sistema de referencia

de la herramienta en la punta del eslabón 2 no tiene movimiento alguno, por lo que la tabla de parámetros DH queda:

Eslabón	α	**a**	θ	**d**
E0	0	0	0	0
E1	0	l_1	0	0
E2	0	l_2	0	0

(3.2.22)

4.- Cálculo de las matrices de transformación homogéneas.

Dado un θ o un **d** determinada para cada uno de las articulaciones de nuestro robot manipulador, tendremos una respectiva matriz de transformación homogénea de que nos lleva de un sistema de referencia a otro.

5.- Representación en MATLAB.

Uno de los objetivos de esta tesis es la elaboración de una herramienta en MATLAB que nos permita analizar de manera eficiente las características de modelos de robots manipuladores. Puesto que a estas alturas de este texto ya podemos obtener las características representativas de un robot manipulador,es momento de verificarlas y para esto acudimos a la herramienta desarrollada para representar eslabones en nuestro toolbox de MATLAB:

Nombre de la clase: ESLABON

Descripción: Clase que nos permite construir el objeto eslabón. Este objeto tiene todos los elementos relacionados a un eslabón, tales como masa, material, cinemática de la articulación y parámetros de inercia de cuerpo rígido.

Tipo de llamadas:

E=Eslabon. Eslabón inicial con todos los parámetros igual a cero.

E=Eslabon(eslabon). Crea copia del parámetro eslabón (Clonación)

E=Eslabon([alfa, a, theta, d, Mov], Peso, Material, Convencion, Dinamica).

Los valores mínimos que se deben dar son los parámetros DH (α, **a**, θ, **d**). Los valores de los ángulos de alfa y theta deben estar en radianes.En cuanto a los otros parámetros:

Mov es el movimiento del eslabón o tipo de articulación sobre la que se monta, trabajamos con 0 si es revoluto y 1 si es prismático.

Peso es el peso del eslabón.

Material se refiere al material utilizado en la construcción del eslabón.

Convencion : Tipo de convención utilizada para asignar ejes de articulación; con 'DHS' nos referimos a la definida por Paul y con 'DHM' a la definida por Craig.

Dinamica: Contiene parámetros como el vector del centro de gravedad del eslabón, velocidad, su matriz simétrica de inercia, inercia del motor, radio de engranes, fricción de viscosidad y los límites de la articulación.

Con la clase anterior podemos generar en MATLAB los eslabones utilizando la asignación de sistemas de referencia de Paul y enfocándonos en que solo tenemos la información de los parámetros DH y que $l_1 = 10$ $l_2 = 15$(Figura 3.8).

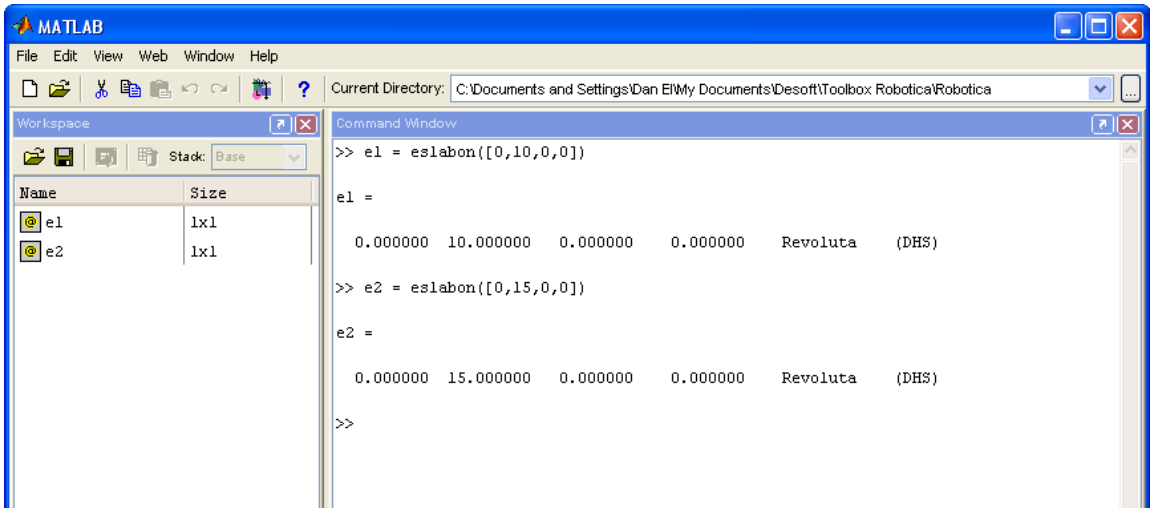


Figura 3.8: Uso de comando ESLABON en MATLAB

Teniendo en memoria estos 2 objetos eslabón procedemos a construir nuestro robot con el siguiente comando:

Nombre de la clase: ROBOT

Descripción: Clase que nos permite construir el objeto robot. Este objeto tiene todos los elementos relacionados a un robot, tales como datos, dirección de gravedad, convención DH, base de robot, opciones de dibujo.

Tipo de llamadas:

R=Robot. Robot inicial con todos los parámetros igual a cero.

R=Robot(robot). Crea copia del objeto "robot" (Clonación)

R=Robot({E1,...,En}, Datos). Crea un nuevo objeto robot con los eslabones E1

a En y con la información especificada en *Datos* (Nombre y comentarios). Otras características del objeto robot accesibles a través de funciones públicas son el número de articulaciones, convención utilizada, dirección de la gravedad respecto al sistema, una matriz homogénea que representa la localización de la base del robot, una matriz homogénea del elemento terminal, ángulos actuales de la articulación y opciones de dibujo.

Aplicando este objeto en MATLAB a nuestros eslabones antes definidos obtenemos lo mostrado en la Figura 3.9.

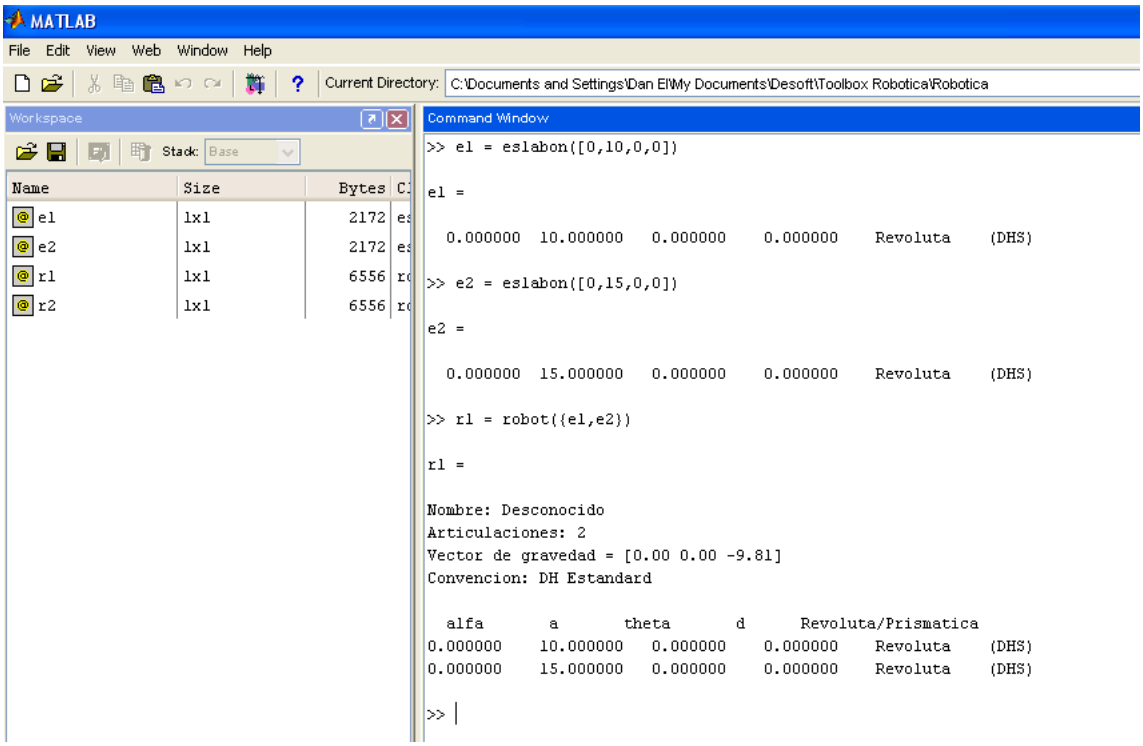


Figura 3.9: Uso de comando ROBOT en MATLAB

Para verificar que nuestro modelo hacemos uso del siguiente comando desarrollado:

Nombre del comando: DRIVEBOT

Descripción: Manejador gráfico de un objeto ROBOT

Tipo de llamadas:

DRIVEBOT(ROBOT).

DRIVEBOT(ROBOT, Q).

Si no hay objeto ROBOT gráfico, creamos la ventana de dibujo. Q son los ángulos

iniciales de las articulaciones, si no hay Q se inicializa con los valores existentes en el gráfico.

Utilizando el comando *drivebot* nos queda lo mostrado en las Figuras 3.10 y 3.11.

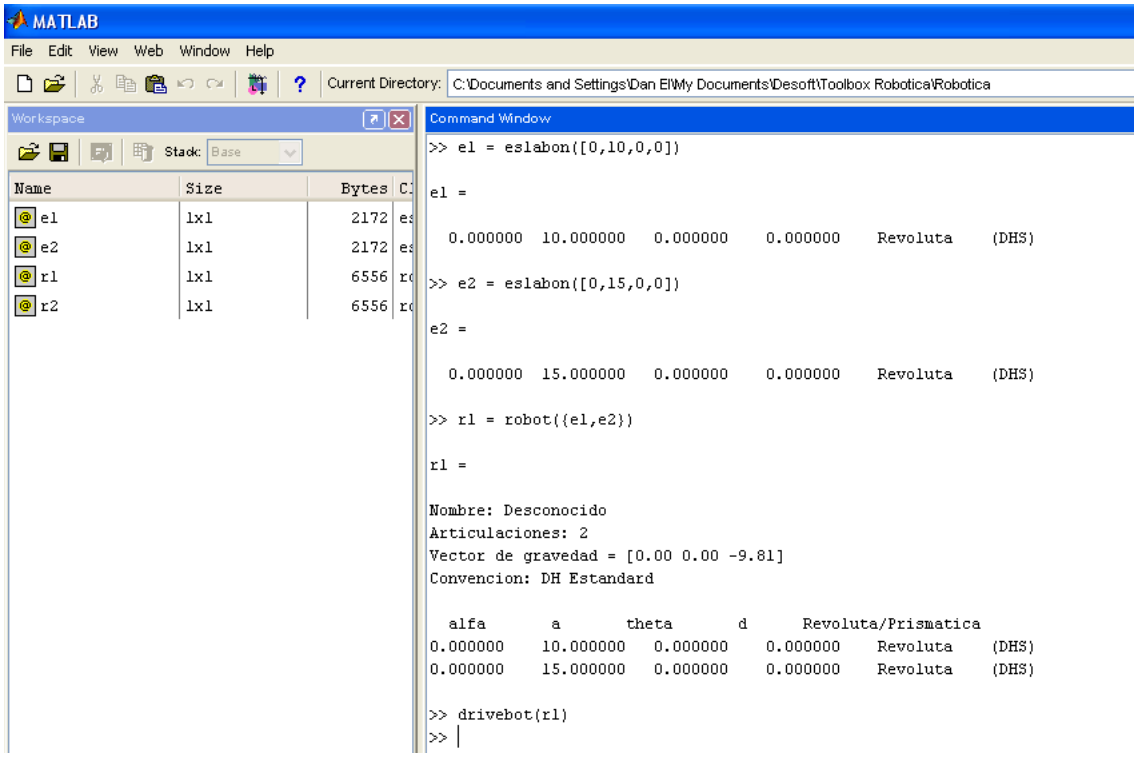


Figura 3.10: Uso de comando DRIVEBOT en MATLAB

En la figura 3.11 podemos observar 2 ventanas, una donde se despliega la imagen de nuestro robot manipulador con nombre "Desconocido" y la otra nos permite mover los parámetros de las articulaciones de nuestro "Desconocido". Lo mostrado no es arte de magia, si estas interesado en como se obtuvieron, sigue leyendo :).

3.3. Cinemática Directa

Con lo aprendido en las secciones anteriores podemos modelar cualquier robot manipulador, ahora es tiempo de probar o simular su comportamiento con tal de verificar que el modelo cumple con las especificaciones necesarias para cumplir su tarea u objetivo.

Una de las características que es siempre necesario evaluar es saber la posición y orientación de la herramienta o elemento terminal de nuestro robot manipulador en

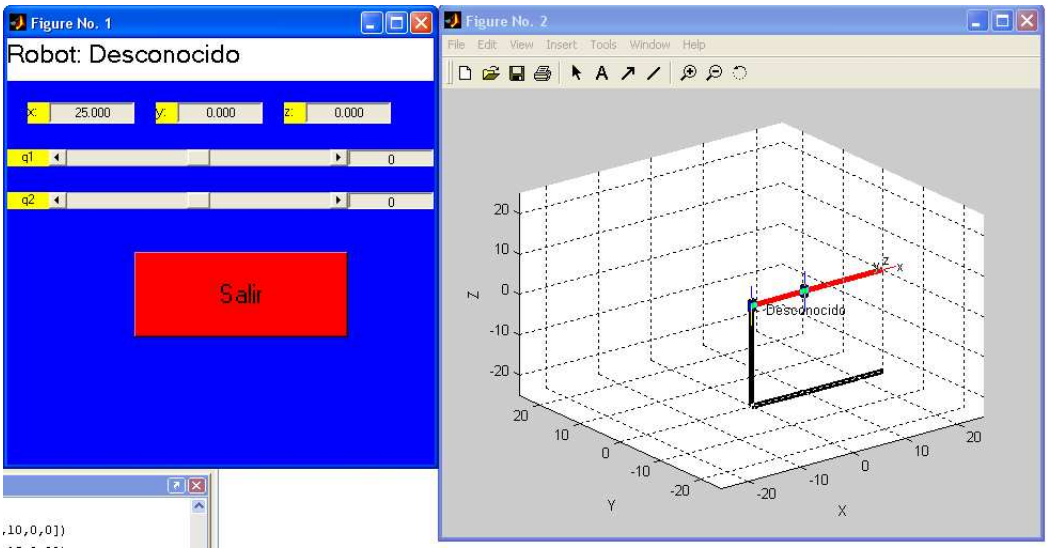


Figura 3.11: Resultado de comando DRIVEBOT en MATLAB

todo momento, y la técnica que nos permite resolver esta duda es la de *Cinemática Directa*.

Puesto que los sensores en cada una de las articulaciones de nuestro robot manipulador dan en cada instante que movimiento ha realizado la articulación, la cinemática directa nos podrá decir con esta información la posición y orientación de nuestro elemento terminal.

Sea $F : j \in \Re_n \rightarrow SE(3)$ una función que representa el resultado de la cinemática directa donde F regresa la posición y orientación del elemento terminal, j es un vector n-dimensional (donde n es el número de articulaciones) en el cual cada entrada es lo que se ha movido cada articulación (también llamado vector de variables articulables)y SE(3) es una matriz homogénea que aporta la información de posición y orientación del elemento terminal.

Puesto que el problema de cinemática directa se reduce a iterar el proceso de búsqueda de las transformaciones necesarias para pasar desde el sistema asociado a la base del robot hasta el extremo, F esta dada por la fórmula 3.3.23.

$$F(j) = \mathbf{T}^{base}_{extremo} = \mathbf{T}^{base}_{1} \cdot \mathbf{T}^{1}_{2} \cdot \mathbf{T}^{2}_{3} \cdots \mathbf{T}^{n-1}_{n} \cdot \mathbf{T}^{n}_{extremo} \tag{3.3.23}$$

Cada $\mathbf{T}^{i-1}_{i}$ representa la matriz de transformación homogénea que nos lleva del sistema de referencia i-1 al i (ver sección 3.2.5).

La cinemática directa se implemento en el toolbox de MATLAB a través del

comando:

Nombre del comando: CINEMA_DIR.

Descripción: Calcula la cinemática directa para cada conjunto de parámetros definidos. Arroja una matriz homogénea con la información de posición y orientación del elemento terminal o extremo del último eslabón.

Tipo de llamadas:

CINEMA_DIR(ROBOT, Q). En esta llamado al comando *ROBOT* es un objeto robot ya definido y Q es una matriz de $m \times n$ elementos, donde m es la cantidad de cinemáticas a calcular como secuencia de puntos en una trayectoria y n es el número de articulaciones.

Aplicando este comando a nuestro robot planar (mecanismo RR) haciendo girar $pi/2$ radianes ambas articulaciones obtenemos como resultado lo mostrado en la ventana de comandos de MATLAB, lo cual coincide con los parámetros asignados al manipulador a través del comando DRIVEBOT (Figura 3.12).

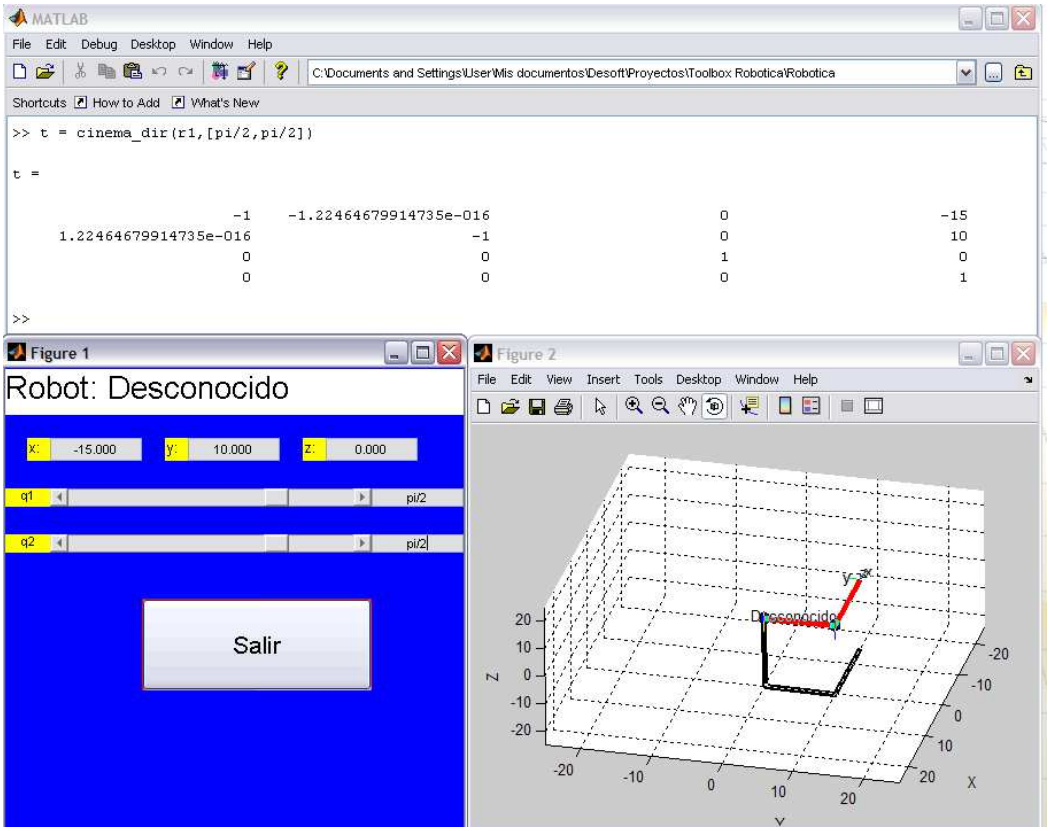


Figura 3.12: Ejemplo de comando CINEMA_DIR en MATLAB

3.4. Cinemática Inversa

En la sección anterior pudimos obtener la posición y orientación del elemento terminal a partir de la información aportada por las articulaciones; pero otro de los problemas mas importantes en el funcionamiento de un robot manipulador es saber cual es el desplazamiento de cada articulación para que el elemento terminal tenga una posición y orientación determinada; a este problema se le conoce como *Cinemática Inversa*.

El problema de cinemática inversa se dice que tiene solución si todos los conjuntos de valores de articulaciones para una posición y orientación pueden ser determinados usando un algoritmo; por lo que si hay multiples soluciones, el algoritmo debe ser capaz de encontrarlas todas.

Podemos formular el problema de cinemática inversa de la siguiente manera: Dada $H \in SE(3)$ una matriz homogénea, necesitamos encontrar $j \in \Re^n$ tal que $F(j) = H$. Ya que H define 6 restricciones, el problema esta bien definido si el numero de variables de articulación independientes es 6. Si $dim(j) < 6$, el problema esta sobrecondicionado y por lo general no es posible solución alguna. Por otra parte, si $dim(j) > 6$ se dice que el problema posee grados de libertad redundantes y para este caso puede haber un numero infinito de soluciones.

Hay 2 principales tipos de estrategias para este problema: solución numérica o solución analítica (solución cerrada o algebraica). Los métodos de solución analítica son generalmente mas eficientes que los numéricos (los numéricos presentan alta complejidad computacional), pero requieren que la estructura del manipulador cumpla ciertas características.

Los algoritmos de solución analítica solo son útiles para resolver problemas donde se involucran mecanismos con estructuras simples y con menos de 6 grados de libertad [**?**]; sin embargo si un manipulador de 6 DOF tiene 3 articulaciones revolutas consecutivas con sus ejes de articulación intersectandose en un solo punto, el problema de cinemática inversa en este manipulador tiene solución [**?**].

Ante la situación de que los métodos de solución analítica no es fácil aplicarlos a manipuladores con mas de 6 DOF, dentro de la medida de lo posible, dividimos la estructura de las articulaciones del manipulador en unidades cinemáticas para los cuales soluciones analíticas pueden ser encontradas y cuando una solución puramente analítica no puede ser obtenida usamos combinación de técnicas analíticas y numéricas con tal de mejorar rapidez y eficiencia al resolver el problema de cinemática inversa [**?**].

El método implementado dentro del toolbox de MATLAB para robótica esta basado en ecuaciones diferenciales y tiene la ventaja de ser eficiente.

3.4.1. Método basado en Ecuaciones Diferenciales

Aquí la idea que resuelve nuestro problema de cinemática inversa es que las velocidades de las articulaciones ($\dot{q}$) pueden ser integradas desde 0 a t_f para producir los ángulos de las articulaciones tales $F(q) = H$ (3.4.24).

$$q(t_f) = q(0) + \int_0^{t_f} \dot{q}(t)dt \tag{3.4.24}$$

Esta técnica también recibe el nombre de *Control resuelto de la taza de movimiento* [**?**] y en cuanto a implementación tiene la gracia de que si usamos un valor fijo de paso $\triangle t$ y un esquema de integración de primer orden, este método es prácticamente idéntico a cuando se usa métodos de Newton-Raphson para resolver $G(j) \equiv F(j) - H$.

Ahora bien, para hallar $\dot{q}(t)$, recordemos que dada la formulación de nuestro problema $F(j) = H$ y permitiéndonos un abuso de notación podemos redefinir la función como $F(q) = x$, donde $q = j$ y x es un vector que contiene los 6 elementos que determinan la orientación del elemento terminal así como los 3 elementos de posición.

Si diferenciamos con respecto al tiempo obtenemos $\dot{x} = J(q)\dot{q}$, donde $J(q)$ es el jacobiano de F respecto a q, por lo que $\dot{q}$ esta dada por la ecuación 3.4.25.

$$\dot{q} = J^{-1}(q)\dot{x} \tag{3.4.25}$$

Con tal de optimizar el método de resolución del problema de cinemática inversa optamos por trabajar con la seudo-inversa del jacobiano por lo que la fórmula iterativa para obtener $\dot{q}$ esta dada por 3.4.26.

$$\dot{q} = J^{+}(q)\triangle(F(q) - H) \tag{3.4.26}$$

La seudo-inversa del jacobiano esta determinada por $J^{+}(q)$ y $\triangle$ representa un cambio diferencial entre la posición y orientación del manipulador entre intervalos de tiempo de $F(q)$ y H.

Puesto que la técnica utilizada no funciona en el caso de que nuestro robot manipulador tenga menos de 6 DOF, en estos casos trabajamos con una mascara M que tiene 6 elementos y el cual activamos colocando un 1 donde vayamos a considerar una articulación y 0 donde no.

Recordemos que la solución obtenida depende de la condición inicial dada; la implementación en MATLAB se realizó en el comando *CINEMA_INV*.

Nombre del comando: CINEMA_INV.

Descripción: Regresa los valores de las articulaciones del robot manipulador *ROBOT* necesarios para llegar a la posición y orientación H. Por lo general la solución no es única y a la que se llega depende de la condición inicial.

Tipo de llamadas:

CINEMA_INV(ROBOT, H, Q, M). Q es la condición inicial con la cual se empezara a iterar. M es un vector de 1×6 que sirve de mascara cuando trabajamos con robot manipuladores de menos de 6 grados de libertad; en esta mascara se coloca un 1 activando la información de una articulación y 0 en caso contrario.

3.5. Dinámica

En esta sección estudiamos las fuerzas requeridas para producir movimiento. En si hay 2 problemas a resolver en dinámica de manipuladores; el primero, en el cual se nos proporciona un punto de trayectoria, con velocidad y aceleración requerida y deseamos encontrar el vector requerido de momentos de torsión de articulación (**Dinámica Inversa**) y el segundo problema llamado **Dinámica Directa** en el que necesitamos calcular como se moverá el mecanismo bajo la aplicación de un conjunto de momentos de torsión de articulación.

La dinámica inversa es útil para controlar el manipulador mientras que la dinámica directaa nos ayuda a simular el comportamiento del manipulador dándonos posición, velocidad y aceleración en cualquier instante de tiempo.

Para el estudio de la dinámica de manipuladores hemos de explicar el significado y el efecto de velocidad, aceleración, distribución de masa, fuerzas de gravedad y fricción. Si bien es cierto que existen otros efectos no considerados en esta modelación como efectos de flexión, los tratados en esta tesis nos dan un modelo bastante completo [**?**].

3.5.1. Velocidad

Al estudiar la velocidad de un robot manipulador, se analiza la velocidad lineal y angular de cuerpos rígidos y como estas se propagan a través del manipulador eslabón a eslabón.

La velocidad de un vector de posición V_Q (3.5.27)puede definirse como la **velocidad lineal** del punto en el espacio representado por el vector posición, y al igual

que con cualquier vector, un vector de velocidad se puede describir en términos de cualquier sistema de ejes, sin embargo, a pesar de estar asociado a un punto en el espacio, los valores numéricos que describen la velocidad de ese punto dependen de 2 ejes: una respecto a la que se realizó la diferenciación $\{B\}$ y otra respecto a la que expresó el vector de velocidad resultante $\{A\}$.

$$(V_Q^B)^A = \frac{d^A}{dt} Q^B = Rot_B^A V_Q^B \tag{3.5.27}$$

Para facilitar las descripciones, en vez de considerar la velocidad de un punto general relativa a un eje arbitrario, consideramos la velocidad de origen de un eje $\{C\}$, relativo a un eje de referencia universal $\{U\}$ y usamos la notación $v_C = V_{Corigen}^U$.

Dado un eje B unido a un cuerpo rígido, si deseamos describir el movimiento de un punto Q respecto a un eje A fijo, la velocidad lineal V_Q^A la obtenemos mediante la fórmula 3.5.28

$$V_Q^A = V_{Borigen}^A + Rot_B^A V_Q^B \tag{3.5.28}$$

Un componente mas que tenemos que agregar a nuestra información de velocidad es la **velocidad de rotación o velocidad angular** Ω. Esta velocidad describe el atributo de rotación de un cuerpo y su notación es semejante a la de la velocidad lineal; por ejemplo $(\Omega_B^A)^C$ es la velocidad angular del eje $\{B\}$ relativa a $\{A\}$ expresada en términos de $\{C\}$ y $\omega_C = \Omega_C^U$ es la velocidad angular de $\{C\}$ relativa al eje de referencia $\{U\}$.

En la velocidad angular se trata con 2 ejes ($\{A\}$ y $\{B\}$)cuyos origenes coinciden en todo momento y velocidad lineal cero; así para este tipo de velocidad la orientación del eje $\{B\}$ con respecto al eje $\{A\}$ esta cambiando en el tiempo. Si consideramos un vector Q constante desde la perspectiva del eje $\{B\}$ ($V_Q^B = 0$) obtenemos 3.5.29.

$$V_Q^A = Rot_B^A V_Q^B + \Omega_B^A \times Rot_B^A Q^B \tag{3.5.29}$$

De aquí que al considerar la velocidad lineal y de rotación simultaneamente para un vector fijo Q cuando los origenes no son coincidentes solo sumamos la ecuación de la velocidad lineal en el origen quedando 3.5.30.

$$V_Q^A = V_{Boriegen}^A + Rot_B^A V_Q^B + \Omega_B^A \times Rot_B^A Q^B \tag{3.5.30}$$

Puesto que un manipulador es una cadena de cuerpos, cada uno capaz de moverse

en relación a sus cuerpos adyacentes; para calcular las velocidades lineal y angular de los eslabones de un robot lo hacemos en orden empezando desde la base, de aquí que la velocidad del eslabón i+1 (velocidad lineal del origen del eslabón y velocidad de rotación del mismo) será la del eslabón i mas cualquier nuevo componente de velocidad que se haya agregado por la articulación i+1. De este análisis obtenemos las fórmulas recursivas para cada eje de eslabón i+1 rotacional(3.5.32) teniendo en cuenta que P es la matriz de transformación entre ejes:

$$\begin{aligned} \omega_{i+1}^{i+1} &= Rot_i^{i+1}\omega_i^i + \dot{\theta}_{i+1}\hat{Z}_{i+1} \\ v_{i+1}^{i+1} &= Rot_i^{i+1}(v_i^i v + \omega_i^i \times P_{i+1}^i) \\ \dot{\theta}_{i+1}\hat{Z}_{i+1} &= \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_{i+1} \end{bmatrix}^{i+1} \end{aligned} \tag{3.5.31}$$

En cuanto al caso de una articulación prismática tenemos las ecuaciones respecto al eje i+1 3.5.33:

$$\begin{aligned} \omega_{i+1}^{i+1} &= Rot_i^{i+1}\omega_i^i \\ v_{i+1}^{i+1} &= Rot_i^{i+1}(v_i^i v + \omega_i^i \times P_{i+1}^i) + \dot{d}_{i+1}\hat{Z}_{i+1} \\ \dot{d}_{i+1}\hat{Z}_{i+1} &= \begin{bmatrix} 0 \\ 0 \\ \dot{d}_{i+1} \end{bmatrix}^{i+1} \end{aligned} \tag{3.5.32}$$

3.5.2. Aceleración

En cualquier instante, los vectores de velocidad lineal y angular tienen derivadas llamadas aceleraciones lineal y angular respectivamente.

Dada la ecuación 3.5.29, podemos notar que su lado izquierdo describe como cambia Q_A en el tiempo, por lo que podemos reescribirla como 3.5.33.

$$\frac{d}{dt}\left(Rot_B^A Q_B\right) = Rot_B^A V_Q^B + \Omega_B^A \times Rot_B^A Q^B \tag{3.5.33}$$

Al diferenciar V_Q^A obtenemos expresiones para la aceleración de Q^B visto desde A cuando coinciden los origenes de A y B (3.5.34).

$$\dot{V}_Q^A = \frac{d}{dt}\left(Rot_B^A V_Q^B\right) + \dot{\Omega}_B^A \times Rot_B^A Q_B + \Omega_B^A \times \frac{d}{dt}\left(Rot_B^A Q^B\right) \tag{3.5.34}$$

Aplicando 3.5.33 en ambos lados de 3.5.34 y teniendo en cuenta el caso donde los origenes no coinciden (agregamos la aceleración lineal del origen de B) obtenemos la ecuación para la **aceleración lineal** 3.5.35.

$$\dot{V}_Q^A = \dot{V}_{Borigen}^A + Rot_B^A \dot{V}_Q^B + 2\dot{\Omega}_B^A \times Rot_B^A V_Q^B + \dot{\Omega}_B^A \times Rot_B^A Q^B + \Omega_B^A \times \left(\Omega_B^A \times Rot_B^A Q_B\right) \tag{3.5.35}$$

La ecuación anterior se reduce cuando Q^B es constante porque $V_Q^B = \dot{V}_Q^B = 0$ (caso de articulaciones giratorias).

Para la **aceleración angular** consideramos el caso en el que el eje B está girando relativamente a A con Ω_B^A y C esta girando de manera relativa a B con Ω_C^B. Para calcular Ω_C^A sumamos los vectores en el eje A y al derivar obtenemos 3.5.36.

$$\dot{\Omega}_C^A = \dot{\Omega}_B^A + \frac{d}{dt}\left(Rot_B^A \Omega_C^B\right) \tag{3.5.36}$$

Por lo que aplicando 3.5.33 nos queda la ecuación de la aceleración angular 3.5.37.

$$\dot{\Omega}_C^A = \dot{\Omega}_B^A + Rot_B^A \dot{\Omega}_C^B + \Omega_B^A \times Rot_B^A \Omega_C^B \tag{3.5.37}$$

3.5.3. Distribución de masa

El término momento de inercia sale a relucir cuando se da el movimiento giratorio de un cuerpo sobre un solo eje; en este caso necesitamos caracterizar la distribución de la masa del cuerpo rígido, por lo que usamos un **tensor de inercia**.

El tensor de inercia sólido rígido (I) se define como un tensor simétrico de segundo orden tal que la forma cuadrática a partir del tensor y la velocidad angular Ω da la energía cinética de rotación E_{cRot} (3.5.38).

$$E_{cRot} = \frac{1}{2}\Omega I \Omega^T = \frac{1}{2}\left(\Omega_x, \Omega_y, \Omega_z\right) \begin{pmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{pmatrix} \begin{pmatrix} \Omega_x \\ \Omega_y \\ \Omega_z \end{pmatrix} \tag{3.5.38}$$

Los elementos escalares de la matriz del tensor de inercia, también llamados momentos de inercia de masas están dados por:

$$\begin{aligned} I_{xx} &= \int\int\int_V (y^2+z^2)\rho dv \\ I_{yy} &= \int\int\int_V (x^2+z^2)\rho dv \\ I_{zz} &= \int\int\int_V (x^2+y^2)\rho dv \end{aligned} \tag{3.5.39}$$

Se llaman productos de inercia de masas a los elementos de la matriz de inercia a los elementos con índices mezclados y estos están dados por:

$$\begin{aligned} I_{xy} &= \int\int\int_V -xy\rho dv \\ I_{xz} &= \int\int\int_V -xz\rho dv \\ I_{yz} &= \int\int\int_V -yz\rho dv \end{aligned} \tag{3.5.40}$$

Es de mencionar que se dan las siguientes igualdades: $I_{xy} = I_{yx}$, $I_{yz} = I_{zy}$ y $I_{zx} = I_{xz}$. Dentro de las cosas a considerar en términos de las propiedades de los tensores están que (1) los momentos de inercia siempre deben ser positivos mientras que los productos de inercia pueden tener cualquier signo. (2) La suma de los 3 momentos de inercia es invariante bajo cambios de orientación. (3) Los valores propios de un tensor de inercia son los momentos principales para el cuerpo y los vectores propios asociados son los ejes principales.

3.5.4. Fuerza de gravedad

El efecto de carga de la gravedad influye en la dinámica del cuerpo y puede describirse haciendo que $v_0^0 = G$, siendo G la magnitud del vector de gravedad apuntando en dirección opuesta. Esto representaría una aceleración ficticia de la base del robot hacia arriba de 1g, lo cual produce el mismo efecto que la gravedad produciría en los eslabones.

3.5.5. Fricción

Los elementos de velocidad, aceleración y gravedad producen fuerzas que surgen de mecanismos de cuerpo rígido y si ponemos atención recordaremos que todos los mecanismos se ven afectados por fuerzas de fricción ¡las cuales no hemos incluido!.

Para reflejar la realidad de un dispositivo mecánico es importante aproximar la **fricción viscosa** en la que el momento de torsión debido a la fricción es proporcional a la velocidad del movimiento de la articulación; es decir $\tau_{friccion} = k\dot{\theta}$, donde k es una constante de fricción viscosa.

Otra posible aproximación es la **fricción de Coulumb**, la cual es constante salvo por una dependencia de signo de la velocidad de una articulación $\tau_{friccion} = csigno(\dot{\theta})$. El valor de c es la constante de fricción de Coulumb y se toma a cierta cantidad cuando $\dot{\theta} = 0$, pero a un menor valor cuando $\dot{\theta} \neq 0$.

Puesto que la articulación de un manipulador tenga fricción depende de lubricación, engranaje y demás situaciones, es adecuado incluir ambas fricciones es una sola ecuación ya que son probables (3.5.41).

$$\tau_{friccion} = k\dot{\theta} + csigno(\dot{\theta}) \tag{3.5.41}$$

El modelo mas completo pero que en esta tesis no se usa debido a la complejidad del mismo y a que no estamos evaluando engranajes y sus características, involucra la información de la posición de la articulación $\tau_{friccion} = f(\theta, \dot{\theta})$.

3.5.6. Dinámica Inversa

Dada la consideración de los eslabones como cuerpos rígidos, tenemos determinada su distribución de masa gracias a la información del centro de masas y el tensor de inercia del eslabón y para moverlos tenemos que acelerar y desacelerar. No me la van a creer, pero las fuerzas requeridas para estos cambios en la aceleración son función de la aceleración deseada y la distribución de masa del eslabón, por lo que recurrimos a la ecuación de Newton y a la ecuación de Euler para describir la manera en que se relacionan fuerzas, inercias y aceleraciones.

La ecuación de Newton nos dice que para un cuerpo rígido cuyo centro de masas tiene una aceleración $\dot{v}$, la fuerza esta dada por $F = m\dot{v}$, siendo m la masa total del cuerpo.

Por su parte en la ecuación de Euler si sobre un cuerpo rígido actúa un momento

N para producir velocidad angular ω y aceleración angular $\dot{\omega}$, estos elementos se relacionen mediante la ecuación $N = I\dot{\omega} + \omega \times I\omega$, donde I es el tensor de inercia del eslabón.

Procederemos a calcular los momentos de torsión que corresponden a una trayectoria dada de un manipulador conociendo posición, velocidad y aceleración de las articulaciones [**?**].

El procedimiento consiste en calcular de manera iterativa velocidad de rotación (3.5.42), aceleración rotacional de eslabón rotacional (3.5.43) o prismático (3.5.44), aceleración lineal de eslabón rotacional (3.5.45) o prismático(3.5.46), aceleración lineal del centro de masas de cada eslabón(3.5.47) así como la fuerza(3.5.48) y momento de torsión(3.5.49) inerciales que actúan en el centro de masas de cada eslabón en cualquier instante; empezando con el eslabón 1, avanzando sucesivamente hasta el eslabón n y tomando $\omega_0^0 = \dot{\omega}_0^0 = 0$.

$$\begin{aligned}
\omega_{i+1}^{i+1} &= Rot_i^{i+1}\omega_i^i + \dot{\theta}_{i+1}\hat{Z}_{i+1} && (3.5.42)\\
\dot{\omega}_{i+1}^{i+1} &= Rot_i^{i+1}\dot{\omega}_i^i + Rot_i^{i+1}\omega_i^i \times \dot{\theta}_{i+1}\hat{Z}_{i+1} + \ddot{\theta}_{i+1}\hat{Z}_{i+1} && (3.5.43)\\
\dot{\omega}_{i+1}^{i+1} &= Rot_i^{i+1}\omega_i^i && (3.5.44)\\
\dot{v}_{i+1}^{i+1} &= Rot_i^{i+1}\left[\omega_i^i \times P_{i+1}^i + \omega_i^i \times \left(\omega_i^i \times P_{i+1}^i\right) + \dot{v}_i^i\right] && (3.5.45)\\
\dot{v}_{i+1}^{i+1} &= Rot_i^{i+1}\left(\dot{\omega}_i^i \times P_{i+1}^i + \omega_i^i \times \left(\omega_i^i \times P_{i+1}^i\right) + \dot{v}_i^i\right) \\
&\quad +2\omega_{i+1}^{i+1} \times \dot{d}_{i+1}\hat{Z}_{i+1}^{i+1} + \ddot{d}_{i+1}\hat{Z}_{i+1}^{i+1} && (3.5.46)\\
\dot{v}_{Ci}^i &= \dot{\omega}_i^i \times P_{Ci}^i + \omega_i^i \times \left(\omega_i^i \times P_{Ci}^i\right) + \dot{v}_i^i && (3.5.47)\\
F_i &= m\dot{v}_{Ci} && (3.5.48)\\
N_i &= I_i\dot{\omega} + \omega_i \times I_i\omega_i && (3.5.49)
\end{aligned}$$

Como segundo paso tenemos el calcular los momentos de torsión de articulación que producirán las fuerzas calculadas anteriormente y los momentos de torsión netos para cada tipo de articulación prismática o rotacional. Si f_i es la fuerza ejercida en el eslabón i por el vínculo i-1, al sumar las fuerzas que actúan sobre el eslabón obtenemos la ecuación de balance de fuerzas F_i^i(3.5.50). Y si n_i es el momento de torsión ejercido en el eslabón i por el i-1 y sumar los momentos de torsión obtenemos la ecuación de momentos de torsión N_i^i.

$$F_i^i = f_i^i - Rot_{i+1}^i f_{i+1}^{i+1} \tag{3.5.50}$$

$$N_i^i = n_i^i - n_{i+1}^i + (-P_{Ci}^i) \times f_i^i - (P_{i+1}^i - P_{Ci}^i) \times f_{i+1}^i \tag{3.5.51}$$

Con lo anterior podemos reordenar las ecuaciones de fuerza (3.5.52) y momento de torsión (3.5.53) para el algoritmo y finalmente el momento de torsión rotacional esta dado por la ecuación 3.5.54 mientras que en un eslabón prismático es 3.5.55.

$$
\begin{aligned}
f_i^i &= Rot_{i+1}^i f_{i+1}^{i+1} + F_i^i && (3.5.52)\\
n_i^i &= N_i^i + Rot_{i+1}^i n_{i+1} + P_{Ci}^i \times F_i^i + P_{i+1}^i \times Rot_{i+1}^i f_{i+1}^{i+1} && (3.5.53)\\
\tau_i &= (n_i^i)^T \hat{Z}_i^i && (3.5.54)\\
\tau_i &= (f_i^i)^T \hat{Z}_i^i && (3.5.55)
\end{aligned}
$$

Las anteriores son las ecuaciones que tenemos que evaluar eslabón por eslabón pero empezando por el n-ésimo y trabajando hacia la base del robot se conocen como iteraciones entrantes de fuerza.

3.5.7. Dinámica Directa

Para la dinámica directaa es necesario expresar las fórmulas anteriores de una forma mas eficiente, en la cual se aprecie la estructura de las ecuaciones(3.5.56).

$$\tau = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta) + F(\Theta, \dot{\Theta}) \qquad (3.5.56)$$

En la ecuación anterior $M(\Theta)$ es la matriz de masas de dimensión $n \times n$ del manipulador, $V(\Theta, \dot{\Theta})$ es un vector de $n \times 1$, $G(\Theta)$ es un vector de $n \times 1$ de términos de gravedad y $F(\Theta, \dot{\Theta})$ es el vector de $n \times 1$ de elementos de fricción.

Para simular el movimiento de un manipulador hacemos uso de la ecuación dinámica para la aceleración(3.5.57) la cual tenemos que integrar numéricamente para calcular posición y velocidades futuras. Sin embargo mediante el uso de la formulación de Newton-Euler vista en la dinámica inversa se resuelve de manera eficiente.

$$\ddot{\Theta} = M^{-1}(\Theta)[\tau - V(\Theta, \dot{\Theta}) - G(\Theta) - F(\Theta, \dot{\Theta}) \qquad (3.5.57)$$

3.6. Generación de rutas y trayectorias

Los robots manipuladores son construidos para realizar tareas complejas que requieren movimientos que podemos considerar altamente no-lineales; de aquí que es-

pecificar el movimiento deseado para lograr un objetivo en particular es una tarea bastante difícil y que abordaremos en esta sección.

Si bien esta sección se enfoca en los métodos para calcular una trayectoria que describa el movimiento deseado de un manipulador en un espacio multidimensional, primero hemos de identificar la diferencia entre ruta y trayectoria.

Aunque el nombre de rutas y trayectorias tienden a entenderse como sinónimos, entendemos estos conceptos de la siguiente manera:

Ruta : Representación geométrica en el espacio 3-dimensional de los movimientos de un robot.

Trayectoria : Historial en el tiempo de la posición, velocidad y aceleración de cada grado de libertad de nuestro robot.

Combinando una representación visual(rutas) y la generación de trayectorias, podemos tener un buen entendimiento del comportamiento de un robot. Las rutas o trayectorias por lo general quedan determinadas por el movimiento deseado del elemento terminal de un robot manipulador.

3.6.1. Requerimientos

Dentro de los requerimientos básicos para determinar trayectorias están:

1.- Una trayectoria debe ser especificada con respecto a un sistema de referencia. Esto es debido a que la especificación de una trayectoria o ruta depende de la tarea a realizar y debe ser independiente del robot.

2.- Una trayectoria debe ser suave (primera y segunda derivada continuas). Esto debido a que evita desgaste de las articulaciones, motores y reductores además de que reduce fuerzas de impulso cuando se aplica una carga al elemento terminal y evita dificultades en términos de agarre en el caso de un gripper (mano robótica), daños a este mismo y al objeto que se maneja en el elemento terminal.

3.- Una trayectoria debe satisfacer los requerimientos de tiempo para una tarea determinada.

3.6.2. Restricciones

Dentro de las restricciones que debemos tomar en cuenta al momento de diseñar rutas o trayectorias están:

1.- Restricciones de movimiento espacial. Las restricciones de movimiento es una manera de incluir mas detalle en la descripción de las rutas del manipulador. El objetivo de estas restricciones puede ser evitar obstáculos, colisiones o arrinconamientos donde sea muy difícil maniobrar. Una forma de resolver este problema es a través de una secuencia de *puntos vía* deseados (puntos intermedios entre las posiciones inicial y final), de aquí que el completar un movimiento el elemento terminal debe pasar a través de un conjunto de posiciones y orientaciones intermedias (los puntos vía).

2.- Restricciones de movimiento temporal. En este tipo de restricciones la especificación del movimiento incluye el hecho de que entre cada par de puntos via consecutivos haya un tiempo determinado.

3.6.3. Los métodos

Así, siendo analíticos, podemos determinar un número de pasos para planear una trayectoria de un robot.

1 : Especificación de configuración inicial y final para un movimiento dado.

2 : Especificación de puntos via si son requeridos.

3 : Conversión del conjunto de puntos inicial, final e intermedios en términos de variables de articulación, este paso es opcional, pero requerido si generamos trayectorias en el espacio de las articulaciones (usamos cinemática inversa).

4 : Generación de rutas suaves que empiecen en el punto inicial, pasen a través de los puntos via y terminen en la posición final dada.

5 : Generación de características de velocidad adecuadas para la ruta.

6 : Calculo de fuerzas en la articulación y torques a lo largo de la trayectoria para verificar el desempeño del manipulador.

7 : Generación de un conjunto de datos para el control del manipulador.

Del paso 3 en adelante, los detalles del método dependen del método elegido para generar las trayectorias.

La generación de rutas o trayectorias que manejamos en esta tesis se pueden agrupar en 2 esquemas:

1.- Esquemas en el espacio de las articulaciones. Es un método de generación de rutas en los que las formas de estas (en tiempo y espacio) se describen en términos de funciones de ángulos de articulación. En este esquema, cada punto de la ruta se especifica en términos de una posición y orientación deseadas del elemento terminal; así, cada uno de los puntos via se traduce en un conjunto de ángulos de articulación deseados mediante la aplicación de cinemática inversa. Posteriormente procedemos a encontrar una funciones suaves e independientes para cada una de las articulaciones que pasan a través de los puntos via y terminan en el punto final (objetivo).

Una de las características de este esquema de trabajo es que el tiempo requerido por segmento es el mismo para cada articulación, con lo que todas las articulaciones llegarán al punto via al mismo tiempo generando así la posición cartesiana deseada en cada punto via.

Con todas estas características podemos formar rutas, que pueden ser simples en el espacio de articulación pero complejas en el espacio cartesiano.

Dentro de los métodos implementados en nuestro toolbox de MATLAB dentro del esquema de articulaciones están: Polinomios cúbicos, lineal con mezclas parabólicas, asíncrona punto a punto y síncrona punto a punto.

2.- Esquemas en el espacio cartesiano. Estos son métodos de generación de rutas en las cuales las formas de las rutas se describen en términos de funciones que calculan la posición y la orientación cartesianas como funciones del tiempo, de esta manera podemos especificar la forma espacial de la ruta (líneas rectas, circulares, senoidales, etc.).

Cada punto de la ruta se especifica en términos de una posición y una orientación deseadas del elemento terminal; las funciones que se unen para formar una trayectoria son funciones del tiempo que representan variables cartesianas.

Los esquemas cartesianos requieren de mayor poder de computo para ejecutarse ya que en tiempo de ejecución debe resolverse la cinemática inversa a la frecuencia de actualización de ruta; esto es, una vez que se general la ruta en el espacio cartesiano, como último paso debe realizarse el cálculo de la cinemática inversa para calcular los ángulos de articulación deseados.

Es de mencionar que las rutas cartesianas son propensas a problemas relacionados con el espacio de trabajo y las singularidades: puntos intermedios inalcanzables, velocidades de articulación altas cerca de una singularidad, inicio y destino alcanzables mediante distintas soluciones.

3.6.4. Polinomios cúbicos entre 2 puntos

Consideremos llevar el elemento terminal desde su posición inicial hasta una destino en cierta cantidad de tiempo. Usando los pasos para determinación de trayectoria, comencemos por usar cinemática inversa con los puntos inicial y final. Lo que necesitamos ahora es una función para cada articulación, cuyo valor en t_0 sea la posición inicial de la articulación y en t_f sea la posición deseada de esa articulación. Puesto que hay muchas funciones suaves que cumplen estas características, nos enfocaremos en aquellas que nos generan un movimiento uniforme y ante esto necesitamos 2 restricciones de valores inicial $\theta(0) = \theta_0$ y final $\theta(t_f) = \theta_f$ y que las velocidades inicial y final sean cero (continuidad en velocidad) $\dot{\theta}(0) = 0$ y $\dot{\theta}(t_f) = 0$ respectivamente. Usando un polinomio cubico obtenemos el siguiente conjunto de ecuaciones:

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$
$$\dot{\theta}(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$
$$\ddot{\theta}(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

Combinando las ecuaciones anteriores con las restricciones nos queda un sistema de 4 ecuaciones y 4 incógnitas cuya solución para las a_i es:

$$a_0 = \theta_0$$
$$a_1 = 0$$
$$a_2 = \frac{3}{t_f^2}(\theta_f - \theta_0)$$
$$a_3 = -\frac{2}{t_f^3}(\theta_f - \theta_0)$$

3.6.5. Polinomios cúbicos para ruta con puntos via

En este tipo de método deseamos pasar a través de un punto via sin detenernos, por lo que necesitamos generalizar la forma en que ajustaremos ecuaciones cúbicas a las restricciones de las rutas con tal de que conecten entre si los valores de los puntos via para cada articulación de manera uniforme.

Si se conocen las velocidades deseadas de las articulaciones en los puntos via, podemos construir polinomios cúbicos como en la sección anterior, sin embargo, ahora las restricciones de velocidad en cada extremo no son iguales a cero $\dot{\theta}(0) = \dot{\theta}_0$ y $\dot{\theta}(t_f) = \dot{\theta}_f$ de aquí que las 4 ecuaciones que describen el polinomio cubico son:

$$\theta(0) = a_0$$
$$\theta_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3$$
$$\dot{\theta}(0) = a_1$$
$$\dot{\theta}_f = a_1 + 2a_2 t_f + 3a_3 t_f^2$$

Al resolver las ecuaciones anteriores para a_i obtenemos:

$$a_0 = \theta_0$$
$$a_1 = \dot{\theta}_0$$
$$a_2 = \frac{3}{t_f^2}(\theta_f - \theta_0) - \frac{2}{t_f^2}\dot{\theta}_0 - \frac{1}{t_f}\dot{\theta}_f$$
$$a_3 = -\frac{2}{t_f^3}(\theta_f - \theta_0) + \frac{1}{t_f^2}\left(\dot{\theta}_f + \dot{\theta}_0\right)$$

Si tenemos las velocidades de articulación deseadas en cada punto via, entonces solo aplicamos las ecuaciones anteriores a cada segmento para encontrar los polinomios cúbicos requeridos. Existen varias formas para especificar la velocidad deseada en los puntos via:

1 : Velocidades especificadas por el usuario.

2 : Uso de heurísticas.

3 : Selección automática de tal forma que la velocidad y aceleración sea continua en los puntos via.

En esta parte optamos por resolver el problema de manera automática, buscando que la velocidad y aceleración sea continua en los puntos via, el algoritmo utilizado es el interpolante de trazador cúbico natural y sujeto (dependiendo de los gustos del usuario) [**?**].

3.6.6. Lineal con mezclas parabólicas

Una opción de ruta es interpolar linealmente para realizar un movimiento desde la posición de la articulación actual hasta la posición final (el elemento terminal no se mueve en general en linea recta), sin embargo esta interpolación lineal directa haría que la velocidad fuera discontinua al inicio y al final del movimiento, por lo que para crear una ruta uniforme con posición y velocidad continuas, empezamos con la función lineal pero agregamos una región de mezcla parabólica en cada punto de la ruta (en esta mezcla la aceleración es constante).

Supondremos que las secciones parabólicas de la ruta tendrán la misma duración y que la respuesta sera simétrica sobre el punto intermedio en el tiempo t_m y sobre el punto intermedio de la posición θ_m. La velocidad al final de la región de mezcla debe ser igual a la velocidad de la sección lineal:

$$\ddot{\theta} t_b = \frac{\theta_m - \theta_b}{t_m - t_b} \tag{3.6.58}$$

donde θ_b es el valor de θ al final de la región de mezcla y $\ddot{\theta}$ es la aceleración durante la región de mezcla. Si el valor de θ_b se obtiene mediante

$$\theta_b = \theta_0 + \frac{1}{2}\ddot{\theta} t_b^2 \tag{3.6.59}$$

al combinar las ecuaciones 3.6.58 y 3.6.59 y teniendo en cuenta que si t es el tiempo de duración deseada del movimiento $t = 2t_m$, obtenemos

$$\ddot{\theta} t_b^2 - \ddot{\theta} t t_b + (\theta_f - \theta_0) = 0 \tag{3.6.60}$$

Dado cualquier valor para θ_f, θ_0 y t, podemos seguir cualquiera de las rutas dadas con base en las elecciones de $\ddot{\theta}$ y t_b. Por lo general se elige una aceleración $\ddot{\theta}$ y se resuelve (3.6.60)la ecuación para obtener t_b (3.6.61).

$$t_b = \frac{t}{2} - \frac{\sqrt{\ddot{\theta}^2 t^2 - 4\ddot{\theta}(\theta_f - \theta_0)}}{2\ddot{\theta}} \tag{3.6.61}$$

Para que exista la solución anterior, la aceleración debe cumplir:

$$\ddot{\theta} \geq \frac{4(\theta_f) - \theta_0}{t_2} \tag{3.6.62}$$

3.7. Gripper

Con tal de que nuestra herramienta pueda simular agarres y las fuerzas que aplica en estas actividades, se implementó la simulación de un elemento terminal de 2 dedos [**?**] también llamado *gripper* 3.13.

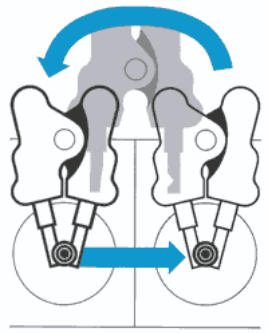


Figura 3.13: Gripper de 2 dedos

En cuanto a la simulación de las fuerzas de agarre, es necesario un control óptimo de la fuerza para asegurar la integridad de los objetos que manipulara ya que la gravedad, la aceleración del elemento terminal o disturbios externos pueden provocar que el objeto que se esta cargando con el gripper resbale si no se aplica la fuerza adecuada; por otra parte, si esta fuerza aplicada por el gripper es muy grande o cercana a los limites del actuador, no es posible tener un agarre fuerte sin riesgo de romper o aplastar el objeto que se trabaja (¡Imagínese lo que pasaría si fueran tomates!). Ante esto, es importante limitar la máxima aceleración que el gripper puede tolerar sin poner en riesgo el objeto.

Si el controlador del gripper tiene información acerca de los disturbios, la acción de agarre del gripper será mas estable, además, si el control puede poner limites a estos disturbios, la operación de agarre puede mejorar aun mucho mas. En esta situación, es necesario elaborar un algoritmo para limitar la aceleración vertical (Estimado lector, agradecería, si fuera usted tan amable de tener muy en cuenta el párrafo que acaba de leer porque es la razón de que hayamos incluido la sección 3.7.3).

Controles difusos han sido propuestos para resolver el problema de agarre de un gripper ya que este tipo de controles pueden trabajar con variables cuantitativas y cualitativas [**?**].

Una de las ventajas de aplicar un modelo difuso es que no necesita el modelo matemático del gripper [**?**]. Para efectos de esta tesis se implementó la simulación del elemento terminal así como un control jerárquico difuso para resolver el problema de agarre.

3.7.1. Modelo del gripper

Como habíamos mencionado, para lograr un agarre satisfactorio, la fuerza optima es requerida para evitar el riesgo de que el objeto resbale del gripper o que pueda ser dañado por la fuerza aplicada.

El problema de definir una fuerza de agarre requerida es importante y puede ser definido como un problema de optimización de lo mas interesante [**?**].

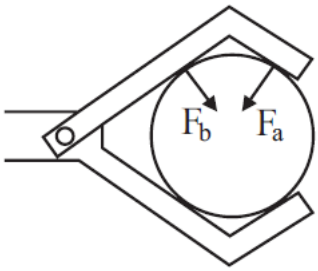


Figura 3.14: Gripper de agarre de fuerza con configuración cerrada

El gripper que hemos de modelar es de 2 dedos, este tipo de gripper es bastante utilizado porque su dinámica es muy sencilla. La fuerza que el gripe puede aplicar para prevenir deslizamientos depende de varios factores; por lo que primero debemos considerar su configuración para determinar en que dirección el objeto resbala. Por ejemplo, el gripper de agarre de fuerza de configuración cerrada solo tiene la aceleración vertical como fuerza capaz de inducir deslizamientos del objeto (Figura 3.14) mientras que en el gripper de agarre de configuración cerrada parcial (Figura 3.15), la aceleración horizontal y/o la fuerza centrífuga provocada por la velocidad angular del gripper pueden inducir que el objeto que se agarra resbale, de aquí que trabajemos con el primer gripper.

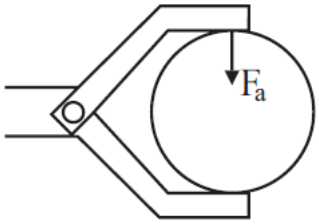


Figura 3.15: Gripper de agarre con configuración cerrada parcial

La fuerza que el gripper de agarre de fuerza de configuración cerrada debe aplicar para evitar el deslizamiento vertical del objeto depende del ángulo θ_g entre la superficie de agarre y la horizontal y el coeficiente de fricción μ entre la superficie de contacto

del gripper y la superficie del objeto. La figura 3.16 ilustra como se mide θ_g mientras que la figura 3.17 muestra el diagrama de un gripper sosteniendo una carga fuera del eje del centro de gravedad. Con esta información la fuerza requerida para prevenir desplazamiento vertical esta dado por la ecuación 3.7.63 donde m es la masa de carga, g es la aceleración debida a la gravedad, d es el ancho de la zona de presión y a es la aceleración vertical del gripper.

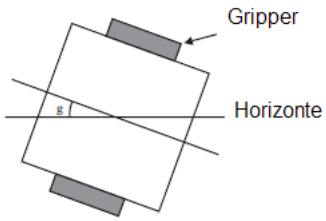


Figura 3.16: Medición de ángulo θ_g

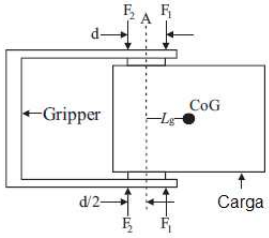


Figura 3.17: Gripper con carga

$$FV_T = \frac{2m(gsen(\theta_g) + a)L_g}{\mu d} + \frac{2m(gsen(\theta_g) + a)}{\mu} \tag{3.7.63}$$

Notemos que la aceleración vertical esta medida a lo largo del eje Z del elemento terminal.

Cuando el gripper en modelado tiene una forma cerrada parcial puede tener deslizamiento horizontal y vertical; la fuerza requerida para evitar deslizamiento horizontal esta dado por 3.7.64, donde R es la longitud del brazo de agarre y ω es la velocidad angular del elemento terminal.

$$FH_T = \frac{mR\omega^2}{\mu} \tag{3.7.64}$$

El resultado de la simulación del gripper también se hizo en MATLAB (3.18).

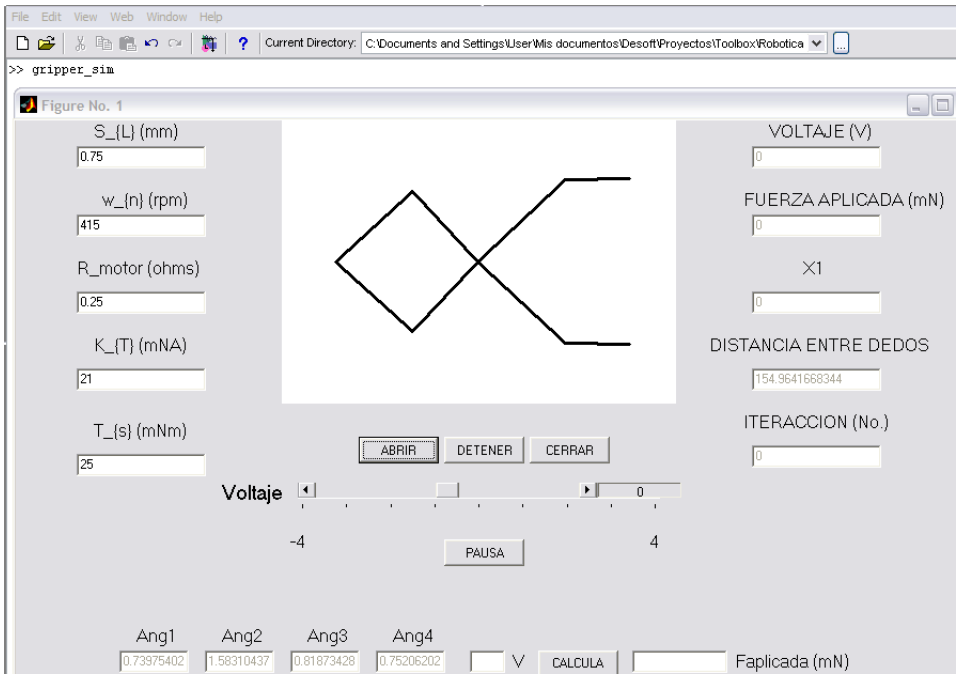


Figura 3.18: Simulación de gripper

3.7.2. Implementación del control difuso

La mayoría de los grippers usan información de los sensores (de fuerza) para mejorar el control, sin embargo si la información de un sensor de deslizamiento es utilizado, el agarre puede ser mejorado con una fuerza mínima en las yemas de los dedos del gripper [?] y esto reduce el riesgo de deformar el objeto.

Como estamos trabajando con un gripper de agarre de fuerza de configuración cerrada, solo debemos preocuparnos de los efectos de la aceleración vertical, pero si el gripper tiene información de estos efectos, el controlador puede compensar los efectos de la aceleración para tener un agarre mas estable.

Para la implementación del control difuso hay que tener en cuenta su problema de dimensionalidad pues el número de reglas empleadas incrementa exponencialmente cuando el número de reglas se incrementa,por lo que debemos buscar un punto intermedio entre simplicidad y alta eficiencia [?].

El problema de dimensionalidad puede ser solucionado combinando el modelo difuso con un modelo jerárquico como los modelos parsimónicos [?].

En la figura 3.19 se muestra el modelo parsimónico para nuestro problema. Las entradas para el control difuso son: cantidad de deslizamiento del objeto, la fuerza

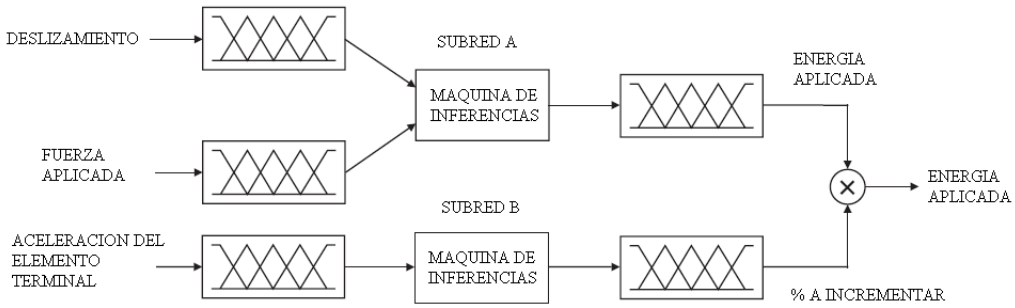


Figura 3.19: Modelo parsimónico para un control difuso de 3 entradas.

aplicada y la aceleración vertical del elemento terminal. Como siguiente nivel se encuentran 2 salidas parciales: el voltaje del motor requerido (salida de subred A que involucra fuerza aplicada vs. deslizamiento) y el porcentaje de incremento de voltaje (salida de subred B que involucra aceleración vertical), se dan estas salidas porque el torque del motor es directamente proporcional al voltaje usado por el motor. La multiplicación de estas salidas parciales da la salida general del sistema (energía aplicada).

El rango operacional de cada una de estas variables fue determinado experimentalmente con ayuda del simulador realizado en la sección anterior.

Las variables lingüísticas usadas para la característica de **deslizamiento** son: Cero(C), Casi Nada(CN), Poco(P), Medio(M) y Grande(G).

En cuanto al conjunto difuso de la **fuerza aplicada**, las variables son: Cero(C), Poco(P), Medio(M) y Grande(G).

El **voltaje** de motor aplicado tiene como conjuntos miembro: Cero(C), Muy Poco(MP), Poco(P), Medio(M), Grande(G), Muy Grande(MG) y Muy Muy Grande(MMG). Esta característica tiene mas elementos para poder tener una salida mas suave.

	Fuerza de agarre aplicada			
Deslizamiento	C	P	M	G
C	MMG	M	C	NMP
CN	MG	M	MP	C
P	G	L	M	P
M	MG	MG	G	M
G	MMG	MG	G	M

Figura 3.20: Base de reglas para subred A.

Dados los requerimientos de deslizamiento contra fuerza aplicada, se aplica una cantidad de voltaje determinado, por lo que las reglas base para la subred A están

dadas por la tabla 3.20, por ejemplo *SI deslizamiento es Cero Y fuerza aplicada es Cero ENTONCES aplicar voltaje Muy Muy Grande.*

Hablando de la **aceleración vertical**, esta tiene los siguientes miembros: Negativo Grande(NG), Negativo Medio(NM), Negativo Poco(NP), Cero(C), Poco(P), Medio(M) y Grande(G).

Por último el **incremento de porcentaje** esta dado por: Cero(C), Poco(P), Medio(M) y Grande(G).

	Aceleracion vertical del elemento terminal						
% Incremento	NG	NM	NP	C	P	M	G
C	---	---	---	✓	---	---	---
P	---	---	✓	---	✓	---	---
M	---	✓	---	---	---	✓	---
G	✓	---	---	---	---	---	✓

Figura 3.21: Base de reglas para subred B.

La tabla 3.21 asignada a la subred B son las reglas base a la respuesta del efecto de la aceleración vertical, por ejemplo *SI aceleración es Cero ENTONCES incremento de porcentaje es Cero.*

Las funciones miembro son triangulares para todas las señales con tal de aportar simplicidad y economía computacional.

Como punto de comparación podemos poner a un sistema difuso que solo contempla la tabla de la subred A pues este sistema tiene la ventaja de tener menos reglas (20 a comparación del parsimonio que entre las 2 tablas junta 27 reglas).

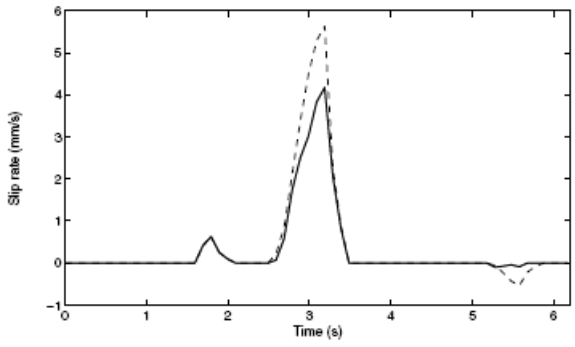


Figura 3.22: Comparación de resultados simulados de la taza de deslizamiento entre 2 sistemas difusos: con información de la aceleración (linea solida) y sin ella (linea punteada).

Implementado estas condiciones los resultados de deslizamiento se muestran en la figura 3.22 donde la linea continua se muestra el desempeño del sistema parsimónico

y la linea punteada es la del sistema difuso que solo considera la tabla 3.20.

Notemos en la figura 3.23 el comportamiento de la aceleración vertical.

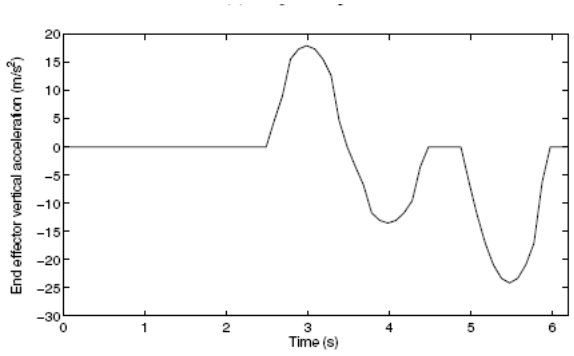


Figura 3.23: Comportamiento del desplazamiento vertical en el gripper

3.7.3. Control jerárquico difuso

Recordemos que el control del manipulador basa su toma de decisiones en información imprecisa percibida por los sensores. Podemos observar que la información dada de esta manera es jerárquica por si misma y nos lleva a la necesidad de un método de control jerárquico [**?**] y este es una implementación que se realiza de la siguiente manera: dividir el problema de control en varios sub-problemas mas simples [**?**].

El control parsimónico difuso reacciona a los disturbios incrementando el voltaje aplicado, pero hay un limite a la fuerza que el gripper puede aplicar al objeto: limite superior porque puede romper el objeto si aplica demasiada fuerza y limite inferior porque puede deslizarse el objeto si no aplica la fuerza suficiente.

Cuando el gripper no puede aplicar mas fuerza significa que no se puede dar mas aceleración y esto, damas y caballeros son los límites que habíamos estado buscando para un control jerárquico eficiente.

El diagrama del control jerárquico difuso esta dado en la figura 3.24. El control difuso es el mismo descrito en la sección anterior (el parsimónico) mientras el sistema limitante de aceleración es un control difuso de 2 entradas, la fuerza aplicada y el voltaje aplicado y una salida el porcentaje permitido de incremento de aceleración.

La fuerza aplicada y el voltaje aplicado fueron seleccionados como entradas porque estos 2 factores indican si es posible tener un agarre fuerte sin riesgo de dañar el objeto o que se deslice del gripper.

La nueva variable difusa (**porcentaje permitido de incremento de acel-**

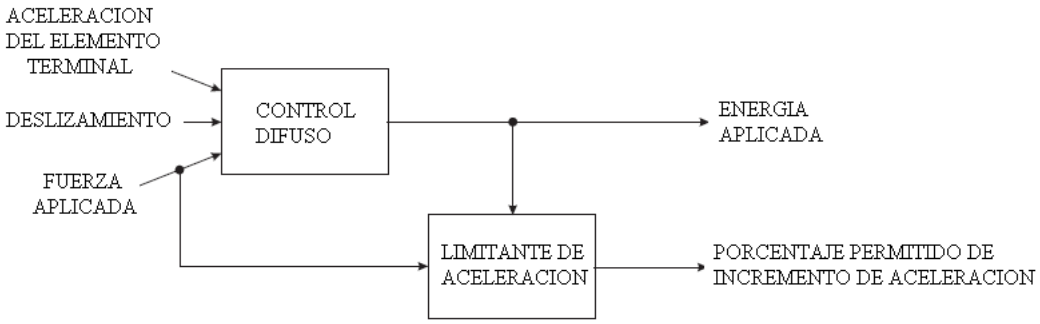


Figura 3.24: Diagrama del control jerárquico difuso.

eración) tiene como conjuntos miembros: Cero(C), Poco(P), Medio(M), Grande(G) y Muy Grande(MG), Las reglas base están dadas por la tabla 3.25.

	Voltaje aplicado al motor							
Fuerza de Agarre	NMP	C	MP	P	M	G	MG	MMG
C	MG	MG	MG	MG	MG	MG	MG	MG
P	G	G	G	G	G	M	M	M
M	G	G	M	M	M	P	P	P
G	P	P	P	C	C	C	C	C

Figura 3.25: Reglas base para el delimitador de aceleración difuso.

Para comparar la efectividad del control jerárquico difuso se trabaja contra el modelo parsimónico de la sección anterior. Enfocándonos en el deslizamiento del objeto podemos notar que el sistema jerárquico lo reduce casi por completo mostrando solo deslizamiento al momento de que se agarra por primera vez el objeto (evidencia basada en 10 experimentos donde se cambio la fricción, la masa del objeto y los limites de capacidad del motor), ejemplo de esto es la figura 3.26.

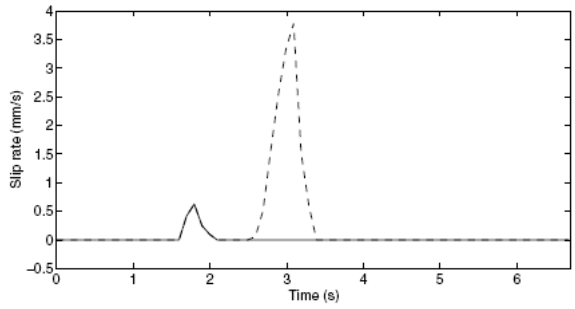


Figura 3.26: Comparación de resultados simulados de la taza de deslizamiento entre 2 sistemas difusos: jerárquico (linea solida) y parsimónico (linea punteada).

En cuanto a la aceleración vertical, su comportamiento (Figura 3.27) tiene a

cambiar varias veces (3 en el ejemplo mostrado), lo cual se traduce en reducción del riesgo de deslizamientos.

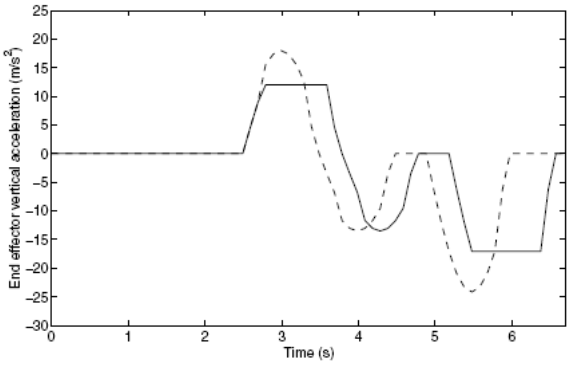


Figura 3.27: Comparación de resultados simulados de la taza de aceleración vertical entre 2 sistemas difusos: jerárquico (linea solida) y parsimónico (linea punteada).

Si bien es cierto que el sistema jerárquico tiene 59 reglasa diferencia del parsimonio que tiene 27, pero debido a la jerarquía, sigue siendo un sistema fácil de implementar y de fácil entendimiento.

Capítulo 4

Optimización de diseño para robots manipuladores

El uso de robots industriales en diferentes campos de la tecnología es mas común e importante con cada año que pasa y dentro de esta área es importante ser capaz de mejorar su eficiencia en términos de consumo de energía y precisión en las tareas que desempeñan. Optimizar siempre es importante y ya que que nosotros podemos modelarlos, en esta sección nos enfocaremos a mejorar nuestros diseños en base a ciertas características determinadas.

Hay varias técnicas de optimización de robots manipuladores, dependiendo de la característica a trabajar como por ejemplo diseño óptimo basado en tareas específicas con restricciones cinemáticas dinámicas y de estructura, reducción de grados de libertad o síntesis dimensional de mecanismos (longitud de eslabones).

Es dentro de este capítulo donde trabajaremos estos 3 enfoques para optimizar robots manipuladores; vale la pena mencionar que estos enfoques se resuelven mediante técnicas evolutivas, en particular, algoritmos genéticos y que son implementados como parte de nuestra herramienta de simulación de robots manipuladores [**?**].

Le sección 4.1 describe la técnica de optimización mediante síntesis de mecanismos. En la sección siguiente, la 4.2 tratamos de optimizar robots manipuladores tratando de reducir sus grados de libertad y con ciertas restricciones. Finalmente, en la sección 4.3 tenemos otra opción de optimización basada en sus características estructurales, de cinemática y de trayectorias.

4.1. Optimización mediante síntesis de mecanismos

En el área de diseño de robots óptimos tenemos la síntesis de mecanismos [**?**].

La elección de la morfología de un mecanismo (tipos de articulación) es un problema complejo pues este problema esta relacionado a la determinación de variables no continuas, no numéricas o no discretas. Todo esto se da porque no existe una métrica en el conjunto de posibles morfologías.

En esta sección estamos interesados en la elección de robots que tienen que alcanzar una trayectoria cerrada entre obstáculos, además de que se asocia este problema con el criterio de la tarea(alcanzar puntos via en una trayectoria) a realizar y determinadas restricciones para evitar colisiones [**?**].

Las variables a optimizar son el tipo de articulación, la dimensión de los eslabones y la localización del robot, teniendo en cuenta que podemos trabajar con todas estas variables a la vez o trabajarlas por separado o en grupos de 2.

Nuestro problema consiste en determinar el mejor mecanismo para seguir una trayectoria determinada en un ambiente con obstáculos. El mecanismo debe ser una cadena cinemática n-dimensional y contamos con infinitas articulaciones para armarlo.

El conjunto de búsqueda esta compuesto por un conjunto de morfologías y en cuanto a la trayectoria, esta debe ser seguida sin discontinuidades.

Con las consideraciones de nuestro problema le asociamos el problema de optimización:

$$max_X f(X) \tag{4.1.1}$$

bajo la restricción de que no hay colisiones con el ambiente y los limites de las articulaciones del mecanismo. Aquí $f(X)$ es la longitud de la trayectoria seguida por el elemento terminal del mecanismo. Por otra parte:

$$X = \begin{bmatrix} \mathrm{T} \\ \mathrm{L} \\ \mathrm{x} \end{bmatrix} \tag{4.1.2}$$

donde T es el tipo de mecanismo del individuo X y pertenece a la base de datos de mecanismos seleccionados a priori, L es el vector que contiene las longitudes de cada

uno de los eslabones (L_i es la i-ésima coordenada y representa la longitud del eslabón i) y x es el vector donde cada x_i es la i-ésima coordenada de la posición y orientación de la base del mecanismo sobre una línea.

Puesto que tenemos las restricciones, hemos de cambiar 4.1.1 en uno nuevo con penalizaciones pero sin restricciones 4.1.3.

$$max_X F(X) \tag{4.1.3}$$

Para este nuevo problema $F(X)$ es la longitud de la trayectoria alcanzada sin colisión por el individuo X.

El problema de optimizar $F(X)$ se resuelve con un algoritmo genético con elitismo [**?**] que a continuación se describe:

1 : Inicializamos la población generándola aleatoriamente. Los elementos de la población pertenecen a la base de datos de mecanismos.

2 : Evaluamos cada uno de los elementos de la población. Esta evaluación puede ser en 4.1.1. Primero probamos si alcanzamos el objetivo y en caso de que así sea probamos si existe colisión con el ambiente para esta posición y orientación específica. El resultado es la longitud total de los segmentos de trayectoria que son alcanzables sin colisión. El alcance esta determinado por la cinemática inversa de cada mecanismo.

3 : Seleccionamos los elementos de la población. La selección se basa en el principio de la ruleta.

4 : Realizamos la cruza entre los diferentes elementos de la población. Esta operación corresponde a permutación de T, L o x entre 2 individuos.

5 : Mutamos los cromosomas de los elementos de la población dada una probabilidad determinada y teniendo en cuenta que la posición de la base tiene que estar sobre una linea.

6 : Repetimos los pasos 2 a 5 hasta que se haya encontrado una solución adecuada o se cumpla cierto número de generaciones.

4.2. Optimización mediante reducción de DOF

La modularidad fue introducida en el campo de diseño de robots manipuladores para dar flexibilidad, economía, mantenimiento y rápido desarrollo. Esto también im-

plica que uno puede construir una configuración de robot para una tarea en específico con pocos módulos y DOFs para mejorar una tarea. Los 2 tipos de modulos que se manejan son rotacional y prismático.

El concepto de reducción de grados de libertad (DOF) es introducido cuando se empezó a trabajar en optimizar la modularidad de robots configurables para tareas especificas. Con pocos módulos y grados de libertad un robot puede tener configuración simple, alta capacidad de carga, bajo consumo de energía con tal de llevar a cabo su tarea mas eficientemente.

El diseño óptimo de robots modulares configurables para una tarea en específico también es un problema difícil [**?**] pues involucra síntesis de mecanismos y síntesis de dimensión. El espacio de diseño es discreto y crece exponencialmente con el número de módulos.

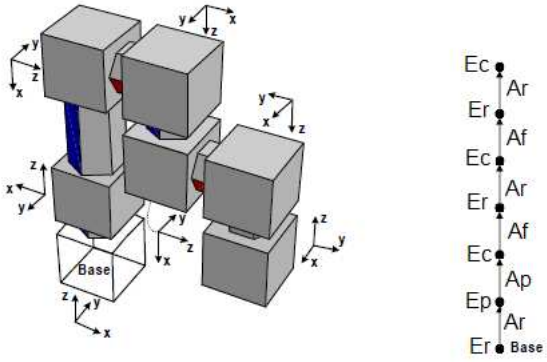
El propósito de esta sección es diseñar un manipulador que es óptimo para realizar una tarea determinada.

En resumidas cuentas, el toolbox de MATLAB que implementamos para este tipo de optimización usa la representación llamada *AIM* (Assembly Incidence Matrix); una suma pesada del número de módulos es la función objetivo y varias tareas relacionadas a medidas cinemáticas son consideradas como restricciones y un algoritmo evolutivo es empleado por la naturaleza discreta del problema para resolverlo [**?**].

Para este tipo de problema un sistema modular reconfigurable es propuesto por [**?**] para propósitos de automatización en fabricación. Basados en construcción por bloques, todos los módulos son diseñados como cubos que pueden ser conectados en diferentes orientaciones. Tres tipos de módulos son considerados: cúbicos, con articulación revoluta y con articulación prismática. En este esquema de configuración cinemática los módulos son considerados como eslabones y las conecciones como articulaciones (fija, revoluta o prismática).

El AIM (Assembly Incidence Matrix) es una representación de ensamblado de un robot modular propuesto por [**?**] y esta basado en la idea de grafos cinemáticos para análisis mecánicos y síntesis(Figura 4.1(a)). Es decir, puesto que un robot manipulador es una colección de eslabones y articulaciones, entonces admite una representación gráfica en la cual los vertices representan eslabones y las aristas representan articulaciones. En si, la matriz de incidencia representativa de un grafo, expresa la topología del robot (Figura 4.1(b)).

Dada una matriz que representa el grafo de nuestro robot (Figura 4.1(b)) de dimensión (eslabones+1)x(articulaciones+1), los 1's se sustituyen con un par ordenado $P = (p_1, p_2)$ llamado *vector de puertos* donde p_1 es la dirección normal en la que se encuentra el elemento con el que esta conectado y p_2 es la dirección de giro. Por usar



(a) Robot y su grafo.

$$\text{RobotP} = \begin{bmatrix} (z,-y) & 0 & 0 & 0 & 0 & 0 & \text{Er} \\ (-z,y) & (z,y) & 0 & 0 & 0 & 0 & \text{Ep} \\ 0 & (x,-y) & (z,y) & 0 & 0 & 0 & \text{Ec} \\ 0 & 0 & (-x,-y) & (z,y) & 0 & 0 & \text{Er} \\ 0 & 0 & 0 & (-x,-y) & (z,y) & 0 & \text{Ec} \\ 0 & 0 & 0 & 0 & (y,-x) & (z,x) & \text{Er} \\ 0 & 0 & 0 & 0 & 0 & (z,x) & \text{Ec} \\ \text{Ar} & \text{Ap} & \text{Af} & \text{Ar} & \text{Af} & \text{Ar} & 0 \end{bmatrix}$$

(b) Matriz de incidencia del robot.

direcciones para definir P, tengamos en cuenta que $p_i \in \{\pm X, \pm Y, \pm Z\}$.

Los tipos de eslabones y articulaciones usados en el ensamblado son mostrados dentro de la matriz en un renglón y una columna adicional respectivamente(Figura 4.1(b)). E_p, E_r y E_c denotan eslabones prismático, revoluto y cúbico respectivamente.

mientras que A_p, A_r y A_f son articulaciones prismática, revoluta y fija.

Definimos la tarea a realizar por nuestro robot configurable como una colección de puntos a ser seguidos por el elemento terminal (puntos via) en el espacio de trabajo. Ante esto, ahora necesitamos definir nuestro problema a optimizar.

El modelo de optimización discreto orientado a tareas consiste de 3 partes: diseño de parámetros, función objetivo y restricciones.

Diseño de parámetros : Según el modelo de Chen, los tipos de articulaciones usados son determinados por los eslabones, de aquí que los tipos de articulaciones no son considerados como parámetros de diseño. Los parámetros independientes que determinan la configuración de un robot modular son: el número de módulos eslabón (N), los tipos de módulo eslabón (E donde $E \in E_c, E_r, E_p$) y la orientación relativa de ensamble (Vector de puertos P) entre módulos consecu-

tivos. Así el conjunto de parámetros de diseño X es $X \rightarrow N, L, P$.

Con tal de poder comparar mecanismos con diferentes cantidades de módulos y aplicar las ventajas de optimización discreta, introducimos el concepto de modulo de eslabón virtual (E_v)y tipo de articulación virtual (A_v). Su función es compensar la representación de tamaño de los distintos mecanismos y poder aplicarle un algoritmo genético al problema de optimización; con esto, estas entidades virtuales existen en la descripción pero no son reales, por ejemplo, en la figura 4.1 los mecanismos tienen distinta cantidad de módulos (3,4 y 5), pero con ayuda de los elementos virtuales tienen la misma dimensión en términos de su cadena cinemática.

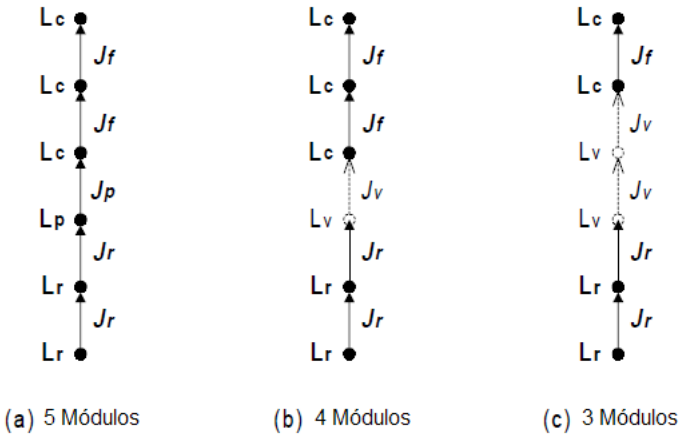


Figura 4.1: Uso de eslabón virtual para comparación de configuraciones.

Con todo lo anterior, todos los parámetros de diseño pueden ser representados en un AIM y para cada posible diseño (N,L,P) existe una matriz única AIM, de aquí que utilizaremos las matrices AIM como parámetro. El tamaño del espacio de búsqueda es igual al número de todos los posibles AIM's.

Función objetivo : Evalúa que tan bueno es una determinada configuración de ensamble. Puesto que usar excesiva cantidad de módulos puede disminuir la capacidad de carga de nuestro mecanismo, nuestro principal preocupación sera usar la mínima cantidad de módulos para construir un robot que cumpla con determinada tarea. De esta manera, nuestra función objetivo F es una suma pesada del número de diferentes tipos de módulos usados en el robot 4.2.4.

$$F = k_p N_p + k_r N_r + k_c N_c \tag{4.2.4}$$

donde N_p, N_r y N_c representan el número de módulos prismáticos, revolutos y cúbicos usados, respectivamente y k_p, k_r y k_c son los respectivos pesos de

N_p, N_r y N_c (Constantes positivas). Puesto que la función a evaluar debe ser minimizada los pesos actuarán como coeficientes de penalización y haremos uso del recíproco de F ($F > 0$), $1/F$ como la función a maximizar en el algoritmo genético.

Restricciones : Las restricciones consideradas son: alcance, rango de trabajo de articulación, y construcción mecánica.

La **restricción de alcance** es usada para verificar que se alcance todos los puntos via en el espacio de trabajo de un manipulador; por lo que si la restricción se cumple la configuración del robot es adecuada para la tarea asignada. Para esta restricción asemos uso de la cinemática inversa, si una solución de cinemática inversa es encontrada para cada uno de los puntos via, la restricción de alcance se satisface.

En cuanto a la **restricción del rango de la articulación**, esta se define como la detección de si los ángulos de articulación están dentro de los límites permitidos cuando se alcanza un punto via. Esto es que para cada articulación θ_i verificamos la fórmula 4.2.5 donde a_i y b_i son los límites permitidos a la articulación i.

$$a_i \geq \theta_i \geq b_i \tag{4.2.5}$$

La **restricción mecánica** verifica la viabilidad de construcción del manipulador, de aquí que tenga las siguientes reglas: (1) un robot modular tiene al menos 2 módulos reales, (2) el sistema de ejes de la base coincide con el mundo real.

En cuanto al algoritmo genético que usaremos, a continuación describimos su funcionamiento.

1 : Inicializamos la población generándola aleatoriamente y consiste de 2 partes, generación del tipo de articulación y generación del vector de puerto en base a una máxima cantidad permitida de módulos (n_m).

2 : El procedimiento de evaluación consiste de 2 partes; detección de funcionamiento con las restricciones y cálculo de la función de aptitud (4.2.5). Si un AIM satisface todas las restricciones su valor de aptitud es calculado, en caso contrario se le asigna un valor infinito.

3 : Seleccionamos los elementos de la población. La selección se basa en el principio de la ruleta.

4 : Realizamos la cruza entre los diferentes elementos de la población. Esta cruza aplica a 2 renglones de 2 AIM's, estas son elegidas aleatoriamente y después intercambiadas.

5 : La mutación actúa sobre las entradas no cero de un AIM y ya que estas están divididas en 2 categorías vectores puerto o tipo de eslabón (última columna) son necesarios 2 tipos de mutación. La mutación de tipo de eslabón actúa solo en la última columna y altera el tipo de eslabón sin afectar los vectores de puerto con probabilidad (p_{mE}). La mutación de vector de puertos altera una de las entradas de tipo vector de puertos con probabilidad (p_{mV}).

6 : Repetimos los pasos 2 a 5 hasta que se haya encontrado una solución adecuada o se cumpla cierto número de generaciones.

4.3. Optimización de robots con tarea específica y restricciones de cinemática y estructura

Los eslabones de robots manipuladores son diseñados para ser capaces de soportar la estructura del manipulador por lo que en muchas ocasiones se requiere actuadores de gran potencia y si aumentamos la cantidad de eslabones y el tamaño de estos requeriremos un alto consumo de energía.

Esta sección está enfocada a mejorar el diseño de robot basados en especificaciones de tarea usando optimización evolutiva. Aquí, la función objetivo minimiza el torque requerido para el movimiento [**?**] y añade restricciones donde las variables de diseño tienen que cumplir ciertas características físicas [**?**].

Viendo que es una característica necesaria a optimizar no podemos dejarla de lado para incluirla en nuestro toolbox con sus respectivas modificaciones.

El problema se define como 4.3.6 donde $f(x)$ es la función objetivo, $g_i(x)$ es el conjunto de restricciones y $x \in \Re^n$ es el vector de variables de diseño (n es el número de variables de diseño).

$$minf(x) \text{ sujeto a } g_i(x) \leq 0 \tag{4.3.6}$$

Para verificar que el manipulador cumple con la tarea especificada se hacen 2 análisis: Análisis de características cinemáticas (posición del elemento terminal) y análisis de características dinámicas (tiempo requerido para completar un movimiento considerando características de inercia). En cuanto a las restricciones, estas tienen rangos en los que deben encontrarse los parámetros del eslabón (longitud del eslabón).

Diseño de parámetros : Dada la configuración de un robot manipulador en cuanto a cantidad de eslabones y parámetros DH, las longitudes de nuestro eslabón son las variables de diseño.

Función objetivo : La función objetivo se define como la suma acumulativa de los torques de cada articulación durante el movimiento del manipulador (4.3.7). En esta ecuación t_{ij} es el torque al tiempo i de la articulación j.

$$f(x) = \sum_{i=1}^{\text{tiempo}} \sum_{j=1}^{\text{articulaciones}} t_{ij}^2 \tag{4.3.7}$$

Restricciones : La **restricción de rango de articulación** es la misma empleada en la sección anterior. En la **restricción de característica estructural** nos encontramos que evaluamos la longitud del eslabón.

El primer paso en el proceso de resolución de nuestro problema es definir el problema, definir las variables de diseño, asignar valores a estos parámetros y definir nuestro vector de restricciones. Para definir el problema a resolver necesitamos la estructura cinemática del manipulador a ser analizado (parámetros DH), la posición inicial y final en el espacio cartesiano, el tiempo de movimiento y la carga a trabajar. Estos valores son usados en una rutina de análisis para obtener valores para las variables de diseño; después se verifica si nuestras variables de diseño violan alguna de las restricciones y si no pasamos a evaluar nuestra función objetivo. Estas evaluaciones son usadas en la rutina de optimización donde nuevos valores para las variables de diseño son generadas.

La rutina de análisis esta formada por:

Cinemática inversa : La cinemática inversa nos ayuda a encontrar las variables de articulación para la posición inicial y final (o incluso puntos via).

Trayectoria en espacio de articulaciones : Aquí se genera la trayectoria $\theta(t)$, velocidad $\dot{\theta}(t)$ y aceleración $\ddot{\theta}(t)$ necesarias para cumplir con una tarea especificada.

Con esto, el algoritmo genético lo aplicamos de la siguiente manera:

1 : Inicializamos la población generándola aleatoriamente. Lo que hay que generar son las longitudes de los eslabones.

2 : El procedimiento de evaluación consiste en la función 4.3.8 donde Z es un valor mas grande que la función objetivo e I es 1 si se viola alguna restricción y 0 en otro caso.

$$F = (Z - f) + (1 - I)Z \tag{4.3.8}$$

3 : Seleccionamos los elementos de la población. La selección se basa en el principio de la ruleta. Se trabaja con elitismo (opcional).

4 : Realizamos la cruza entre los diferentes elementos. Se escogen 2 individuos vectores y un elemento del vector a partir del cual intercambiaran información con una probabilidad p_c.

5 : En términos de mutación, para cada elemento en la población cambiamos sus cromosomas con probabilidad p_m

6 : Repetimos los pasos 2 a 5 hasta que se haya encontrado una solución adecuada o se cumpla cierto número de generaciones.

Capítulo 5

Sistema de Visión

Recordemos que nuestro tema principal es que nuestro robot pueda jugar air-hockey y puesto que ya hemos modelado y optimizado nuestro robot, ahora necesitamos proporcionarle a nuestro sistema de control la información necesaria para que tome decisiones y asigne una tarea determinada al robot. El encargado de todo esto es nuestro sistema de visión. El sistema de visión es el encargado de predecir la trayectoria futura exacta de nuestro puck en el espacio de la mesa de air-hockey.

Como quien dice, en este capítulo estamos interesados en la información visual que puede ser extraída de los cambios espaciales y temporales en una secuencia de imágenes y con esta poder realizar predicciones del movimiento del puck.

Definición 5.0.1 *Una secuencia de imágenes es una serie de N imágenes o frames, adquiridos a instantes de tiempo discreto* $t_k = t_0 + k\triangle t$*, donde* $\triangle t$ *es un intervalo de tiempo fijo y* $k = 0, 1, 2, ..., N - 1$*.*

La secuencia de imágenes es obtenida de una cámara web y procesada en una PC, la información obtenida es enviada al simulador. En la primera sección de este capítulo (5.1) hablaremos del equipo y software utilizado para la captura de imágenes. En la sección 5.2 describiremos el procesamiento aplicado a la secuencia de imágenes captada por la cámara para obtener las características del objeto de estudio (puck) y por último en la sección 5.2.5 nos enfocaremos a la predicción de movimiento. La información recopilada se envía al simulador.

5.1. Captura de video

La captura de la secuencia de imágenes se realiza a través de una *QuickCam Pro for Notebooks* de la marca Logitech (5.1).

Figura 5.1: QuickCam Pro for Notebooks

Se realizaron 3 versiones de software de captura: una para MATLAB que se realizó utilizando el Toolbox de Adquisición de Imágenes 2.1, otra en C++ con base en la clase VFW y finalmente una en java teniendo como base la API Java Media Framework (JMF).

A través de cualquiera de las versiones de software de captura se puede acceder a las opciones de configuración de la cámara, permitiendo grabar una secuencia de imágenes, analizarlas, modificarlas y extraer características en tiempo real.

Toda la información recibida por los programas de captura realizados en C++ o java se puede enviar a MATLAB y los capturados en MATLAB se puede procesar en C++ o java.

En el análisis de información no se diferencia de tiempo de procesamiento significativa entre cada una de las versiones hechos, esto, principalmente a que las operaciones realizadas de extracción de características no son complejas y que el hardware de la cámara nos da la opción de trabajar algunas operaciones, por lo que para términos de facilidad de control de usuario y de la simulación se trabajo en el ambiente de MATLAB.

5.2. Procesamiento de video

Teniendo la posibilidad de capturar la secuencia de video y poder trabajar con ellas en tiempo real, nos enfocaremos al proceso de extraer la información a través de las librerías de captura

Las operaciones realizadas para este análisis con cada frame esta dado por el diagrama 5.2.

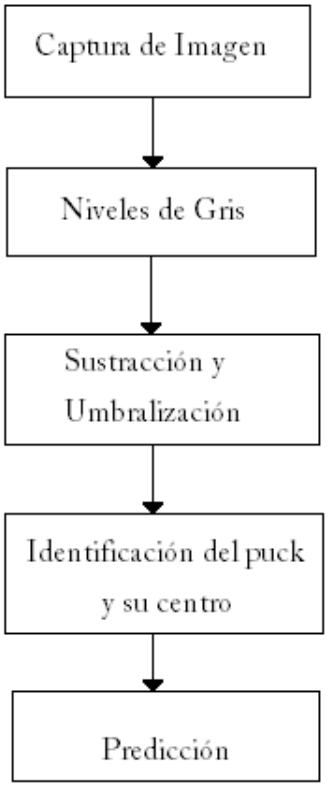


Figura 5.2: Diagrama de procesamiento de imagen

5.2.1. Captura de imagen

La captura de imagen se lleva a cabo de nuestra interface en MATLAB a una resolución de 320 x 240, 30 fps. y una velocidad de obturador de 1/400. Todas estas características asignadas a través del driver de la webcam quien a su vez es llamado por nuestro programa.

5.2.2. Niveles de Gris

Puesto que nuestras imágenes están en color, tenemos 2 opciones para pasar a niveles de gris, via hardware de la webcam o via software. Por simplicidad y manejo de tiempos de procesamiento se trabajo con la primera opción.

5.2.3. Sustracción y Umbralización

Utilizamos la sustracción y umbralización para encontrar el puck, el método se describe a continuación:

Paso 1 : Se captura una imagen base sin el puck I_b (5.3).

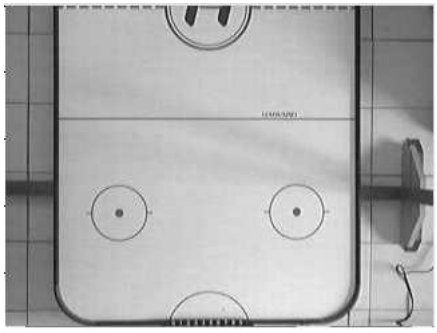


Figura 5.3: Imagen base

Paso 2 : Iniciamos la captura dinámica de imágenes I_i (5.4).

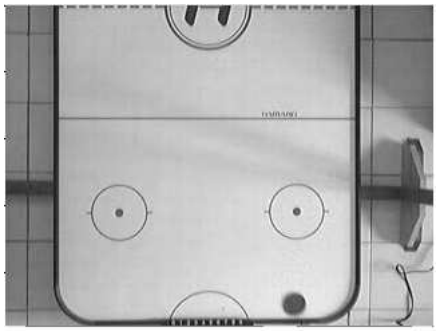


Figura 5.4: Frame i

Paso 3 : A través de funciones virtuales obtenemos la imagen a umbralizar donde solo se encuentra el puck $I_p = I_i - I_b$ (5.5).

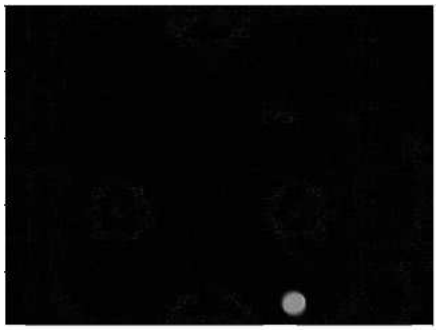


Figura 5.5: Frame i menos Imagen Base

Paso 4 : Umbralizamos I_p con un valor de um $I_{sal} = (I_p > um)$ (5.6).

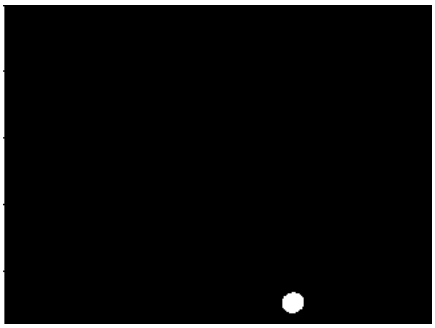


Figura 5.6: Imagen después del umbral

La imagen de salida I_{sal} solo consta de la información aproximada de posición del puck pues dada la velocidad del puck podemos tener cierta dilatación de la imagen.

5.2.4. Identificación del puck y su centro

A pesar de la dilatación presente en I_{sal} nos interesa saber el centro del puck pues con esta distancia y teniendo conocimiento a priori del tamaño del puck podemos estimar su posición sin problemas.

Para resolver el problema de identificación del centro del puck recurrimos a calcular el histograma por líneas y por columnas de la imagen I_{sal} y al calcular los máximos y mínimos sobre este histograma obtenemos la posición aproximada del centro (5.7).

Cabe mencionar que estas operaciones se realizan de manera eficiente al trabajar con programación con archivos MEX quienes permiten a MATLAB trabajar con base

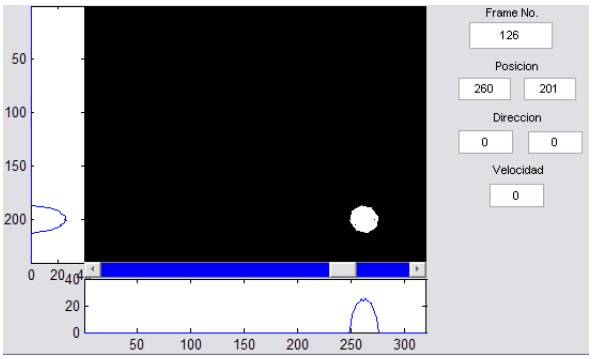
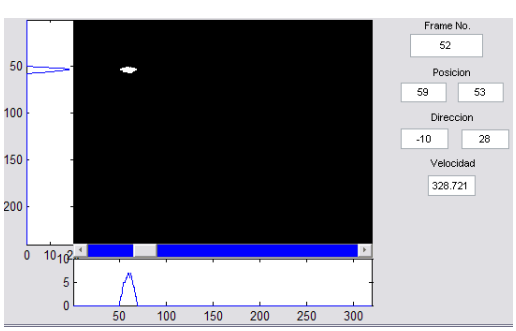


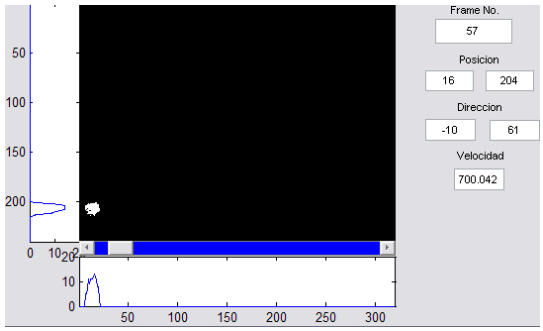
Figura 5.7: Centro del puck recurriendo al histograma

a programas en C++.

En las figuras 5.8 tenemos algunos ejemplos de cálculo de centros.



(a) Centro de puck dilatado



(b) Centro de puck cerca de una esquina.

Figura 5.8: Centros de puck recurriendo al histograma

Teniendo los centro y sabiendo el diámetro del puck tenemos su localización completa.

5.2.5. Predicción de movimiento

Cuando jugamos air-hockey el movimiento del puck por lo general es bastante rápido por lo que debemos construir una función de predicción para el sistema que nos provea de una correcta posición del puck en tiempo y espacio con tal de que el robot pueda pegarle, ya sea para atacar o defender.

Supongamos la figura 5.2.5 el caso con 2 impactos en los lados de la mesa. Dadas 2 imágenes ya procesadas tenemos 2 puntos $P1$ y $P2$ en el espacio (antes del primer impacto) con los que podemos estimar la dirección de movimiento y por ende el punto

de primer choque; si alargamos esta posición hasta la linea de la portería Lp tenemos un punto ficticio $Q1$; reflejando sobre la esquina mas cercana a $Q1$ obtenemos $Q2$. Si $Q2$ se encuentra dentro del área de la mesa ese es el punto de impacto del puck, en caso contrario iteramos este procedimiento hasta que Qn este dentro de la mesa. Para el caso de la figura 5.9, reflejamos $Q2$ con respecto a la orilla mas cercana obteniendo el punto correcto donde pegara el puck Qp.

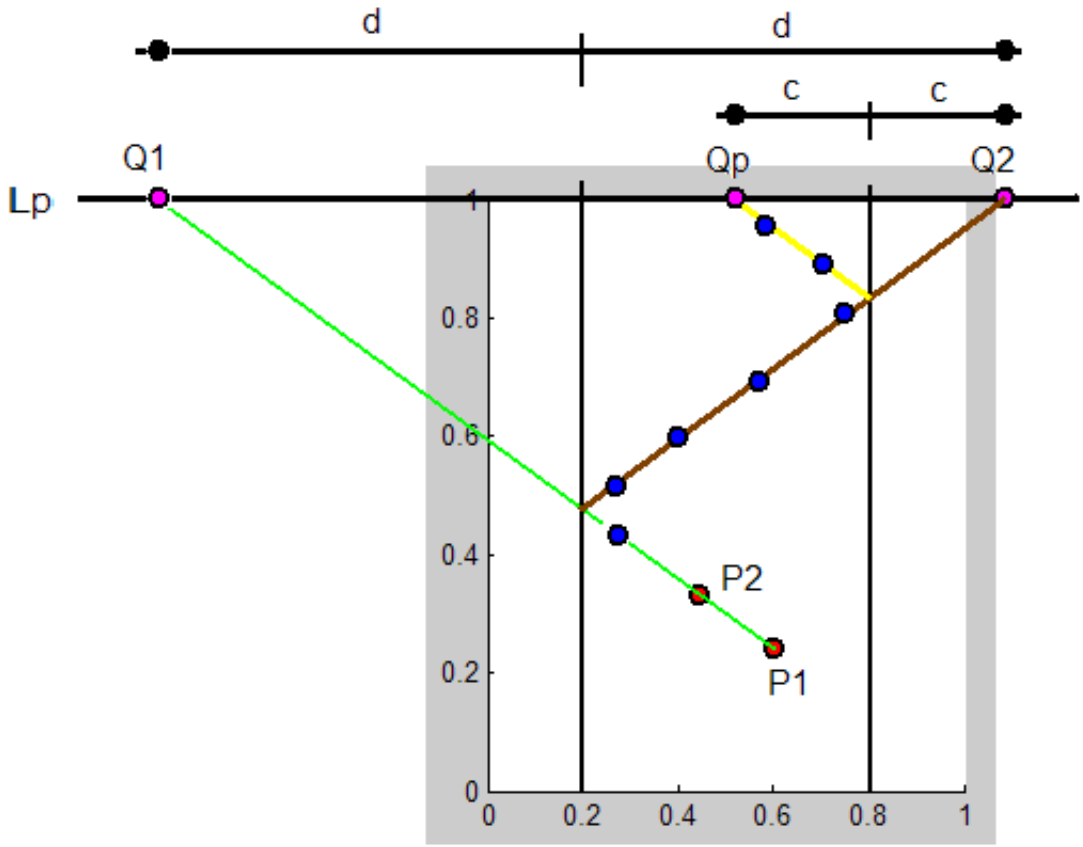


Figura 5.9: Centros extraídos de un puck en movimiento.

En cuanto al tiempo en que el robot debe pegar al puck con precisión, debemos omitir el giro y la fricción entre el puck y la mesa con tal de tomar en cuenta que la velocidad del puck puede variar. Teniendo en cuenta lo anterior la predicción del tiempo se puede dar después del último impacto.

Dada la ecuación de movimiento con aceleración uniforme modificada en el eje y $s = s_y(t) + v_y(t)\triangle t + (a_y(t)(\triangle t)^2)/2$ donde $s_y(t)$ es la posición, $v_y(t)$ es la velocidad y $a_y(t)$ la aceleración en el último dato recopilado. Este algoritmo necesita la información de los puntos en t, $t-1$ y $t-2$ para las velocidades y aceleraciones.

Puesto que consideramos la aceleración suficientemente pequeña la ecuación de distancia en y se simplifica a $s = s_y(t) + v_y(t)t_c$ por lo que el tiempo de impacto es 5.2.1, asumiendo velocidad constante.

$$t_c = \frac{s - s_y(t)}{v_y(t)} \tag{5.2.1}$$

Tanto la posición como la velocidad de impacto calculadas anteriormente se llevaron a cabo tomando la linea l_p alineada con la portería como referencia, si tomamos que esta linea pueda variar en el eje y, tendremos el punto de impacto y la posición a cualquier altura y de la mesa.

Capítulo 6

Control inteligente

Para nuestra aplicación, el robot manipulador requiere realizar una secuencia de movimientos para una tarea determinada, el sistema de control es el encargado de coordinar la posición de las articulaciones, velocidad, aceleración, fuerza y torque.

Existen 4 técnicas de control: De lazo abierto, con retroalimentación, adaptativo y predictivo. Mas allá de los problemas de captura de información del sistema de visión (retardos en la transmisión de información), el problema de air-hockey es difícil de resolver con los métodos tradicionales de control debido a que no la fricción en la mesa de air-hockey es variable, además de que tenemos alta sensibilidad del movimiento del puck ante condiciones iniciales, entradas al sistema de control discontinuas y cambios en la dinámica del sistema [**?**].

Uno debe esperar una alta variabilidad de la fricción en la superficie de la mesa de juego a través del tiempo, por lo que si añadimos que hay irregularidades en la mesa y en las fronteras, nos daremos cuenta de que es una tarea muy difícil obtener un modelo matemático del ambiente en que se juega (uno suficientemente preciso para predecir el movimiento del puck).

Ante esta situación lo mas viable es un método adaptativo cuya representación sea sencilla pero que trabaja en las condiciones tan irregulares que tenemos con la mesa de air hockey; una respuesta son los controles difusos, pero estos necesitan las definiciones de un experto para tomar decisiones además de que no son adaptativos.

Los métodos neurodifusos tienen aprendizaje, por lo que combinado con la facilidad con que podemos modelar la situación a través de conjuntos difusos, representan la mejor opción de control para nuestro problema.

En la sección 6.1 mostraremos la teoría de control difuso y la implementación de este método se lleva a cabo en la sección 6.2. La teoría de un sistema neurodifuso se

expondrá en la sección 6.3 con su respectiva implementación y comparación con el sistema difuso en la sección 6.4.

6.1. Control difuso

Los controles difusos son de los citados y trabajados hoy en día a pesar de que no trabajan igual que los controles convencionales pues para describir un sistema necesitan de un *experto* que pueda tomar decisiones en base a la información recibida, ocupando el lugar de las ecuaciones diferenciales que siempre modelaban el problema. En esta teoría de control difuso el conocimiento del experto es expresado a través de variables lingüísticas relacionadas con conjuntos difusos y teoría difusa [**?**]. El modelo a resolver es construido a partir de reglas difusas para que un proceso de inferencia procese estas reglas, las evalué y de una salida; en si las reglas relacionan una situación con una acción.

A continuación damos las definiciones básicas:

Definición 6.1.1 *Una* ***Variable lingüística*** *es una variable cuyos valores son palabras u oraciones en un lenguaje natural o artificial.*

Definición 6.1.2 *Una* ***Variable difusa*** *esta definida las 4 características X, L, χ, M_x donde X es el nombre simbólico de una variable lingüística (ejemplo: temperatura de un elote), L es el conjunto de etiquetas lingüísticas asociadas con X (ejemplo: frio, tibio, caliente), χ es el dominio sobre el cual L esta definido (ejemplo: $[10^oC, 40^oC]$) y M_x es la función semántica que regresa el significado de una etiqueta lingüística dada.*

Una variable difusa es *completa* si para cada entrada hay una interpretación lingüística.

Definición 6.1.3 *Un* ***Conjunto difuso*** *(fuzzy set) A es un conjunto que puede contener elementos de forma parcial, lo cual podemos entenderlo como que la propiedad $x \in A$ puede ser cierta con un grado de verdad.*

La posibilidad de pertenencia es una probabilidad llamada *grado de pertenencia* de x a A y se expresa como $\mu_A(x)$ la cual puede ser visto como una *función de pertenencia* (6.1.1). Si $\mu_A(x) = 0$ entonces x no pertenece a A, si $\mu_A(x) = 1$ entonces x pertenece a A totalmente y si $0 < \mu_A(x) < 1$, x pertenece a A parcialmente.

$$\mu_A : U \rightarrow [0,1]\} \tag{6.1.1}$$

Cualquier función continua de la forma 6.1.1 puede ser catalogada como una función de pertenencia. Dos de las características mas importantes que deberían tener una función de pertenencia son: que tenga la mayor cantidad de propiedades difusas (unimodal, soporte compacto, etc.) y que debe tener una representación simple con tal de que el sistema pueda ser almacenado y evaluado fácil y eficientemente [**?**]. Dentro de las mas usadas están: triangular, trapezoidal, campana y gaussiana.

Ahora bien, dado un sistema, debemos determinar cual es la función de pertenencia ideal para el problema. Si la relación entrada versus salida es lineal a pedazos, entonces es apropiado trabajar con conjunto lineales a pedazos, pero si es continua y suave es mejor usar funciones de pertenencia cuadrática [**?**]. Recordemos que la función de pertenencia puede variar dependiendo de la forma del concepto que se quiere representar [**?**].

Por lo general la función de pertenencia mas utilizada es la triangular porque su descripción es de lo mas simple sino que además es relativa a la mayoría de los problemas y fácil de analizar. Sin embargo, existen métodos para estimar funciones de pertenencia: método horizontal, vertical, de inferencia, etc.

Regresando a las definiciones, un conjunto difuso A puede ser representado como un conjunto de pares ordenados de un elemento en el universo U y su grado de pertenencia (6.1.2).

$$A = \{(x, \mu_A(x))|x \in U\} \tag{6.1.2}$$

Definición 6.1.4 *El **Soporte** de un conjunto difuso A es el conjunto de los x que pertenecen en cierta medida, a A. Es decir, verifican $\mu_A(x) > 0$.*

$$Soporte(A) = \{x \in U|\mu_A(x) > 0\} \tag{6.1.3}$$

Definición 6.1.5 *El **Ancho** de un conjunto difuso A se usa para definir el rango de soporte y se da en términos de ínfimos(inf) y supremos(sup).*

$$Ancho(A) = sup(Soporte(A)) - inf(Soporte(A)) \tag{6.1.4}$$

Si el soporte es compacto (en términos topológicos), supremo e ínfimo pueden ser reemplazados por máximo y mínimo respectivamente.

Definición 6.1.6 *La **Altura** de un conjunto difuso A en U es el grado de pertenencia mas grande de un elemento en A.*

$$Altura(A) = sup_{x \in U}(\mu_A(x)) \tag{6.1.5}$$

Un conjunto difuso se dice que es *normal* si $Altura(A) = 1$ y *subnormal* si $Altura(A) < 1$. Por otra parte si la altura de A tiene 1 solo elemento se dice que A es unimodal y multimodal en otro caso.

Definición 6.1.7 *Los Puntos de Cruce son los puntos del conjunto difuso para los cuales $\mu_A(x) = 0{,}5$.*

Definición 6.1.8 *El Centro o Núcleo de un conjunto difuso A es el conjunto de todos los puntos para los cuales $\mu_A(x) = 1$.*

Definición 6.1.9 *Una conjunto difuso es **singleton** si su soporte es un solo punto con una función de pertenencia de 1.*

Definición 6.1.10 *Un conjunto difuso se dice que es **Convexo** si y solo si para cualesquiera $x_1, x_2 \in U$ y cualquier $\lambda \in [0, 1]$ se cumple:*

$$\mu_A(\lambda x_1 + (1 - \lambda)x_2) \geq min\mu_A(x_1), \mu_A(x_2) \tag{6.1.6}$$

La estructura de un control difuso la podemos resumir en el diagrama de la figura 6.2. Usando un procedimiento creado por Mamdani [**?**] un controlador difuso se divide en 3 partes principales: fusificación, la aplicación de la base de conocimientos a través de la maquina de inferencias y la defusificación. La normalización y la denormalización es optativa.

Normalización : Esta función mapea los valores físicos (del proceso a ser controlado) en un universo normalizado y por ende realiza un escalado del módulo de fusificación. Esta operación no es estrictamente necesario en un control difuso, pero presenta la ventaja de si se realiza, la fusificación, los procesos de la maquina de inferencia y defusificación pueden ser asignados independientemente del dominio físico. Este mapeo se lleva a cabo multiplicando las entradas del dominio predeterminado x, por un factor de normalización N_x (6.1.7).

$$x_{nor} = N_x \cdot x \tag{6.1.7}$$

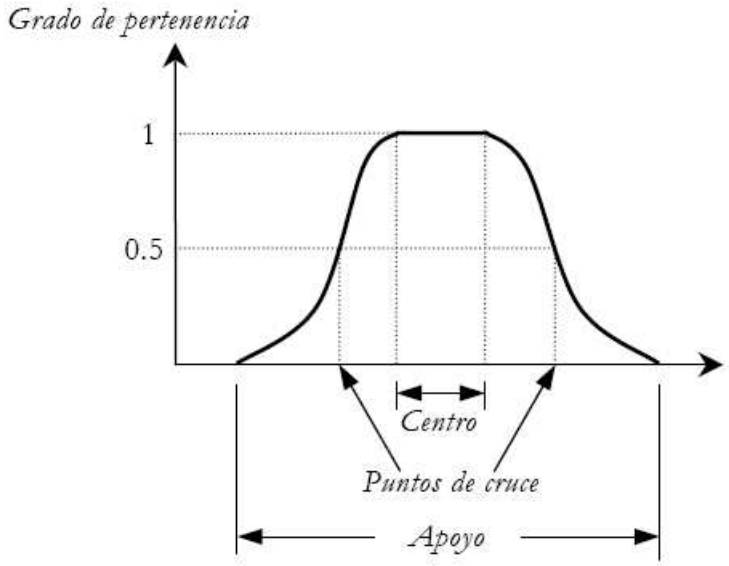


Figura 6.1: Algunas características de funciones de pertenencia

Fusificación : Transforma un término objetivo en un término difuso, o como quien dice, el fusificador (módulo de fusificación convierte) convierte los valores reales de entrada a grados de pertenencia de un conjunto difuso. Estas conversiones se realizan usando las funciones de pertenencia que se encuentra en la base de conocimientos. Hablando propiamente, hay 2 tipos de estrategias de fusificación; (1)fusificación cuando la inferencia esta basda en composiciones y (2) fusificación cuando la inferencia esta basada en disparo de reglas. De aquí que la elección de estrategia esta determinada por el tipo de *máquina de inferencia* empleada en el control difuso.

Base de conocimientos : También llamado evaluador de reglas. Procesa las reglas produciendo una lista de salidas difusas usando los valores de entradas actuales. Tiene 2 componentes, (1)la *base de reglas* de decisión y (2) la *base de datos*.

Base de reglas : Contiene las *reglas de decisión* expresadas como sentencias SI-ENTONCES. Consiste de N reglas de la forma 6.1.8 que dan transparencia al sistema.

$$\text{SI Antecedente/Estado ENTONCES Consecuente/Acción} \tag{6.1.8}$$

Cada regla consiste de 2 partes: La entrada o regla *Antecedente* que define de manera imprecisa el estado del sistema y la salida o *Consecuente* que representa la acción a realizar por el sistema para solucionar la situación del sistema. Tanto como antecedente como consecuente pueden tener mas de un conjunto

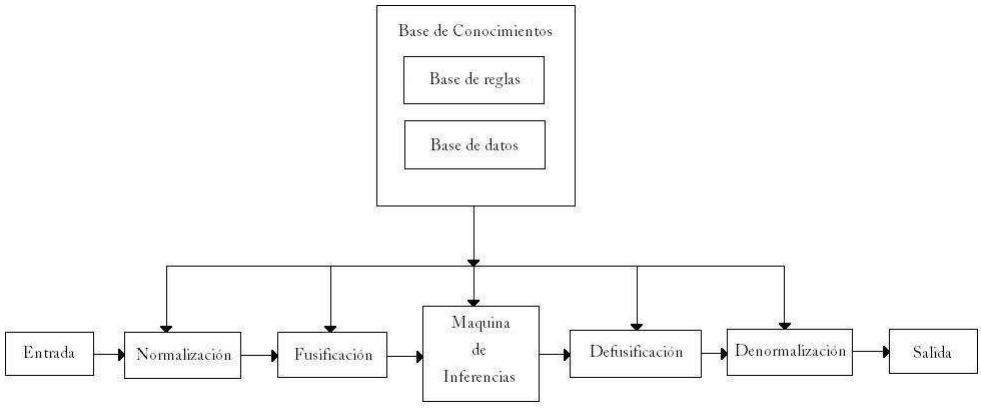


Figura 6.2: Diagrama de un control difuso

difuso si es necesario; producto de combinar estos conjuntos difusos usando los operadores AND y OR (6.1.9). Hay que tomar en cuenta que un peso w_i conocido como *importancia de la regla i* se puede añadir para dar mas énfasis a ciertas reglas que a otras y que el consecuente pude ser fijo o una función del antecedente.

$$\begin{gathered}\text{SI } x_1 \text{ es } S_{11} \text{ AND/OR } \ldots \text{ AND/OR } x_n \text{ es } S_{n1} \text{ ENTONCES } y_1 \text{ es } T_{11} \text{ AND } \ldots \text{ AND } y_q \text{ es } T_{q1} \\ \ldots \\ \text{SI } x_1 \text{ es } S_{1m} \text{ AND/OR } \ldots \text{ AND/OR } x_n \text{ es } S_{nm} \text{ ENTONCES } y_1 \text{ es } T_{1m} \text{ AND } \ldots \text{ AND } y_q \text{ es } T_{qm}\end{gathered} \tag{6.1.9}$$

El número total de reglas o combinación de términos lingüísticos de entrada p (6.1.10) depende del número de entradas n y el número de conjuntos difusos en cada entrada p_i; de esta manera podemos darnos cuenta de que si el número de entradas incrementa, los requerimientos para el problema aumentan exponencialmente (memoria para los datos), por lo que se dice que los sistemas difusos tienen el problema de dimensionalidad.

$$p = \prod_{i=1}^{n} p_i \tag{6.1.10}$$

Para encontrar el conjunto de reglas los métodos disponibles son (1) Experiencia de un experto (Creación de un conjunto de reglas en base en la retroalimentación dada por un usuario experto), (2) Acciones de control del operador (Deducción de reglas difusas SI-ENTONCES de observaciones de las acciones del operador de control), (3) Modelo difuso del proceso (Invirtiendo un modelo difuso del

proceso)y (4)Aprendizaje (Utilizar un sistema capaz de auto-organizarse como un sistema neurodifuso).

Base de datos : Provee toda la información necesaria a los otros módulos para permitir su funcionalidad (conjuntos difusos, funciones de pertenencia, significado de valores linguisticos, dominios físicos, factores de normalización y denormalización).

Máquina de inferencias : Evalua las reglas de control almacenadas en la base de reglas. El resultado de la máquina de inferencia es un conjunto de varios conjuntos difusos de salida y tiene 4 tareas: Disparo de reglas, cálculo de fuerza, implicación difusa y agregación de reglas. Una entrada x **dispara una regla** a un grado $s_i \in [0, 1]$. El **cálculo de la fuerza de disparo** f_x se realiza usando los valores de las funciones de pertenencia usadas en el antecedente y depende de como están combinadas las sentencias en el antecedente (si se usan AND o OR). Así, sustituyendo en cada regla los conectivos lógicos entre antecedentes por uno de sus operadores lógicos equivalente, se podrá operar con los escalares que representa cada antecedente y obtener como resultado del multiantecedente un escalar. La intersección AND de 2 conjuntos difusos se modela mediante una familia de operadores t-normas, siendo el *mínimo* y el *producto* algebraico los tipos de t-normas más utilizados; la unión OR de 2 conjuntos difusos se modela mediante otra familia de operadores t-conormas o s-normas, cuyo representante es el máximo y en cuanto a la negación (NOT) también tiene una familia de operadores que la modelen, siendo la complementariedad aditiva la habitual. Por ejemplo, considerando el antecedente 6.1.11, el método de este cálculo de la fuerza de disparo usando el operador mínimo (min)esta dado por la ecuación 6.1.12.

$$\text{SI } x_1 \text{ es } S_{11} \text{ AND } \ldots \text{ AND } x_n \text{ es } S_{n1} \tag{6.1.11}$$

$$\mu_T = min(\mu_{A_1}, \ldots, \mu_{A_n}) \tag{6.1.12}$$

El conectivo lógico $ENTONCES$ representa la **implicación difusa** entre antecedente y consecuente se calcula por una t-norma cuyos representantes mas utilizados son el *mínimo* y *producto* algebraico ya que preservan la relación causa-efecto y el sentido físico. Si se elige el operador mínimo para la implicación usamos 6.1.13; mientras que si usamos el operador producto recurrimos a 6.1.14.

$$\mu_{T'}(x) = mins, \mu_{T'}(x) \tag{6.1.13}$$

$$\mu_{T'}(x) = s * \mu_{T'}(x) \tag{6.1.14}$$

Una vez evaluadas todas las reglas y obtenidos los conjuntos difusos de salida modificados, hay que realizar la **agregación** de todas las reglas para obtener un resultado único de todas ellas. Esta agregación es una unión lógica traducida por una t-conorma (máximo o suma algebraic), obteniéndose así el conjunto difuso de salida. Si se usa el operador máximo el conjunto difuso de salida esta dado por 6.1.15 y en el caso del operador suma, la agregación se da por 6.1.16.

$$\mu_{salida}(x) = max\{\mu_{T_1'}(x), \mu_{T_2'}(x), \ldots, \mu_{T_m'}(x)\} \tag{6.1.15}$$

$$\mu_{salida}(x) = \mu_{T_1'}(x) + \mu_{T_2'}(x) + \ldots + \mu_{T_m'}(x) \tag{6.1.16}$$

Defusificación : Mapea el valor de una variable linguistica a un grado de pertenencia a conjunto difusos. La entrada al bloque desdifusor es el conjunto difuso de salida, resultado del bloque de inferencia. Hay mas de 30 métodos para realizar la desfusificación pero **Centro de sumas o CoS** (6.1.17 en el caso continuo) y **Centro de gravedad o CoG** (6.1.18 en el caso continuo)son las mas usadas. En el caso de CoG, este método tiene la ventaja de que da suavidad a la salida y transición gradual entre reglas pero como desventaja tiene un alto costo computacional, por lo que tiende a usarse mas el CoS.

$$\bar{u} = \frac{\int x_i \cdot \sum(x_i)dx}{\int \mu_{salida}(x)dx} \tag{6.1.17}$$

$$\bar{u} = \frac{\int x\mu_{salida}(x)dx}{\int \mu_{salida}(x)dx} \tag{6.1.18}$$

Denormalización : Mapea los valores normalizados de la defusificación en valores físicos a través de la fórmula $y_{den} = N_y \cdot y$ donde y es el valor de salida normalizado y N_y es el factor de denormalización.

6.2. Implementación del Control difuso

Tengamos en cuenta que la estrategia de control se aplicara a nuestro robot manipulador para jugar air-hockey, por lo que podemos reducir el sistema a 3 estrategias: defender, atacar o esperar. La selección de entre estas estrategias esta dada por el control difuso, por lo que en si, la decisión difusa, tomará las condiciones para hacer una buena decisión.

Del análisis del juego de air-hockey y la simulación del robot manipulador a ciertas velocidades, tenemos que, para velocidades muy altas o en los casos en que el

puck tiene demasiados impactos durante su trayectoria la predicción es lenta y muy imprecisa, por lo que en este caso la estrategia a tomar es la de **DEFENSA**.

En este mismo análisis podemos obtener que si la velocidad es lenta y el movimiento es simple, entonces la estrategia de juego es **ATAQUE**.

La estrategia de **ESPERA**, la tomamos cuando el puck esta fuera de cierto rango en el área de juego o cuando se mueve a la portería del jugador contrario.

Puesto que de dos puntos iniciales calculados, el último punto de impacto puede ser calculado de acuerdo al algoritmo de predicción descrito en la sección 5.2.5; la decisión del sistema difuso se hará acorde a la distancia d entre el último punto de impacto, a la linea de golpeo Lp y la velocidad v del puck (véase Figura 6.3).

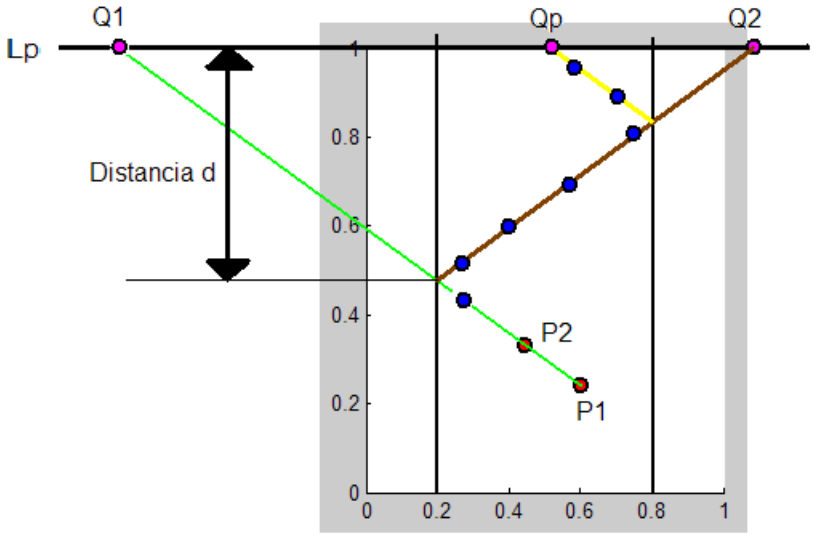


Figura 6.3: Distancia d para uso del control difuso.

Cada variable esta fusificada por 3 funciones de pertenencia triangulares que representan las variables difusas: Grande (G), Media (M) y Pequeña (P). Las funciones de pertenencia de salida y constan de un solo punto (singleton) en 0, 0.25, 0.5, 0.75 y 1. Existen 9 reglas de pertenencia mostradas en la tabla 6.4 y en cuanto a la estrategia de decisión; esta esta determinada por un valor H:

Ataque : Si $y \geq H$

Defensa : Si $y > H$

Espera : Si $y = 0$

	Distancia		
Velocidad	P	M	G
P	0.5	0.25	0
M	0.75	0.5	0.25
G	1	0.75	0.5

Figura 6.4: Reglas base para sistema difuso.

6.3. Control Neurodifuso

Los métodos de control difuso pueden modelar problemas lineales y altamente no-lineales, imitan las decisiones humanas, realizan sus cálculos de manera rápida si se trabaja con procesamiento en paralelo, puede procesar lenguaje natural y manejar información imprecisa; pero una de sus mas grandes desventajas esta en que sus parámetros no pueden ser calculados de manera analítica y por lo tanto no se puede trabajar con aprendizaje para la precisión de las reglas difusas.

Con tal de mantener las ventajas que ofrecen los sistemas difusos pero aplicar algoritmos de aprendizaje utilizamos las llamadas técnicas **neurodifusas**, las cuales unen un sistema difuso con redes neuronales artificiales (ANN).

Definición 6.3.1 *Las **redes neuronales artificiales (ANN)** son sistemas organizados de tal forma que simulan las células del cerebro del ser humano y tienen la habilidad de aprender de las relaciones subyacentes de los datos, aunque la ANN no necesita conocer las relaciones entre los datos.*

El modelo básico de una neurona artificial es un mapeo no-lineal de la neurona de entrada $x_i \in \Re^n, i = 1, ..., n$ a la neurona de salida $y_k \in [0, 1]$ o al rango de amplitud normalizado $y_k \in [-1, 1]$ (Véase figura 6.5).

En una ANN las entradas son combinadas linealmente y transformadas a un sistema no-lineal mediante una función de activación; las funciones de activación mas adoptadas son la de umbral, lineal a pedazos, sigmoidal y con funciones hiperbólicas. Es de mencionar que se mejora el funcionamiento de una red neuronal ajustando el centro y la forma de la función de activación.

El proceso de una neurona k esta dado por $y_k = \varphi(v_k)$, donde y_k es la salida de la neurona, φ es la función de activación y el campo local inducido esta dado por 6.3.19.

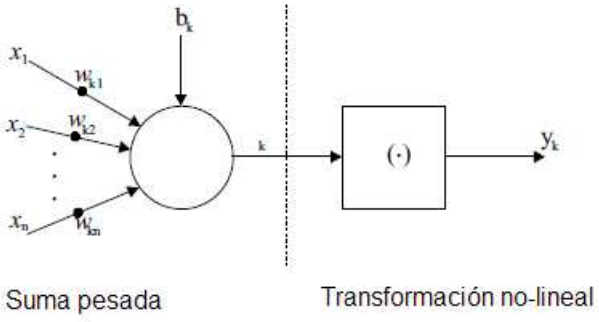


Figura 6.5: Diagrama de una neurona

$$v_k = \sum_{j=0}^{n} w_{kj} x_j \tag{6.3.19}$$

En 6.3.19, n es el número de entradas, x_j son las entradas a la neurona y w_kj son los pesos sinápticos adaptables. En esta misma formula el sesgo se trabaja como $x_0 = 1$ con un pero sináptico $w_{k0} = b_k$.

Las características de una red neuronal ayudan a su clasificación, por lo que podemos clasificarlas según: (1) los tipos de entrada y salida, (2) función de activación, (3) la topología de la red y (4) mecanismo de aprendizaje .

Los tipos de entrada pueden ser continuos o discretos, las funciones de activación pueden ser umbral, lineal a pedazos, sigmoidal, con funciones hiperbólicas, etc.

Las habilidades de una ANN se da a través de la interacción de estas simples unidades de procesamiento llamadas neuronas, las cuales están organizadas y estructuradas en topologías. Estas topologías se pueden dividir en recurrentes (tienen al menos un ciclo de retroalimentación), monocapa (la información va de la capa de entrada a la de salida) y multicapa (posee capas ocultas que interactuan entre las capas de entrada y de salida de la red), para mayor referencia véase la figura 6.6.

Los mecanismos de aprendizaje o entrenamiento son: supervisado, no supervisado y aprendizaje por refuerzo.

Una vez que la ANN ha sido entrenada es capaz de generalizar para producir una respuesta razonable de salida a una situación no presente durante su entrenamiento.

En un sistema neurodifuso el sistema difuso es definido como una estructura de red neuronal artificial pero que mantiene sus componentes fundamentales. Aquí el sistema aprende de su ambiente a través del proceso de ajustar pesos y sesgos, un

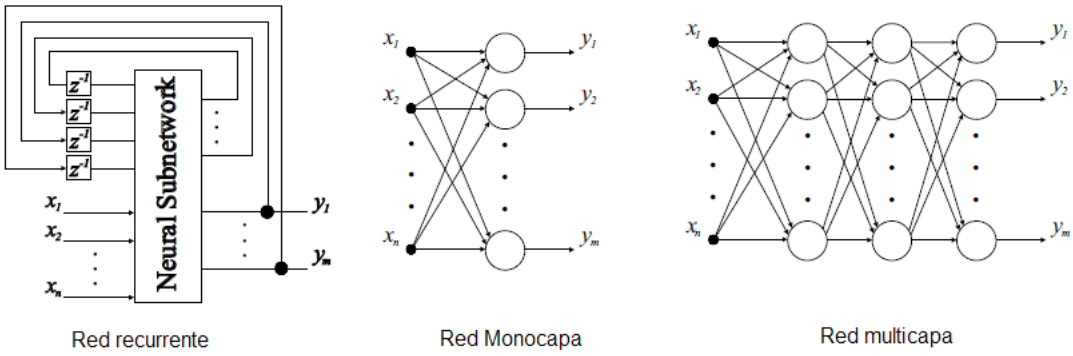


Figura 6.6: Clasificación de ANN por topología

proceso en el que los parámetros libres de una red neuronal artificial se adaptan a través de un proceso de simulación del ambiente en que la red se encuentra. El sistema neurodifuso se representa como una red multicapa pero donde los pesos sinápticos, la propagación y las funciones de activación difiere de las ANN (véase figura 6.7).

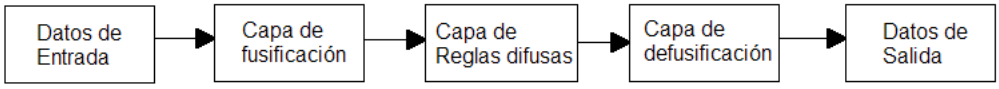


Figura 6.7: Diagrama de un sistema neurodifuso

Como se ve en la figura 6.7, en el sistema neurodifuso las capas de entrada, oculta y salida corresponden a la fusificación, aplicación de las reglas difusas y defusificación. El mapeo entre las 2 primeras es no-lineal y entre las 2 últimas es lineal [**?**].

Un diagrama mas detallado del sistema neurodifuso esta dado por la imagen 6.8 [**?**]. Cada circulo representa una neurona que conceptualmente son elementos orientados lógicamente que realizan operaciones de lógica difusa, modificación lingüística y operaciones de inferencia en vez de las funciones de activación. Cada neurona en la capa de fusificación representa una función de pertenencia del antecedente de una regla difusa, el peso del antecedente y del consecuente requiere tantos parámetros como parámetros modificables tienen las funciones de pertenencia. La salida de la capa de reglas difusas es la fuerza de disparo de las reglas mientras que la cuarta capa, la de desfusificación calcula la desfusificación de las reglas disparadas y por último la quinta capa, la de salida combina las salidas de la capa de desfusificación utilizando el promedio de los datos.

A pesar de que existen otros métodos de implementación de los sistemas neurodifusos como las ANFIS (Adaptive Network Based Fuzzy Inference System) en esta

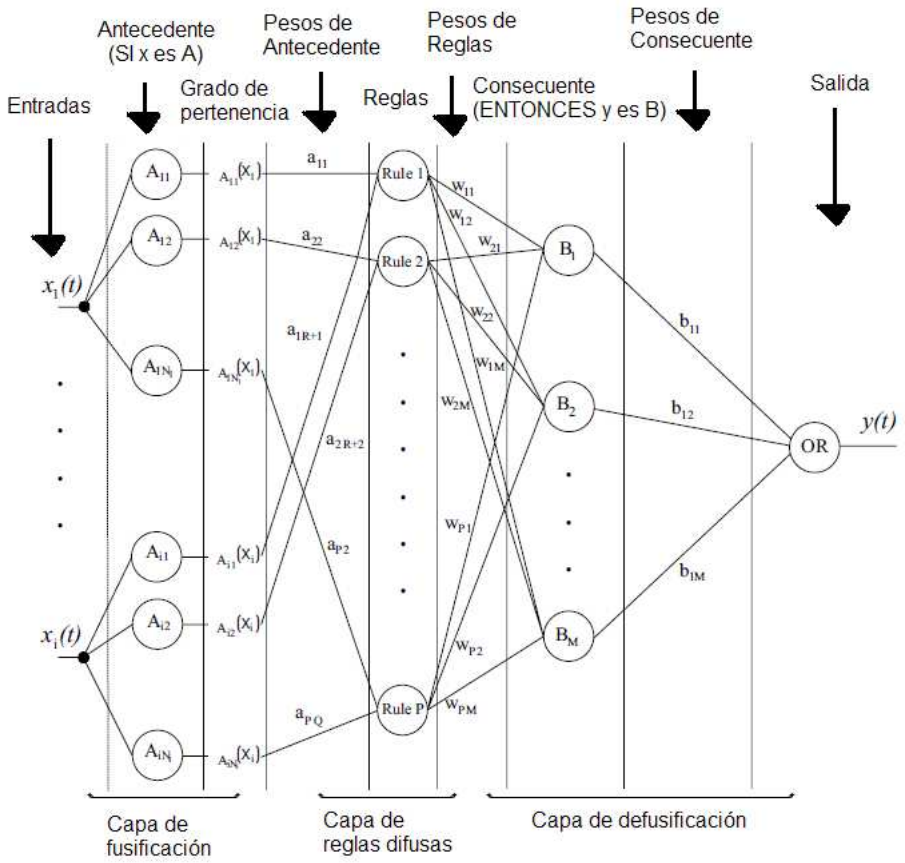


Figura 6.8: Arquitectura de un sistema neurodifuso

tesis se trabajará con el ya descrito por reportar mejores resultados [**?**] y ser adaptable a nuestro sistema de decisión, pues en este esquema tanto el antecedente, como el consecuente y la capa de reglas pueden ser actualizadas además de que hay 2 tipos de aprendizaje que funcionan con gran rapidez y eficiencia: (1) ajustar los pesos de las reglas y (2) ajustar la forma de las funciones de pertenencia cuando son paramétricas o combinación de estas.

Combinando las técnicas de aprendizaje de una **red de memoria asociativa**(AMN) con la transparencia de un sistema difuso tenemos el sistema neurodifuso. El uso de redes de memoria asociativa permite el desarrollo de aprendizaje de reglas y su estructura establece las condiciones bajo las cuales convergencia y estabilidad pueden ser demostradas [**?**]; con todo esto, el modelador de sistema puede entender fácilmente el comportamiento del sistema y modificar fácilmente la *base de conocimientos* y como esta es un conjunto de reglas lingüísticas, el modelador y/o operador puede ajustar manualmente estas reglas.

Para nuestro sistema neurodifuso en términos de aprendizaje es difícil trabajar con el sistema supervisado, pues cada caso de prueba seria un video de la situación, lo cual es tedioso y complicado; además de que con cada cambio en el ambiente de trabajo, un nuevo conjunto de datos de entrada/salida debe ser generado para el sistema. Para acabarla de amolar este sistema no puede ser desarrollado en tiempo real, por lo que el sistema pierde la oportunidad de auto-calibrarse y auto-organizarse para adaptarse a cambios en el ambiente de trabajo. Ante esta situación el tipo de aprendizaje que trabajaremos es el de refuerzo, también llamado *RL* (Reinforcement Learning).

Dentro de los métodos de aprendizaje por refuerzo trabajamos con el entrenamiento de **Actor-Crítico**. Tiene la ventaja de aprender en tiempo real además de un costo computacional mínimo para la elección de la acción a realizar y la habilidad de aprender una política de elección de acción explícitamente estocástica [**?**].

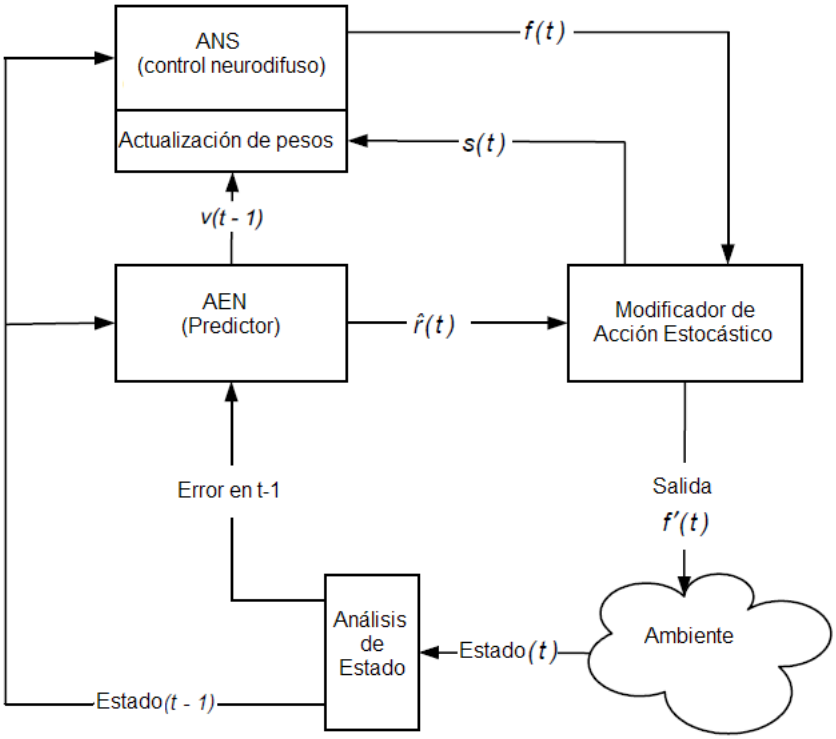


Figura 6.9: Diagrama de funcionamiento GARIC

La arquitectura de trabajo esta basado en la *GARIC* (Generalised Approximate Reasoning based Intelligent Control) la cual consiste de una red neuronal artificial y un control difuso [**?**]. Los bloques importantes de manejo de información en esta técnica son la **Red de Selección de Acción o ASN** (Action Selection Network), la cual como su mismo nombre lo dice, elige la acción a realizar y la **Red de Evaluación**

de Acción o AEN(Action Selection Network) encargado de criticar a acción tomada por el ASN. El diagrama 6.9 muestra como trabaja esta técnica.

A continuación definiremos cada una de las partes del diagrama.

AEN : Mapea un vector de estados y una señal de error en un marcador escalar que indica la calidad del estado(υ); esto también es usado para producir un refuerzo interno $\hat{r}$. Esta es una red de 2 capaz con funciones sigmoidales en todos lados. Su estructura incluye h capas ocultas y n entradas junto con sus respectivos sesgos. En cada capa oculta recibe $n+1$ entradas y tiene $n+h+1$ pesos. La salida de las capas ocultas esta dada por la ecuación 6.3.20 donde g esta dado por la ecuación 6.3.21, t y $t+1$ son pasos de tiempo sucesivos.

$$y_i[t, t+1] = g\left(\sum_{j=1}^{n} a_{ij}[t]x_j[t+1]\right) \tag{6.3.20}$$

$$g(s) = \frac{1}{1+e^{-s}} \tag{6.3.21}$$

En cuanto a la salida de la red de evaluación(υ), esta recibe como entrada la salida de las capas ocultas y_i y los datos de entrada x_i. El valor υ es la predicción de refuerzo y esta definido por la ecuación 6.3.22.

$$\upsilon[t, t+1] = \sum_{i=1}^{n} b_i[t]x_i[t+1] + \sum c_i[t]y_i[t, t+1] \tag{6.3.22}$$

Como hemos comentado, esta red evalúa la acción recomendada por la ASN como una función de la señal de error y el cambio en la evaluación de estado basado en el estado del sistema en el tiempo t+1 ($\hat{r}$). La definición de $\hat{r}$ esta dada en la ecuación 6.3.23, donde $0 \leq \gamma \leq 1$.

$$\hat{r}[t+1] = \begin{cases} 0 & \text{Estado de inicio} \\ r[t+1] - \upsilon(t,t) & \text{Estado de error} \\ r[t+1] + \gamma\upsilon[t, t+1] - \upsilon(t,t) & \text{otro caso} \end{cases} \tag{6.3.23}$$

El diagrama de la red AEN esta dado por la figura 6.10.

ASN En la red de selección de acción, dado un estado actual, esta red selecciona una acción a través de la implementación de un esquema de inferencia basado en reglas de control neurodifuso. Esta red puede ser representado como una red con 5 capas, donde cada una realiza un paso del proceso difuso(Véase Figura 6.8). Es de mencionar que sus parámetros son actualizados de acuerdo a la señal recibida de la AEN.

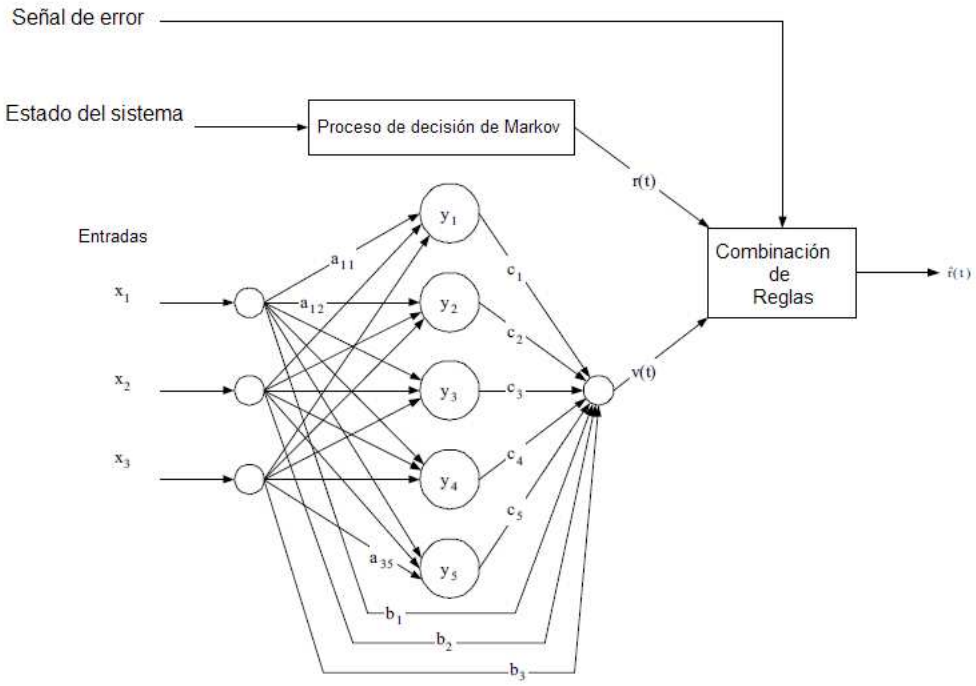


Figura 6.10: Diagrama de funcionamiento de la red AEN.

Modificador de acción estocástico : Da una desviación estocástica a la salida de la ASN con tal de que el sistema tenga una mejor exploración del espacio de estados y una mejor habilidad de generalización [**?**]. La cantidad numérica esta dada por la ecuación 6.3.24 donde F es la acción recomendada por el sistema ASN, F' es una variable aleatoria gaussiana con media F y desviación estandard $\sigma(\hat{r}(t-1))$. Esta σ es una función monótona decreciente no negativa. La magnitud de la desviación $|F' - F|$ es grande cuando $\hat{r}$ es bajo y pequeña cuando el reuerzo interno es alto.

$$s(t) = \frac{F'(t) - F(t)}{\sigma(\hat{r}(t-1))} \tag{6.3.24}$$

6.4. Implementación del Control Neurodifuso

De manera análoga al sistema difuso, el sistema neurodifuso tiene que seleccionar una de las 3 estrategias establecidas: **DEFENSA**, **ATAQUE** o **ESPERA**, por lo que la salida cuenta del sistema es a través de 3 funciones de pertenencia implementadas por medio de funciones de un solo valor(singleton).

Las entradas al sistema neurodifuso también tiene 2 entradas, posición y velocidad del disco y se mantienen las mismas funciones de pertenencia utilizadas en el sistema difuso.

Para términos de implementación solo aplicamos aprendizaje al vector de pesos de las reglas pues este se puede convertir en niveles de confianza para cada regla, reduciendo así, memoria y tiempo de aprendizaje.

El diagrama del sistema neurodifuso implementado esta en la figura 6.11.

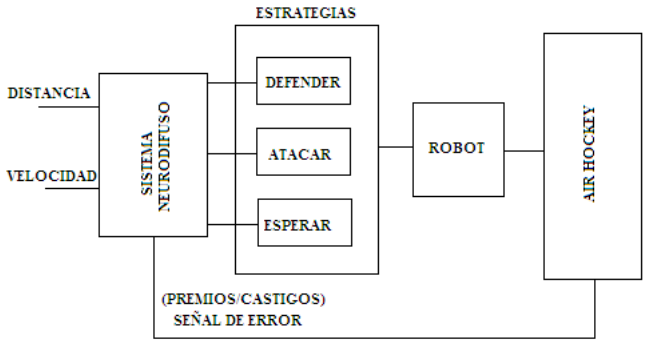


Figura 6.11: Diagrama del sistema neurodifuso para jugar air-hockey

Capítulo 7

Simulador

Crear un simulador de air-hockey es entretenido, pero es una gran aportación al campo de la robótica, sobretodo si tomamos en cuenta que para su sistema de control se trabaja con sistemas neurodifusos, tema nuevo en este tipo de actividades.

Ya en 1989 se hacia investigación acerca de robots jugadores de tennis [**?**], investigación que podemos mencionar como el inicio de la investigación de robots de competencia. A través de los años, la cantidad de robots para competencias se incremento pues después de probar sus sistemas en estas mismas competencias donde se buscaba velocidad de reacción, efectividad en sistemas de detección y procesamiento, estos mismos sistemas se hacían a escalas mayores para servicio de la industria.

El problema básico de jugar air-hockey es controlar un puck circular en una superficie acotada, rectangular y que tiene muy poca fricción con una paleta en un área restringida de la superficie de juego.

Para esta situación, se diseño el hardware especificado en la sección 7.1 con tal de que la información captada por la webcam sea procesada y sirva de entrada a la simulación detallada en la sección 7.2. Las conclusiones para los objetivos planteados en esta tesis son dados a conocer en la última sección de este capítulo.

7.1. Estructura del hardware

El hardware del sistema se muestra en la figura 7.1; este sistema incluye una mesa de air hockey, un bastidor donde se coloca la cámara y una PC.

La mesa de air-hockey tiene las siguientes características: 68 pulgadas de largo

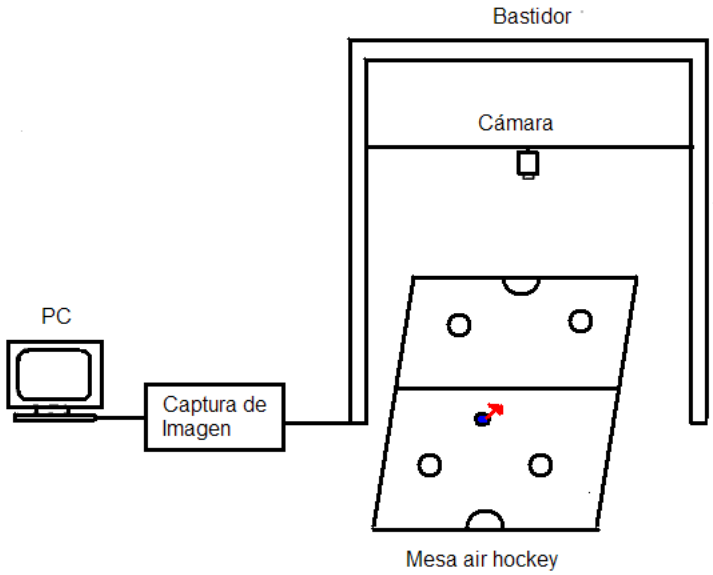


Figura 7.1: Estructura del hardware utilizado.

(172.72 cm.) por 32.75 pulgadas de ancho(83.185 cm.), 67 pulgadas de largo(170.18 cm.) por 31.75 in (80.645 cm.) de superficie de juego y motor generador de aire de 110 V.

En la imagen 7.1, la cámara envía la información a la PC y en esta se realiza la simulación de los robots manipuladores ante el comportamiento de un golpe del puck.

7.2. Implementación del simulador

La figura 7.2 muestra el diagrama de información y estrategia de selección.

Utilizaremos 2 robots en la simulación. Uno de 1 grado de libertad, el cual funcionará como robot portero pues su función sera de solo defender con una heurística y otro desarrollado a través de nuestro proceso de optimización de manipuladores que tendrá el control neurodifuso.

La heurística para nuestro robot portero de un solo grado de libertad se baso en la idea de tratar de estar en la misma posición x con un intervalo de desfase de a lo mas $\pm 3cm$. (por el retardo producido en la detección), este resultado se dio dado que comparamos con las velocidades presentadas por un mecanismo real de la misma configuración y su tiempo de respuesta. La velocidad máxima es la otorgada por este mecanismo (¡Gracias profesor Marrufo!).

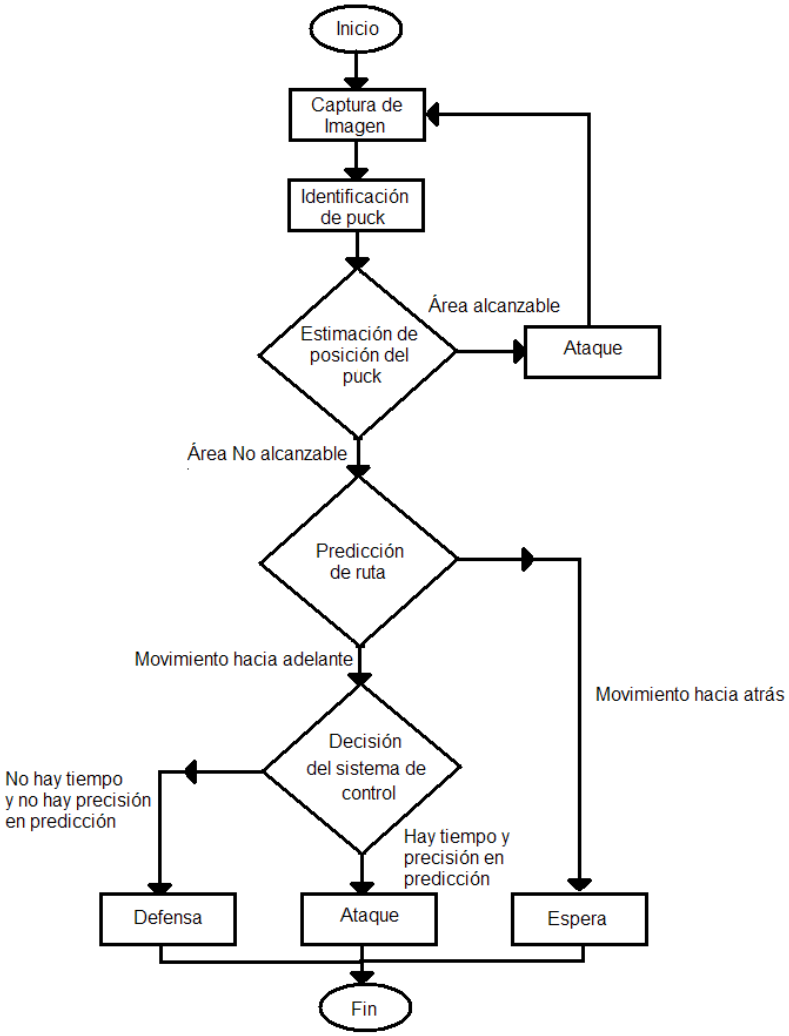


Figura 7.2: Diagrama de flujo de simulación.

Dadas las características de la mesa y el objetivo de jugar air-hockey en la mayor cantidad de área dentro de los primeros 40 cm. de la cancha de air-hockey; se propuso trabajar con robots manipuladores de a lo mas 3 grados de libertad y tratando de obtener la mayor cantidad de área de juego sobre la mesa para técnicas de defensa y ataque. Con nuestro proceso de optimización en términos de síntesis de mecanismos obtuvimos como resultado 2 robots posibles, un robot planar de 2 grados de libertad (mecanismo RR) y uno redundante de 3 grados de libertad rotacionales (mecanismo RRR). El primero resulta el intuitivo para nuestra tarea, pero el segundo mecanismo presenta fluidez en los movimientos del mecanismo y a la posibilidad de tomar la ventaja de inercia al momento del ataque o como quien dice, mayor cantidad de posiciones de espera, defensa y ataque. Cabe mencionar que las posiciones de los robots que presentaron mayor eficacia fue en el centro de la portería y en las esquinas;

siendo la mas beneficiosa en el centro.

Al momento de optimizar en cantidad de grados de libertad solo queda seleccionado el mecanismo RR, pero perdemos capacidad de maniobra y al introducir las restricciones de alcance y rango de articulación también obtenemos el mecanismo RRR. Por último, al trabajar en la optimización con restriciones de cinemática y estructura con los mecanismos RR y RRR, las dimensiones de los eslabones dependen de la posición de la base del robot (resultado del primer proceso de optimización).

Para fines prácticos se desarrollo ambos mecanismos para la simulación con su base en el centro de la portería. En cuanto al funcionamiento de los 2 robots manipuladores hay una diferencia sustancial; puesto que solo podemos golpear el puck con la paleta en el extremo del manipulador, el mecanismo RR solo puede atacar y defender en un rango menor al mecanismo RRR, por lo que esto influyó en el sistema de control, pues mientras el primero en cuestión debe atacar a una distancia mayor de la portería, el segundo puede trabajar la defensa y el ataque a distancias menores.

En la mesa de air-hockey se probó para distintas velocidades y direcciones de disparo (distintos videos) 7.3. Los videos se dieron de entrada siempre cambiando el orden entre los ejemplos 1,2 y 3.

Nuestro sistema de detección de posición, dirección y velocidad del puck es eficiente para los tiempos de respuesta de nuestro manipulador en el puck circular.

Podemos notar que el sistema neurodifuso va reduciendo su tiempo de respuesta debido al aprendizaje además de que esto se traduce en cambios de acción (toma de decisión de ATAQUE por el sistema neurodifuso en la generación 15).

7.3. Conclusiones

Si bien es cierto que ya hay herramientas en MATLAB para simulación de robots manipuladores (la mas sobresaliente la de Corke [**?**]), la realizada para términos de esta tesis presenta grandes ventajas sobre la primera pues además de incluir la información de la simulación del agarre de un gripper de 2 dedos; se implementa una técnica de control difusa y una difusa de control jerárquico para mejora de agarre, también se presentan innovaciones en la generación de rutas y trayectorias, cinemática inversa, así como implementación de técnicas de optimización para la fase de diseño, basados en algoritmos genéticos, técnica que aunque sencilla nos da una pauta para realizar las diversas actividades de un robot manipulador. Estas herramientas nos permitieron hacer un estudio de caso para manipuladores planares con tal de jugar air-hockey, mostrando las características, ventajas y desventajas de cada modelo.

Generación	Ejemplo	Sistema de Control	Tiempo de decisión(s)	Acción
1	1	Difuso	0.31423	Defensa
5	1	Difuso	0.32013	Defensa
10	1	Difuso	0.30931	Defensa
15	1	Difuso	0.31768	Defensa
1	2	Difuso	0.34312	Defensa
5	2	Difuso	0.34127	Defensa
10	2	Difuso	0.33921	Defensa
15	2	Difuso	0.35027	Defensa
1	3	Difuso	0.30234	Defensa
5	3	Difuso	0.30945	Defensa
10	3	Difuso	0.31007	Defensa
15	3	Difuso	0.29087	Defensa
1	1	Neurodifuso	0.41176	Defensa
5	1	Neurodifuso	0.41026	Defensa
10	1	Neurodifuso	0.39617	Defensa
15	1	Neurodifuso	0.38625	Defensa
1	2	Neurodifuso	0.45324	Defensa
5	2	Neurodifuso	0.44355	Defensa
10	2	Neurodifuso	0.43178	Defensa
15	2	Neurodifuso	0.43068	Defensa
1	3	Neurodifuso	0.42381	Defensa
5	3	Neurodifuso	0.42801	Defensa
10	3	Neurodifuso	0.40189	Defensa
15	3	Neurodifuso	0.39723	Ataque

Figura 7.3: Tabla de tiempos y acción de sistemas.

El procedimiento para la detección de movimiento y velocidad en la mesa de air-hockey es una heurística que nos arroja buenos resultados de tiempo de procesamiento pero la efectividad tiene cierta cantidad de error, esto debido al puck usado y a que el motor de la mesa no disipa uniformemente el aire, cosa fundamental para el modelo de detección (estabas considerando una mesa profesional).

El sistema de control neurodifuso para los brazos manipuladores, a pesar de que la fase de entrenamiento es algo engorroso, nos arroja un incremento en la efectividad de la defensa y el ataque sobre lo presentado del sistema difuso, además de que el tiempo en la fase de espera es mas amplio permitiendo menor desgaste del manipulador y eficiente uso del consumo de energía. A pesar de esto en la defensa se tiene un rendimiento inferior al mostrado por el robot portero, pero nuestro robot manipulador tiene un menor consumo de energía, debido a la estrategia de espera. Estos resultados se deben al sistema de aprendizaje por refuerzo con estructura GARIC puesto que se adapta a las características del ambiente.

Por curiosidad verificamos el comportamiento del sistema para diferentes pucks al

circular (triangular y octagonal) a lo cual falló el sistema de predicción de movimiento, por lo que nuestro sistema no es efectivo al no tomar en cuenta error de posicionamiento.

Capítulo 8

Trabajo a futuro

Dentro del área de cinemática inversa aun queda mucho que trabajar, puesto cada vez se requiere mayor rapidez y eficiencia en calcular los valores de las variables de articulación para estar en una posición y orientación determinadas. Existen métodos interesantes que tendrían una repercusión importante si los incluimos en nuestro toolbox de MATLAB para diseñar robots manipuladores, ejemplo de ellos son diseñar funciones de potencial escalar que tengamos que optimizar, métodos basados en optimizaciones de características físicas así como métodos analíticos.

Si nos fijamos en la sección de generación de trayectorias podremos darnos cuenta que apenas se implemento algunos de los métodos de resolución para este problema, por lo que es importante implementar otros métodos para tener un toolbox de robótica mas completo; además de que una de las áreas mas interesantes esta en diseñar trayectorias sin colisiones.

Dentro del área de grippers como trabajo a futuro tenemos el de implementar la diversidad de elementos terminales para nuestro toolbox con tal de analizar su comportamiento con el control parsimónico y el jerárquico (elementos de 2 dedos o mas).

Enfocándonos a la sección de optimización hay mucho mas que hacer, puesto que podemos aplicar nuevos algoritmos evolutivos a las características que estamos optimizando, como por ejemplo, aplicar evolución diferencial, pues podemos trabajar con variables reales. En cuanto al tema de optimización de robots con tareas específicas y restricciones cinemáticas, dinámicas y estructurales, podemos añadir que nuestras restricciones sean mas avanzadas como por ejemplo la cantidad de deflección, característica que puede ser evaluada a través de técnicas de elemento finito. Así también podemos trabajar con características de dinámica.

Dentro de los sistemas de control, pudimos trabajar con un sistema neurodifuso y compararlo con otro difuso, por lo que nos queda de tarea verificar el comportamiento con un modelo neuro-difuso y su comparación en mayor diversidad de aspectos con sistemas de control basado en ecuaciones diferenciales, PID, etc. Además de que uno de los tópicos mas interesantes en la actualidad es el análisis de estabilidad para estos sistemas.

Como lo comentamos en las conclusiones, se trabajo con diferentes pucks y la respuesta del sistema de predicción fue mala, por lo que una linea de investigación es un sistema neurodifuso pero cuya salida no solo sea la acción del sistema sino también la posición a a que se enviará el robot.

En términos del simulador del juego de air-hockey se trabajarán mejoras de despliegue de información así como su comparación con con sistemas implementados en hardware.

Otro punto importante a desarrollarse es implementar todas estas herramientas de trabajo desarrolladas en MATLAB en otros lenguajes que nos podrían aportar mayores beneficios en tiempo de computo como java y C++.

Con todo esto, solo podemos decir, que nos queda trabajo para mucho rato.