

5
25J



UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
ARAGON

APLICACION DE LAS REDES NEURONAL AL
CONTROL DE ROBOTS BASADO EN VISION

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

PRESENTAN:

VERONICA BAHEMA VILLA
HECTOR MENDOZA SORIANO

821692

MEXICO, D.F.

1999



TESIS CON
FALLA DE ORIGEN





Universidad Nacional
Autónoma de México



UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.



UNIVERSIDAD NACIONAL
AUTÓNOMA DE
MÉXICO

**ESCUELA NACIONAL DE
ESTUDIOS PROFESIONALES
ARAGÓN**

**JEFATURA DE INGENIERÍA
EN COMPUTACIÓN**

OFICIO ENAR/JACO/432/98


ASUNTO: Reconocimiento.

DR. JUAN MANUEL IBARRA ZANNATHA
Grupo Robótica y Visión
Centro de Investigación y Estudios Avanzados
del Instituto Politécnico Nacional.
P r e s e n t e .

La Universidad Nacional Autónoma de México, a través de la jefatura de Ingeniería en Computación, de la Escuela Nacional de Estudios Profesionales Plantel Aragón, reconoce su valiosa participación en la elaboración de la tesis: **"APLICACIÓN DE LAS REDES NEURONALES AL CONTROL DE ROBOTS BASADA EN VISION"**, que presentaron los alumnos Verónica Bahena Villa y Héctor Mendoza Soriano, para obtener el título de **Ingeniero en Computación** y lo exhorta a seguir colaborando en este tipo de trabajos que impulsan la investigación y contribuye a la formación de profesionistas.

Sin otro particular, aprovecho la ocasión para enviarle un cordial saludo.

A T E N T A M E N T E
"POR MI RAZA HABLARA EL ESPÍRITU"
San Juan de Aragón, Edo. de Méx., Noviembre 17 de 1998.
EL JEFE DE CARRERA


ING. JUAN GASTALDI PÉREZ

JGP/gga.

A mis padres por todo el amor y apoyo que siempre me han brindado.

A mis hermanos que siempre me respaldan.

Al Dr. Juan Manuel Ibarra por su valiosa participación.

*A Héctor por haber estado a mi lado de manera incondicional ayudándome
en todo momento.*

V.B.V.

A todos los seres queridos que siempre me acompañan.

H.M.S.

Aplicación de las Redes Neuronales al Control de Robots basado en Visión

V. Bahena y H. Mendoza.

16 de noviembre de 1998

Índice General

1	Introducción	5
1.1	Tareas de ensamble robotizado	5
1.2	Objetivo	5
1.3	Por qué usar Visión Artificial	6
1.4	Por qué usar Redes Neuronales	6
1.5	Antecedentes	6
1.6	Descripción de la Tesis	7
2	Descripción del Sistema Actual	8
2.1	¿Qué es un Robot?	8
2.1.1	Sistema Mecánico	8
2.1.2	Sistema de Percepción	9
2.1.3	Sistema de Comunicación Hombre-Máquina (H-M)	10
2.1.4	Sistema de Control y de Toma de Decisiones	11
2.2	Robot <i>UNIMATE S-103</i>	12
2.2.1	Mecánica	13
2.2.2	Hardware	13
2.2.3	Modelo Cinematico del Robot <i>UNIMATE S-103</i>	18
2.3	Sistema de Visión	25
2.3.1	Introducción	25
2.3.2	Captura y preprocesamiento de la imagen	25
2.3.3	Segmentación	26
2.3.4	Reconocimiento	26
2.3.5	Entrenamiento del sistema de visión	33

2.4	Integración Robot-Visión	35
2.4.1	Estrategias de integración	35
2.4.2	Sustitución del <i>Teach Pendant</i>	42
2.4.3	Calibración del Sistema Robot-Visión.	45
3	Introducción a las Redes Neuronales	53
3.1	Introducción	53
3.2	Características	54
3.3	Aplicaciones	55
3.4	Clasificación	56
3.4.1	Topología.	56
3.4.2	Conectividad.	57
3.4.3	Aprendizaje.	57
3.4.4	Regla de Aprendizaje.	58
3.4.5	Asociación de información.	59
3.5	Funciones de Activación	59
3.5.1	Función de Activación Lineal.	59
3.5.2	Función Lineal con Saturación.	60
3.5.3	Función Escalón.	60
3.5.4	Función Sigmoidal.	61
3.5.5	Función Gaussiana.	62
3.6	Propiedades	62
3.6.1	Aprendizaje adaptable.	62
3.6.2	Sistemas dinámicos autoadaptables.	63
3.6.3	Autoorganización.	63
3.6.4	Generalización.	63
3.6.5	Tolerancia a fallos.	64
3.6.6	Operación en tiempo real	64
3.7	Perceptron	65
3.8	BackPropagation	66
4	Aplicación	69
4.1	Análisis y acondicionamiento del problema	69
4.1.1	Sistema Robot - Visión.	69
4.2	Diseño de la RN	71
4.2.1	Selección del paradigma	71
4.3	Entrenamiento	72
4.3.1	Entrenamiento en Simulación.	72
4.3.2	Entrenamiento en la Maqueta	76

5	Resultados y Conclusiones	82
5.1	Resultados	82
5.2	Conclusiones	87
5.3	Mejoras	87
6	Apendice A. Programas de Matlab	88
6.1	Entrenamiento en Simulación	88
6.2	Entrenamiento en la Maqueta	94
7	Referencias	99

Índice de Figuras

2.1	Diagrama funcional de un robot industrial	9
2.2	Identificación de los ejes del robot UNIMATE S-103	14
2.3	Dimensiones del robot UNIMATE S-103	15
2.4	Espacio de trabajo del robot UNIMATE S-103	16
2.5	Protocolo de comunicación C/ROS	17
2.6	Estructura cinemática del robot Unimate.	20
2.7	Plano x-y del robot Unimate.	22
2.8	Elemento Estructurante 4-conexo	26
2.9	Vecinos 8-conexos del punto p	27
2.10	Estrategia de integración Look and Move	37
2.11	Estructura de datos de la estrategia <i>Look and Move</i>	38
2.12	Control con retroalimentación visual	39
2.13	Estructura de datos de un controlador en el espacio articular.	40
2.14	Estructura de datos de un controlador en el espacio cartesiano.	41
2.15	Comunicación de información exteroceptiva.	44
2.16	Lectura de información propioceptiva	45
2.17	Plano de calibración de la cámara	46
2.18	Procedimiento de calibración de cámara.	47
2.19	Transformación del plano Rr' al plano Rm	51
3.1	Estructura de una Red Neuronal	54
3.2	Función Lineal	60
3.3	Función Lineal con Saturación	61
3.4	Función Escalon	62
3.5	Función Sigmondal	63
3.6	Función Gaussiana	64
3.7	Perceptron Monocapa	65

3.8	Distintas formas de las regiones generadas por un perceptrón multivel	67
4.1	Esquema de control actual.	70
4.2	Esquema visuomotor propuesto	70
4.3	Evolución del error RMS.	75
4.4	Error cometido por la RN usada como aproximador.	75
4.5	78
4.6	Aproximación inicial de la RN.	81
4.7	Evolución del error RMS.	82
4.8	Aproximación final de la RN.	83
4.9	Error cometido por la RN usada como aproximador.	84
4.10	Aproximación final de la RN en la fase de verificación.	85
4.11	Error cometido por la RN usada como aproximador en la fase de verificación.	86

Índice de Tablas

2.1	Parámetros de los ejes del robot UNIMATE S-103	18
2.2	Parámetros de las articulaciones	19
2.3	Incrementos al perímetro	28
2.4	Momentos discretos	30
2.5	Momentos discretos para las secciones negativas del contorno . . .	31
2.6	Momentos discretos para píxeles que pertenecen a las dos secciones del contorno.	31
2.7	Formato de comunicación del Teach Pendant.	43
4.1	Espacio de trabajo	73
4.2	Puntos de muestreo	73

1. Introducción

1.1. Tareas de ensamble robotizado

Uno de los principales problemas en la robotización de tareas de ensamble utilizando robots industriales tradicionales, es el alto costo y baja confiabilidad de todos los accesorios mecánicos que deben integrar el puesto de trabajo. En efecto, para que el robot pueda manipular todas las piezas que conforman un cierto subensamble, es necesario que el puesto de trabajo cuente con dispositivos mecánicos (despachadores, alimentadores, vibradores, etc.) que suministren al robot dichas piezas en la posición, orientación e instante adecuados. Para resolver de manera más eficiente este problema, se requiere que el robot disponga de sensores que le permitan detectar la presencia de objetos por ensamblar, así como su posición y orientación dentro del puesto de trabajo.

El ensamble es una de las aplicaciones de los robots industriales que demanda prestaciones más avanzadas. En estas aplicaciones se requiere de lenguajes de programación que permitan no sólo una especificación precisa de cada una de las fases de la tarea (toma del objeto, aproximación al sitio de ensamble, acomodo activo del objeto e inserción), sino que también facilite el uso y programación de sensores exteroceptivos. En este tipo de robots, se necesita implementar una sofisticada integración de sensores de tacto, fuerza, proximetría y visión entre otros.

De entre las posibles tecnologías a emplear en la solución de estos problemas destaca la Visión Artificial (VA) la cual permite resolver problemas aun más complejos que los mencionados.

1.2. Objetivo

El objetivo del presente trabajo es diseñar e implementar una Red Neuronal para realizar la reconstrucción del espacio articular de un robot *Unimation S-103* tipo Scara, a partir de información visual bidimensional (2D) de su espacio de trabajo y su utilización en el control de dicho robot. Las razones para hacerlo de esta manera se presentan a continuación

1.3. Por qué usar Visión Artificial

En una tarea de ensamble robotizado se tienen dos problemas. Se requiere de una serie de dispositivos mecánicos para asegurar que los objetos por manipular se encuentran en el lugar esperado y con la orientación necesaria; y por otro lado, se necesita que la programación de las posiciones de interés se haga con mucha precisión. Los dispositivos mencionados introducen una disminución en la fiabilidad del sistema, mientras que la programación se convierte en una tarea engorrosa cuando se tienen muchos puntos de interés. La utilización de un sistema de VA permite prescindir de accesorios mecánicos y de la exigencia de conocer con precisión y anticipación las posiciones y orientaciones de interés.

1.4. Por qué usar Redes Neuronales

A pesar de sus ventajas, el uso de VA introduce la necesidad de contar con modelos cinemáticos exactos y de una cierta potencia de cálculo para realizar las transformaciones necesarias para obtener las consignas articulares. El uso de redes neuronales (RN) para el cálculo de la cinemática inversa del sistema visión-robot, no necesita del conocimiento de ningún modelo, pues se basa en el aprendizaje de relaciones entrada-salida. Por otro lado, una vez entrenada la RN, su uso requiere de menos operaciones computacionales que las necesarias para realizar la transformación cinemática inversa por métodos tradicionales.

1.5. Antecedentes

En el Laboratorio de Robótica de la Sección de Control Automático se desarrolla un proyecto de investigación sobre la aplicación de técnicas de inteligencia artificial (*fuzzy logic*, redes neuronales) al control de robots, utilizando información sensorial de alto nivel. En este proyecto aparecen temas de investigación y de desarrollo muy variados y complejos de gran actualidad, lo cual genera una serie de problemas particulares que se está tratando de resolver mediante tesis de licenciatura y de posgrado, con temas integrados y coordinados adecuadamente en el proyecto mencionado. En el presente trabajo estamos interesados en aspectos computacionales y de comunicación relativos a la simulación por computadora y a la implementación en tiempo real de esquemas de control (para robots) basados en información sensorial provista por un sistema de visión artificial, utilizando redes neuronales. Los problemas particulares que se exponen en esta tesis son los de la realización de una maqueta en una computadora personal (PC) para la

simulación de esta clase de sistemas, basándonos en un paquete computacional comercial (Matlab[®]), así como la implementación de un sistema de comunicación entre un robot comercial, un sistema de VA y una PC.

1.6.Descripción de la Tesis

El presente trabajo de tesis aborda el problema de diseñar una Red Neuronal que sustituya el procedimiento de calibración y la cinemática inversa del sistema Robot-Visión, así como la elaboración de una metodología de adquisición de datos para el proceso de entrenamiento de dicha red.

En el Capítulo I se da una introducción al tema, incluyendo el objetivo de este trabajo y las razones para utilizar Redes Neuronales.

En el Capítulo II se hace una descripción detallada de los elementos del sistema Robot-Visión: el Robot *Unimate S-103* y el Sistema de Visión *Oculus 200*.

El Capítulo III aborda el tema de las Redes Neuronales. sus características, aplicaciones, propiedades, etc., así como el algoritmo de *Backpropagation*.

En el Capítulo IV se explica el procedimiento llevado a cabo para obtener la Red Neuronal, desde el análisis del problema hasta la obtención de la matriz de pesos de entrenamiento.

Finalmente, en el Capítulo V se presentan las conclusiones y el análisis de resultados del presente trabajo

2.Descripción del Sistema Actual

2.1.¿Qué es un Robot?

A pesar de sus limitaciones tecnológicas, el robot industrial actual (ciego, sordo, con capacidad de comunicación muy restringida, poca movilidad, sin iniciativa, pero infatigable y muy obediente) ha tenido un gran éxito debido a que puede automatizar sencillas, pero muy comunes, tareas productivas (carga y descarga de máquinas, herramientas, soldadura por puntos, pintura, etc.) con gran velocidad, precisión, confiabilidad y a costos competitivos. Además, sustituye a los trabajadores humanos de muchos puestos de trabajo insalubres o peligrosos y permite la realización de trabajos imposibles de asegurar con mano de obra convencional.

Un robot industrial es, según la RIA (Robotics Institute of America), un mecanismo articulado, programable, versátil y adaptable, capaz de manipular cargas o herramientas a lo largo de trayectorias previamente programadas. En esta definición destaca la característica de programabilidad, sin la cual el mecanismo articulado no puede ser considerado como un robot. Desde el punto de vista funcional, un robot industrial está formado por cuatro sistemas (figura 2.1) los cuales se describen brevemente a continuación:

2.1.1.Sistema Mecánico

Este sistema es el encargado de realizar el trabajo y está formado por la estructura mecánica del robot, los actuadores, las transmisiones y los accesorios de perirrobótica (órganos de prensión, mano izquierda, herramientas, bandas transportadoras, etc.) que le permiten la ejecución física de las tareas encomendadas. La estructura mecánica está generalmente constituida por una cadena cinemática abierta de eslabones rígidos unidos dos a dos por articulaciones simples de rotación o de traslación. Esta debe ser capaz de posicionar y orientar su herramienta en cualquier punto del espacio de trabajo, requiriéndose para ello de un brazo con tres grados de libertad (que puede obtenerse con una arquitectura cartesiana, cilíndrica, polar, tipo brazo articulado o "Scara") y de una mano con otros tres grados de libertad (en general, rotaciones alrededor de los tres ejes cartesianos).

Los actuadores ("músculos" del robot) pueden ser motores eléctricos o pistones hidráulicos, aunque también suelen usarse los neumáticos. Al principio eran muy utilizados los motores a pasos por la facilidad con que son controlados (en lazo abierto) pero, por lo mismo que no son muy precisos ni confiables, ahora sólo se

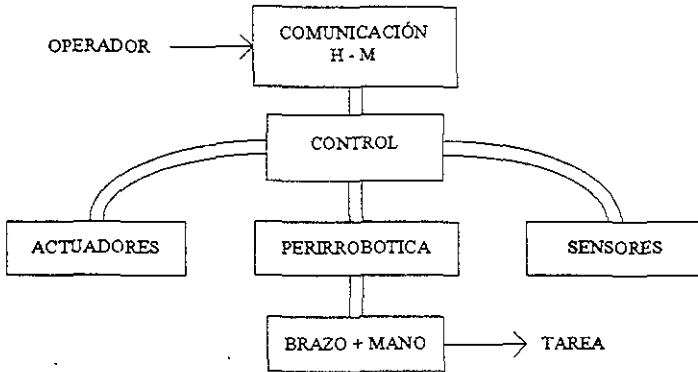


Figura 2.1. Diagrama funcional de un robot industrial

usan en robots de tipo didáctico. Por su robustez, confiabilidad y controlabilidad, han sido muy utilizados los motores de corriente continua, los cuales empiezan a ser desplazados por los de corriente alterna (especialmente diseñados para estas aplicaciones), debido su bajo costo ahora que se dispone de una tecnología de control adecuada para usarlos en servomecanismos de posición. Por lo que respecta a la transmisión de movimiento, diremos que los robots rápidos y precisos usados en ensamble (arquitectura Scara) no utilizan transmisiones, por lo que el motor está acoplado directamente a los ejes articulares (*direct-drive*), mientras que otros robots utilizan poleas y bandas dentadas, engranes con compensación del *backlash*, reductores sinfín-corona, etc

2.1.2. Sistema de Percepción

Este sistema está formado por los sensores del robot, que funcionalmente se clasifican en dos tipos: propioceptivos y exteroceptivos. Los llamados propioceptivos proporcionan información sobre el estado interno del robot, es decir, sobre la posición y la velocidad de cada una de sus articulaciones; la misión de estos sen-

sores es la de informar al sistema de control cuál es la posición real de cada una de las articulaciones del robot y a que velocidad se están moviendo. Entre los sensores de posición más utilizados para estos efectos tenemos los codificadores ópticos, los codificadores incrementales y los *resolvers*. Para medir la velocidad, generalmente se muestrea la señal digital proporcionada por los sensores de posición, con objeto de medir el tiempo que se necesita para pasar de una posición a la siguiente, aunque también se suele utilizar algún tipo de tacogenerador para medir las velocidades articulares. En el sistema de percepción propioceptiva se pueden incluir también sensores de presencia de objetos, interruptores de límite, interruptores de sincronización, etc. y aún los sensores de seguridad (por ejemplo sensores anticolidión).

Los sensores exteroceptivos pueden clasificarse en dos grandes grupos: sensores de contacto, que responden al contacto físico: tacto, deslizamiento, fuerza de interacción robot-entorno, y sensores de no contacto, basados en la respuesta de un detector a las variaciones en la radiación electromagnética o acústica, entre los cuales se pueden mencionar los sensores que miden: la distancia (rango), el posicionamiento relativo pinza-objeto y las propiedades visuales de un objeto. Estos últimos permiten al robot interactuar inteligentemente con su entorno, proporcionándole flexibilidad, versatilidad y adaptabilidad.

2.1.3. Sistema de Comunicación Hombre-Máquina (H-M)

Para que un operador pueda programar un robot, se requiere que éste disponga de un sistema de comunicación H-M, el cual puede ser de tres tipos: gestual, lenguaje de programación explícita y lenguaje de programación por objetivos.

El primero y más común de los métodos sigue siendo el gestual, en el cual el operador debe llevar manualmente al robot a todas y cada una de las distintas posiciones que conforman la tarea dada, registrándolas en la memoria. En este caso se dice que el robot "aprende" la tarea, para después poder repetirla. Para mover el robot en este tipo de programación por aprendizaje, se puede utilizar directamente su estructura mecánica (para lo cual, ésta debe permitir la transmisión de movimientos en los dos sentidos), o bien utilizar una botonera (*teach pendant*), una palanca de mando (*joy-stick*), o un robot copiador para generar las consignas a los servomecanismos del robot.

El método gestual resulta muy lento y tedioso aún para programar tareas relativamente simples, por lo cual fueron desarrollados lenguajes explícitos mediante

los cuales el operador programa los movimientos elementales del robot que le permiten realizar la tarea. Desde el principio de la década de los sesentas en que apareció el primero de estos lenguajes -nivel robot- (MHI desarrollado en el MIT), se han incorporado nuevas prestaciones como coordinación de movimientos, control sobre una trayectoria y algunas otras ligadas a los avances de la programación en general, entre las que destacan la programación estructurada, debiendo cumplir todas ellas con los siguientes requisitos: i) instrucciones de movimiento. ii) manejo de entradas/salidas (para sensores elementales *on/off* y accesorios de perirrobótica respectivamente), iii) control del flujo en la ejecución de un programa (saltos, iteraciones, etc.) y iv) apoyo a la programación (edición, depuración, telecargado, monitoreo, etc.). Algunos ejemplos de estos lenguajes son: AL, AML, VAL, PAL, MCL, MAL, etc.

En los métodos descritos anteriormente, los robots industriales son muy dependientes del operador, ya que es el responsable de especificarle detalladamente todos y cada uno de los movimientos elementales con los cuales podrá efectuarse la tarea requerida. A diferencia de los robots industriales, los robots inteligentes disponen de un método de programación implícito (nivel tarea) por medio del cual el operador tan sólo describe los objetivos de la tarea; de esta manera, el sistema de comunicación H-M de un robot inteligente permite la especificación de la tarea en términos de las operaciones que serán realizadas sobre los objetos. por ejemplo, si deseamos que el robot realice el ensamble de una pieza, programaríamos al robot con la instrucción <ensambla pieza> para que a partir de esta instrucción, se generen las especificaciones de cada uno de los movimientos del robot que permitan realizar la tarea descrita.

2.1.4. Sistema de Control y de Toma de Decisiones

Cuando el programa se introduce de manera gestual, el sistema de control (SC) tan sólo debe asegurar que las articulaciones alcancen los valores de posición memorizados en los instantes deseados, siguiendo una velocidad preestablecida. Para ello, el SC necesita conocer los valores memorizados y los valores actuales para cada articulación, calcular el error en posición correspondiente e intentar anularlo mediante un esquema de control en lazo cerrado (servomecanismo de posición), por ejemplo de tipo PID (Proporcional - Integral - Derivativo). La implementación en hardware del SC implica el uso de un microcontrolador para cada articulación, comandados por un procesador central tipo PC compatible.

Pero cuando se hace uso de un lenguaje de programación nivel robot, la misión del sistema de control es la de asegurar el control de la posición y la orientación del órgano terminal (pinza, herramienta) a cada instante a lo largo de la trayectoria deseada, necesiándose además de los servomecanismos en posición, de un sistema de coordinación de movimientos denominado generador de trayectorias; este método requiere de un hardware más complejo que el anterior. Una alternativa para solucionar este problema consiste en utilizar una red de multiprocesamiento (paralelo) o bien procesadores digitales de señal (DSP's). Desde el punto de vista algorítmico, se requiere de un control más eficiente, usándose para este caso esquemas adaptables de control multivariable y esquemas con ganancias variables preprogramadas tipo *gain scheduling*.

Resumiendo, para que un robot industrial pueda ejecutar una tarea, requiere de dos elementos esenciales: un lenguaje de programación mediante el cual se define de manera explícita la tarea a desarrollar y una descripción precisa y detallada de los objetos a manipular y del entorno del robot. Una opción para lograr una reducción sustancial de los tiempos de puesta a punto del proceso de manufactura robotizada, es la introducción de robots inteligentes a los cuales no es necesario definir con detalle la tarea, ni requieren de un conocimiento *a priori* del puesto de trabajo. Al proporcionar inteligencia al robot se pretende aumentar su autonomía y productividad, dotándolo de ciertas capacidades que le permitan, desde un punto de vista funcional, sustituir completamente a un operador humano en su puesto de trabajo.

2.2. Robot *UNIMATE S-103*.

El robot industrial utilizado en este trabajo, el *Unimate S-100*, ha sido diseñado para realizar tareas de manipulación y ensamble tipo *pick-and-place*. Por lo tanto, es relativamente rápido, con buena repetitividad y resolución adecuada para realizar ensambles de componentes electrónicos. Sin embargo, su sistema de programación y control, denominado C/ROS (Cyber/Robot Operating System), es muy rudimentario y cerrado: es decir, no acepta conexión con otras máquinas, no es posible considerar información complementaria de sensores exteroceptivos y no permite ordenes no previstas en su limitado conjunto de comandos e instrucciones, además, no es posible modificar su algoritmo de control ni actualizar sus ganancias en línea.

2.2.1. Mecánica

En nuestro caso particular, el sistema mecánico del robot está formado por un brazo de arquitectura SCARA (Selective Compliant Articulated for Assembly o Selective Compliant Arm for Robotic Assembly) con 3.5 grados de libertad (gdl), actuado por tres motores de CD y un sistema neumático todo-nada para el cuarto eje traslacional. La transmisión se efectúa por bandas de sincronización (primeros dos gdl) y por engranes cónicos para el tercer gdl. Las tres primeras articulaciones son rotacionales y servocontroladas alrededor de ejes verticales, mientras que la cuarta articulación es traslacional todo-nada a lo largo del eje vertical de rotación por lo cual se le considera, haciendo un abuso del lenguaje, como 1/2 gdl (figura 2.2). De este modo, el movimiento del Robot *Unimate S-103* se realiza en dos planos horizontales: uno superior o alto para navegación y el segundo inferior o bajo para tomar y dejar objetos. La mano del robot baja al plano inferior sólo cuando el brazo se ha detenido y no debe reiniciar su movimiento hasta que su mano regrese al plano superior.

La tabla 2.1 muestra los parámetros de los ejes del robot y la figura 2.3 las dimensiones de los eslabones. El espacio de trabajo del robot (figura 2.4) es un cilindro hueco de 61 cm. de radio y 11.4 cm. de profundidad.

2.2.2. Hardware

Sistema de Percepción. El robot *Unimate S-103* utiliza *resolvers* en el eje de los motores para medir la posición y la velocidad articular de cada grado de libertad, además tiene proximetros magnéticos para sensar los fines de carrera y la posición *home* de cada uno de los primeros tres ejes articulares. En el cuarto eje tiene tan sólo dos proximetros magnéticos para detectar en que posición se encuentra: retraído hacia arriba (plano de navegación) o desplegado hacia abajo (plano de trabajo). Las señales de los *resolvers* y de los proximetros de los primeros tres gdl se envían al servomecanismo correspondiente, mientras que la señal de los proximetros asociados al cuarto eje, se capta mediante los canales binarios de entrada del controlador. Las válvulas neumáticas de este último eje se manejan a través de los canales binarios de salida de dicho controlador.

Sistema de Control. Está formado básicamente por una tarjeta CPU (basada en el procesador Z-80A) encargada de coordinar el movimiento de los tres primeros

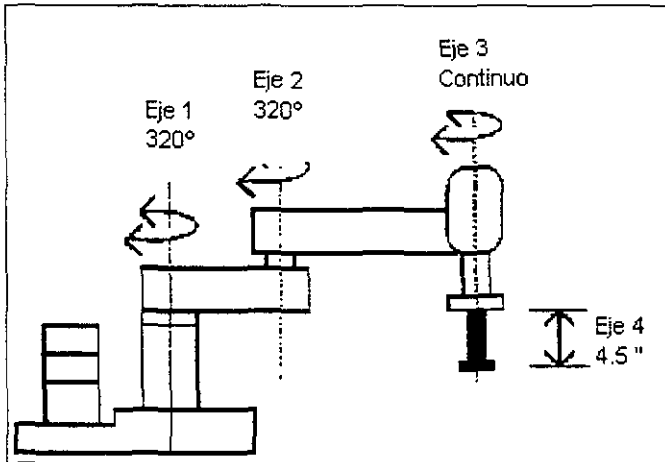


Figura 2.2: Identificación de los ejes del robot UNIMATE S-103

gdi (envío de consignas a los servomecanismos correspondientes) y de manejar los canales de entrada/salida (equivalentes a un controlador lógico programable para controlar los accesorios), una tarjeta de memoria y referencias (para los circuitos asociados a los *resolvers*), tres tarjetas (basada en el procesador Z-80A) para materializar los servomecanismos digitales de posición y tres servomecanismos analógicos de velocidad con sus respectivas etapas de potencia. Además, cuenta con un sistema de alimentación.

Sistema de Comunicación Hombre-Máquina La parte hardware está formada por una botonera (*Teach Pendant*) basada en un microcontrolador 6801, un teclado con 20 teclas multifuncionales, un display de dos líneas con 16 dígitos cada una y un enlace RS-232C con la CPU del controlador. La parte software consiste de un lenguaje propietario denominado C/ROS (Cyber/Robot Operating System) a partir del cual el operador tiene acceso a siete modos de operación del robot.

El paquete de software C/ROS que provee un sofisticado control de movimiento coordinado multiejes y entrada/salida con un mínimo de interacción con el operador, se encuentra almacenado en la memoria del controlador. Al ser un paquete diseñado específicamente para los robots de la serie 100 de Unimation,

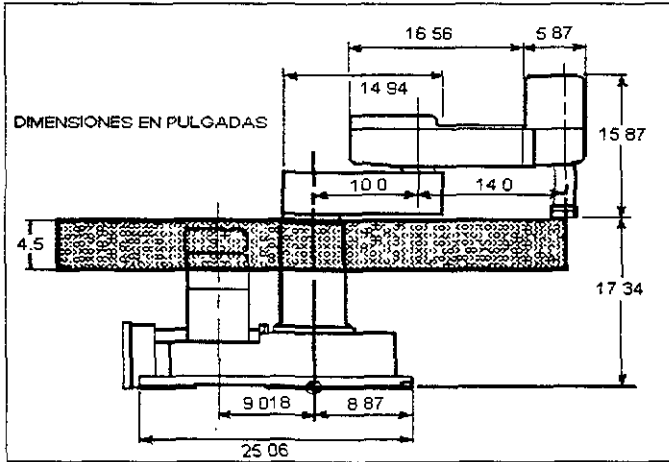


Figura 2.3: Dimensiones del robot UNIMATE S-103

contiene todas las funciones de control de los ejes articulares, coordinación de movimiento, sensado de todas las variables articulares a través de los sensores de posición, etc. La figura 2.5 muestra el protocolo C/ROS para el acceso a los diferentes modos de operación desde el *Teach Pendant*, los cuales se describen a continuación:

Startup Mode: Modo de arranque del robot, permite llevar el robot a su posición *home*, liberar el brazo robot y accesar cualquiera de los parámetros de los ejes o el sistema.

Program Mode: Permite la entrada de un programa de control del robot o la edición de un programa existente

Teach Mode: Permite enseñar al robot los puntos de interés en el espacio de trabajo para el programa creado la edición de estos puntos, el conocimiento del *status* de los módulos de entrada y la selección del *status* de los módulos de salida.

Execute Mode: Permite la ejecución del programa usuario considerando los puntos que se tengan en la memoria aprendidos en el modo *Teach*.

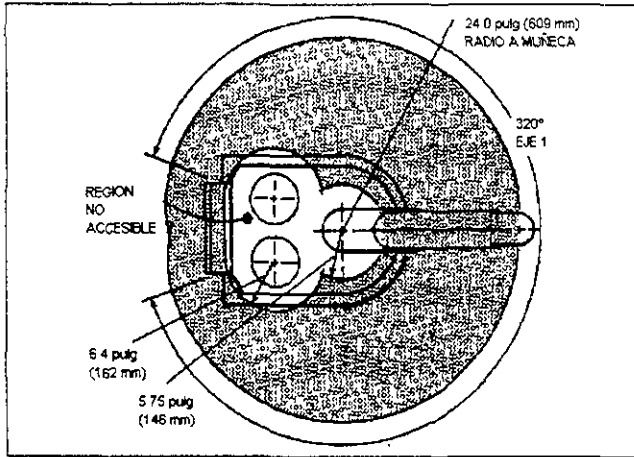


Figura 2.4: Espacio de trabajo del robot UNIMATE S-103

Load/Save Mode: Permite la transferencia de programas usuario y de sus puntos desde o hacia una computadora personal.

Pendant Look Mode: Bloqueo del *Teach Pendant*.

ClearDatasetMode: Usado para borrar el programa usuario o los puntos de la memoria del sistema.

El robot a utilizar posee una arquitectura *Scara* generalmente utilizada para manipular y ensamblar objetos en un plano de trabajo (mesa), obviamente de dos dimensiones. Por otro lado, las piezas industriales por manipular pueden ser consideradas como objetos planos, de altura conocida, sin restricciones en cuanto a su número ni de convexidad, aunque no pueden tocarse ni traslaparse. Estas consideraciones son de utilidad en el diseño del sistema de VA a utilizar

ENCENDIDO DEL CONTROLADOR

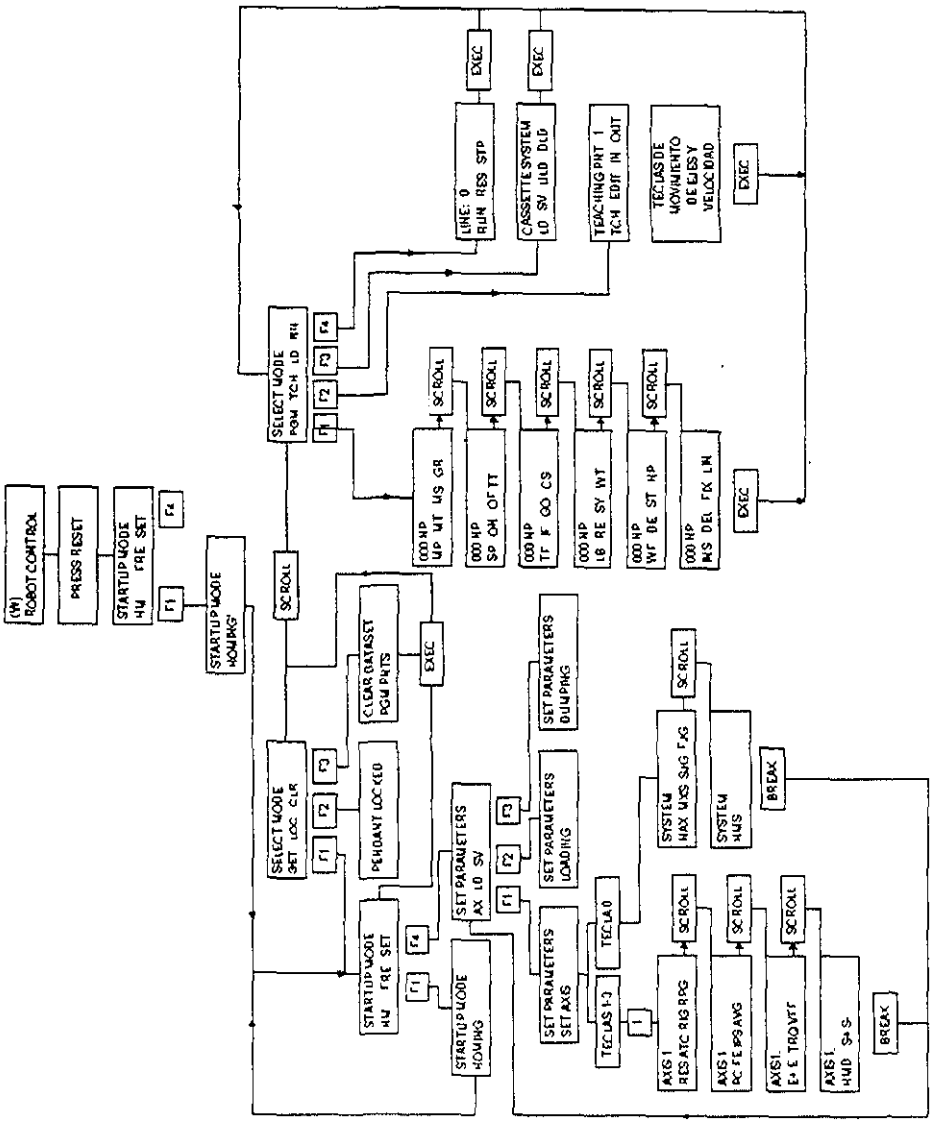


Figura 2.5: Protocolo de comunicación C/ROS

Parámetro	Eje 1	Eje 2	Eje 3	Eje 4
Rango de movimiento	$\pm 160^\circ$	$\pm 160^\circ$	continuo	4.5 pulg
Velocidad máxima	250 grad/seg.	250 grad/seg.	180 grad/seg.	-
Aceleración	30 rad/seg ²	30 rad/seg ²	15 rad/seg ²	-
Fuerza Axial	-	-	-	0.33 lb/psig.

Tabla 2.1: Parámetros de los ejes del robot UNIMATE S-103

2.2.3. Modelo Cinemático del Robot *UNIMATE S-103*.

Modelo Cinemático Directo La metodología a emplear para la obtención de este modelo es la propuesta por Denavit-Hartenberg (D-H). Para obtener el modelo cinemático de un robot mediante esta metodología, se empieza por definir los eslabones y las articulaciones de la cadena cinemática (brazo). Después se hace la detección de los ejes articulares y se asocia un referencial a cada uno de los eslabones de acuerdo con las reglas siguientes:

El eje z_{i-1} yace a lo largo del eje de la articulación.

El eje x_i es normal al eje z_{i-1} y apunta hacia afuera de él.

El eje y_i completa el sistema de coordenadas dextrógiro.

Cada eslabón está caracterizado por dos parámetros:

a_i : Es la distancia de separación desde la intersección del eje z_{i-1} con el eje x_i hasta el origen del sistema i -ésimo a lo largo del eje x_i (o la distancia más corta entre los ejes z_{i-1} y z_i)

α_i : Es el ángulo medido entre el eje z_{i-1} y el eje z_i con respecto al eje x_i (el signo de este ángulo se obtiene utilizando la regla de la mano derecha).

Mientras que las articulaciones se caracterizan por otros dos parámetros.

θ_i : Es el ángulo de la articulación medido entre el eje x_{i-1} y el eje x_i con respecto al eje z_{i-1} (el signo de este ángulo se obtiene utilizando la regla de la mano derecha).

Articulación	a_i	α_i	d_i	θ_i
1	a_1	0°	0	θ_1
2	a_2	180°	0	θ_2
3	0	0°	d	0
4	0	0°	0	θ_3

Tabla 2.2: Parámetros de las articulaciones

d_i : Es la distancia desde el origen del sistema de coordenadas $(i - 1)$ -ésimo hasta la intersección del eje z_{i-1} con el eje x_i medida a lo largo del eje z_{i-1} .

En la figura 2.6 se muestra un esquema del robot *Unimate S-103* con la ubicación de los referenciales asociado a cada uno de sus cuatro eslabones. Los parámetros cinemáticos de este robot, también llamados de D-H, aparecen en la tabla 2.2.

La línea i en esta tabla permite encontrar la matriz de paso generalizada entre el eslabon i y el eslabon $i-1$, denominada A_i , de acuerdo con la siguiente fórmula:

$$A_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) \cos(\alpha_i) & \sin(\theta_i) \sin(\alpha_i) & a_i \cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \cos(\alpha_i) & -\cos(\theta_i) \sin(\alpha_i) & a_i \sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.1)$$

Obtenida por aplicación sucesiva de las cuatro transformaciones que permiten confundir dos referenciales sucesivos

Una vez derivadas las n matrices de paso generalizadas ($n = \text{gdl}$), el modelo cinemático directo se calcula mediante la multiplicación de estas n matrices de acuerdo a la ecuación 2.2.

$$T_0^n = \prod_{i=1}^n A_i \quad (2.2)$$

Aplicando la ecuación 2.1 a cada uno de los renglones de la tabla 2.2, obtenemos:

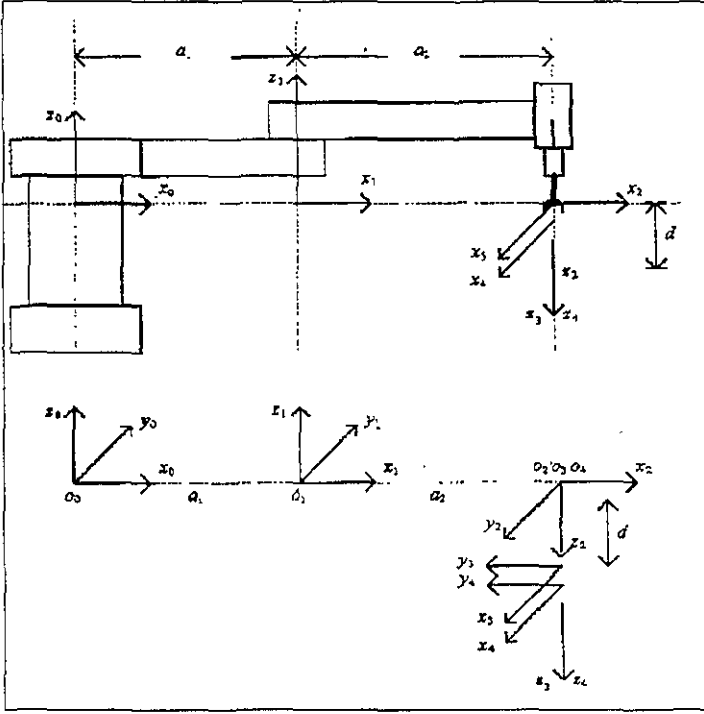


Figura 2.6: Estructura cinemática del robot Unimate

$$A_1 = \begin{bmatrix} c_1 & -s_1 & 0 & a_1 c_1 \\ s_1 & c_1 & 0 & a_1 s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.3)$$

$$A_2 = \begin{bmatrix} c_2 & s_2 & 0 & a_2 c_2 \\ s_2 & -c_2 & 0 & a_2 s_2 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.4)$$

$$A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.5)$$

$$A_4 = \begin{bmatrix} c_3 & -s_3 & 0 & 0 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.6)$$

en donde se utilizó la notación abreviada:

$$s_i = \sin(\theta_i)$$

$$c_i = \cos(\theta_i)$$

Utilizando estas matrices y la ecuación 2.2 tenemos:

$$T_0^4 = \begin{bmatrix} c_{12}c_3 + s_{12}s_3 & -c_{12}s_3 + s_{12}c_3 & 0 & a_1c_1 + a_2c_{12} \\ s_{12}c_3 - c_{12}s_3 & -s_{12}s_3 - c_{12}c_3 & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & -1 & -d \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.7)$$

en donde:

$$s_{i,j} = \sin(\theta_i + \theta_j)$$

$$c_{i,j} = \cos(\theta_i + \theta_j)$$

Como se mencionó anteriormente, el cuarto movimiento del robot *Unimate S-103* es del tipo todo-nada (no servocontrolado), lo cual permite al robot trabajar sólo en dos planos horizontales diferentes uno para desplazar el órgano terminal (mano arriba: $z = 0$) y otro para agarrar o soltar los objetos (mano abajo: $z = -d$) en un plano fijo en el cual se encuentra la banda transportadora, la tornamesa, la mesa de ensamble, etc

Por tal motivo y debido a que θ_4 , ángulo de orientación de los objetos, es proporcionado por el sistema de VA, en el modelo cinemático directo se consideró que el vector de posición del efector final esté compuesto tan sólo por dos coordenadas:

$$x = a_1 \cos(\theta_1) + a_2 \cos(\theta_1 + \theta_2) \quad (2.8)$$

$$y = a_1 \sin(\theta_1) + a_2 \sin(\theta_1 + \theta_2) \quad (2.9)$$

Modelo Cinemático Inverso. Dadas las dificultades de obtención del modelo cinemático inverso mediante la técnica de transformación inversa, no obstante que se trata de un robot de poca complejidad, se aplica un método geométrico para la obtención de las variables articulares en función de las coordenadas cartesianas del efector final.

Solución para las dos primeras articulaciones. Considérese la vista esquemática en planta del robot mostrada en la figura 2.7

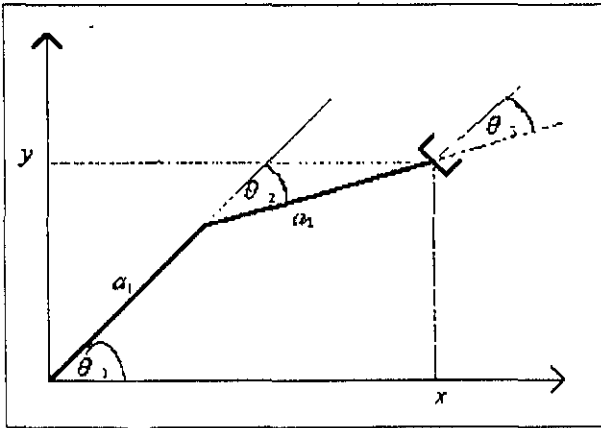


Figura 2.7 Plano x-y del robot Unimate.

Aplicando la ley del coseno al diagrama de la figura 2.7, se tiene:

$$\cos(\theta_2) = \frac{x^2 + y^2 - a_1^2 - a_2^2}{2a_1a_2} \quad (2.10)$$

$$\sin(\theta_2) = \pm \sqrt{1 - \cos^2(\theta_2)} \quad (2.11)$$

Sustituyendo 2.10 en 2.11 y dividiendo miembro a miembro ambas ecuaciones se obtiene:

$$\theta_2 = tg^{-1} \frac{\pm \sqrt{1 - \left[\frac{x^2 + y^2 - a_1^2 - a_2^2}{2a_1 a_2} \right]^2}}{\frac{x^2 + y^2 - a_1^2 - a_2^2}{2a_1 a_2}} \quad (2.12)$$

De manera similar se obtiene:

$$\theta_1 = tg^{-1} \frac{y}{x} - tg^{-1} \frac{a_2 \sin(\theta_2)}{a_1 + a_2 \cos(\theta_2)} \quad (2.13)$$

Las ecuaciones 2.12 y 2.13 representan el modelo cinemático inverso para el posicionado del efector final del robot. Con el propósito de dar una solución consistente al cálculo de los ángulos articulares de modo que permita evaluar θ_1 y θ_2 en todo el plano, o sea $-\pi \leq \theta \leq \pi$, se usa en la aplicación una función arcotangente de dos argumentos, $arctan(y, x)$, que devuelve $tg^{-1}(y/x)$ ajustada al cuadrante apropiado y que se define de la siguiente manera:

$$\theta = arctg2(y, x) = \left\{ \begin{array}{ll} 0 \leq \theta \leq 90 & \text{para } +x, +y \\ 90 \leq \theta \leq 180 & \text{para } -x, +y \\ -180 \leq \theta \leq -90 & \text{para } -x, -y \\ -90 \leq \theta \leq 0 & \text{para } +x, -y \end{array} \right\} \quad (2.14)$$

Solución para la tercera articulación rotacional. La tercera articulación rotacional θ_3 , define la orientación del órgano terminal del robot. De la figura 2.7 se observa que la orientación deseada se obtiene mediante la siguiente ecuación:

$$\theta_3 = \alpha - (\theta_1 + \theta_2) \quad (2.15)$$

donde:

α ángulo de orientación deseado, medido entre la horizontal (eje x) y el eje de la pinza

Las ecuaciones 2.12, 2.13 y 2.15 representan entonces el modelo cinemático inverso del robot, para cuya implementación practica, es necesario tener en cuenta la forma en que son medidas las posiciones articulares por los sensores de posición.

o sea, la forma en que están representadas en los registros de posición que ofrece el C/ROS. En este caso, las posiciones articulares θ_1 y θ_2 son medidas respecto al eje x positivo, mientras que θ_3 es medida respecto al segundo eslabón de la cadena cinemática.

Esto motiva que al ajustar las posiciones articulares para llevar el efector final a la posición y orientación que se desea, a partir de la solución del modelo cinemático inverso, se deben dar las modificaciones que dan las ecuaciones 2.16a, 2.16b y 2.16c, que constituyen los valores reales que hay que enviar al controlador del robot para lograr el objetivo que se desea.

$$\theta_{1(robot)} = \theta_{1(calculado)} \quad (2.16a)$$

$$\theta_{2(robot)} = \theta_{2(calculado)} + \theta_{1(robot)} \quad (2.16b)$$

$$\theta_{3(robot)} = \alpha - (\theta_{1(robot)} + \theta_{2(robot)}) \quad (2.16c)$$

2.3. Sistema de Visión

2.3.1. Introducción

La tarea del sistema de visión consiste en la identificación y localización de objetos dentro del espacio de trabajo del robot. El sistema de visión implantado en la integración robot-visión presentada en este trabajo, toma como elemento primario la tarjeta de adquisición de imágenes Oculus-200, dicho sistema tiene asignadas las siguientes tareas:

- Captura y preprocesamiento de la imagen (realizada por la tarjeta).
- Segmentación (realizada por el software).
- Reconocimiento (realizada por el software).
- Entrenamiento o Aprendizaje.

Como resultado final del desarrollo de estas etapas se obtiene la posición y orientación en coordenadas imagen del componente deseado dentro del espacio de trabajo del robot.

2.3.2. Captura y preprocesamiento de la imagen

La imagen empleada en el sistema integrado forma una matriz de 400x400 píxeles (codificados en 128 niveles de gris). Debido a que en el proceso de reconocimiento se emplean invariantes a cambio de escala, en su representación en memoria la matriz es comprimida a 200 x 200 píxeles (codificados en 128 niveles de gris), lo cual cumple un doble objetivo: por un lado tener un campo de vista de la cámara que permita "ver" si no todo, gran parte del espacio de trabajo del robot, y por otro un considerable ahorro de memoria con una pérdida mínima de información visual consiguiendo con ello una disminución importante del tiempo de procesamiento. Dadas las características del Oculus-200, el acceso a cada píxel requiere de 3 operaciones de salida y 1 de entrada por puertos de E/S, lo que representa una desventaja en relación al tiempo total de procesamiento.

Durante el preprocesamiento se eliminan ruidos presentes en la imagen, lo cual se realiza mediante una erosión con un elemento estructurante 4-conexo tal como se muestra en la figura 2.8. La erosión binaria es una operación morfológica que se define de la siguiente manera

Definición: Sean A y B conjuntos en un espacio euclideo n -dimensional, entonces la erosión de A por B es el conjunto de todos los elementos x para los cuales $x + b \in A$ para todo $b \in B$.

$$A \ominus B = \{x \in E^n : x + b \in A, \forall b \in B\} \tag{2.17}$$

Dicho de otra manera, la erosión de A por B es el conjunto de todos los elementos x de E^n para los cuales B trasladado a x está contenido completamente en A .

$$A \ominus B = \{x \in E^n : B_x \subseteq A\} \tag{2.18}$$



Figura ~2.8: Elemento Estructurante 4-conexo

2.3.3.Segmentación

Para la segmentación de la imagen se parte del hecho de tener buen contraste entre los objetos de interés y el fondo, es por ello que se aplica una de las técnicas más sencillas: la binarización. Este aspecto no resta generalidad a la solución del problema por ser algo alcanzable aún en un ambiente industrial. La binarización consiste en partir en dos el histograma de la imagen: todos los píxeles del fondo se asocian a un sólo valor de gris y los píxeles del objeto se pueden asociar también a un valor de gris o bien dejarlos como estaban. El umbral es fijado de forma manual partiendo del hecho de que una umbralización automática, no obstante de su efectividad, consume mucho tiempo de cálculo. Al iniciarse la ejecución del sistema implementado, el umbral es inicializado en el nivel de gris 35, el cual se encontró aceptable en las condiciones de iluminación del laboratorio. Este valor puede ser ajustado hasta lograr un nivel de binarización adecuado.

2.3.4.Reconocimiento

Para la descripción y posterior reconocimiento de los objetos dentro del campo de vista de la cámara, es necesario establecer un vector de atributos que permita la

diferenciación de cada clase u objeto. Los componentes de este vector de atributos son:

- El factor de compactación
- Los momentos invariantes

Tales componentes fueron elegidos por la robustez que ofrecen a tal objetivo, además de la velocidad de cálculo que se alcanza con los algoritmos utilizados.

Partiendo del hecho de ser el contorno una de las formas más naturales y sencillas de representar los objetos que nos rodean, se estableció un algoritmo de seguimiento de contornos para regiones 8-conexas, el cual a su vez proporciona información correspondiente a la región encerrada que finalmente posibilita el cálculo exacto de los componentes del vector de atributos. El algoritmo de seguimiento de contornos empleado adopta un sentido de recorrido horario de acuerdo a la forma en que se establecen los 8 vecinos del píxel analizado y que se muestra en la figura 2.9.

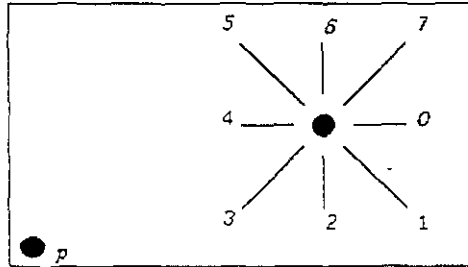


Figura 2.9: Vecinos 8-conexos del punto p.

El factor de compactación. El factor de compactación es un atributo geométrico de los objetos muy empleados en tareas de clasificación por su invarianza a transformaciones tales como traslación, rotación y cambios de escala. Este factor se define de la siguiente manera

$$f_{comp} = \frac{P^2}{A} \tag{2.19}$$

Dirección	Valor agregado
0	largo
1	diagonal
2	ancho
3	diagonal
4	largo
5	diagonal
6	ancho
7	diagonal

Tabla 2.3: Incrementos al perímetro

donde:

f_{comp} : Factor de compactación.

P : Perímetro del objeto.

A : Área del objeto.

Para la obtención del perímetro del objeto se emplea un algoritmo de seguimiento de contornos. Este algoritmo toma en cuenta la dirección en que son encontrados los píxeles (figura 2.9) y aplica la tabla 2.3 para conocer los incrementos que deben sumarse a un acumulador iniciado en cero. De esta manera, al término del recorrido el acumulador contendrá el valor del perímetro.

En la tabla 2.3, largo, ancho y diagonal representan las dimensiones del píxel, las cuales fueron encontradas de manera experimental y que corresponden a los valores de 1.44, 0.88 y 1.04 respectivamente.

El área del objeto es obtenida a través del algoritmo de computación de los momentos invariantes, representada por el momento de orden 0 (m_{00}) y será vista más adelante.

Momentos Invariantes. La teoría de momentos provee una alternativa usual en la representación de formas de objetos. El concepto de momentos invariantes se basa en la teoría de invariantes, la cual trata con las propiedades de ciertas clases

de expresiones algebraicas que no varían a transformaciones lineales tales como la rotación, traslación y escalamiento. Los momentos invariantes han sido utilizados para describir o caracterizar objetos en procesos de reconocimiento cuando se tiene unicidad de forma pero sin perspectiva, independientemente de su localización, medida y orientación.

Para obtener los momentos se optó por emplear un algoritmo basado en el contorno de un objeto, ya que como se mencionó anteriormente, requiere menos tiempo de cálculo además de proporcionar resultados exactos.

Algoritmo de Philips modificado. El algoritmo de Philips fue considerado el más ventajoso en este aspecto; sin embargo, presentaba la desventaja de no analizar la presencia de hoyos en los objetos. Debido a esto, el algoritmo de Philips fue modificado con el objeto de aumentar su velocidad en algunos casos y generalizar su uso a cualquier tipo de objeto plano.

Las modificaciones llevadas a cabo en [13] fueron:

- Por un lado, extender la integración a ambos ejes del plano, es decir, el algoritmo de Philips define el borde o contorno de un objeto respecto al eje x ; el algoritmo de Philips modificado define el borde o contorno de un objeto respecto a ambos ejes del plano. Esta modificación, en conjunto con un algoritmo de decisión para la determinación de la pertenencia de un píxel a las diferentes secciones del contorno, simplificó las expresiones de cálculo de los momentos y condujo a un ahorro de tiempo.
- Por otro, introducir un algoritmo que considera la presencia de hoyos en los objetos.

Los resultados obtenidos integrando a ambos ejes del plano (x y y), son iguales y exactos por lo que es posible adecuar las expresiones de cálculo de los diferentes momentos a integración en uno u otro eje sin alterar el resultado final. La tabla 2.4 resume la adecuación de las expresiones de cálculo de los momentos hasta un orden no mayor que 3 en la cual se ha usado integración respecto al eje x para los 5 primeros momentos y respecto a y para los 5 restantes.

Momento	Valor
m_{00}	$x + 1$
m_{01}	$(x + 1)y$
m_{02}	$(x + 1)y^2$
m_{03}	$(x + 1)y^3$
m_{12}	$\frac{1}{2}(x + 1)xy^2$
m_{10}	$(y + 1)x$
m_{11}	$\frac{1}{2}(y + 1)xy$
m_{20}	$(y + 1)x^2$
m_{21}	$\frac{1}{2}(y + 1)yx^2$
m_{30}	$(y + 1)x^3$

Tabla 2.4: Momentos discretos

Bajo estas consideraciones, si las expresiones de cálculo de los momentos dadas por la tabla 2.4 se aplican a las secciones negativas del contorno, las expresiones resultantes se resumen en las mostradas por la tabla 2.5, donde los valores de x y y son las coordenadas de los píxeles pertenecientes al contorno real del objeto.

El aporte de cada píxel del contorno a los momentos discretos, es sumado o restado a un acumulador iniciado en cero dependiendo de la pertenencia del píxel. Esto hace que en aquellos píxeles que pertenecen a ambas secciones del contorno, el aporte total a los momentos del objeto este dado por la diferencia entre las expresiones dadas por la tabla 2.4 y 2.5, la cual se resume en la tabla 2.6.

Haciendo un análisis de la complejidad del método modificado y tomando en cuenta sólo las operaciones necesarias para el cálculo de los momentos, se tienen.

$$4 \times (n_{x+} + n_{x-}) + 5 \times (n_{y+} + n_{y-}) \text{ multiplicaciones}$$

$$1 \times (n_{x+} + n_{x-}) - 1 \times (n_{y+} + n_{y-}) \text{ adiciones.}$$

donde n_{x+} y n_{x-} son el número de píxeles que pertenecen a las secciones positiva y negativa del contorno definido respecto al eje x , y n_{y+} y n_{y-} son el

Momento	Valor
m_{00}	x
m_{01}	xy
m_{02}	xy^2
m_{03}	xy^3
m_{12}	$\frac{1}{2}(x-1)xy^2$
m_{10}	yx
m_{11}	$\frac{1}{2}(y-1)yx$
m_{20}	yx^2
m_{21}	$\frac{1}{2}(y-1)yx^2$
m_{30}	yx^3

Tabla 2.5: Momentos discretos para las secciones negativas del contorno

Momento	Valor
m_{00}	1
m_{01}	y
m_{02}	y^2
m_{03}	y^3
m_{12}	xy^2
m_{10}	x
m_{11}	yx
m_{20}	x^2
m_{21}	yx^2
m_{30}	x^3

Tabla 2.6: Momentos discretos para pixels que pertenecen a las dos secciones del contorno.

número de píxeles que pertenecen a las secciones positiva y negativa del contorno definido respecto al eje y .

De la tabla 2.6 se concluye que en aquellos píxeles que pertenecen a ambas secciones del contorno (positiva y negativa), la complejidad del método se reduce a:

$3 \times n_{x+/-} + 4 \times n_{y+/-}$ multiplicaciones.
0 adiciones.

donde $n_{x+/-}$ y $n_{y+/-}$ son el número de píxeles que pertenecen a las secciones positiva y negativa.

Usando la misma nomenclatura, la complejidad del algoritmo original de Philips es:

$9 \times (n_{x+} + n_{x-})$ multiplicaciones.
 $2 \times (n_{x+} + n_{x-})$ adiciones.

Si se comparan ambos métodos se concluye lo siguiente:

Si $(n_{x+} + n_{x-}) < (n_{y+} + n_{y-})$ el algoritmo de Philips resulta más rápido que el modificado.

Si $(n_{x+} + n_{x-}) > (n_{y+} + n_{y-})$ el algoritmo de Philips resulta más lento que el modificado.

Si $(n_{x+} + n_{x-}) = (n_{y+} + n_{y-})$ el algoritmo de Philips resulta igual o más lento que el modificado.

La modificación introducida al algoritmo original propuesto por Philips representa una mejora en cuanto al tiempo de cálculo de los momentos discretos a través del contorno viéndose superado únicamente en el caso de regiones alargadas en la dirección exactamente horizontal, cuya probabilidad de ocurrencia es muy inferior a los otros casos en que el algoritmo modificado supera al original.

Consideración de hoyos. Una de las desventajas principales del algoritmo original de Philips es que es aplicable sólo a objetos sin hoyos, lo que restringe su rango de aplicación.

Para la aplicación práctica del método propuesto, fue necesario introducir un algoritmo que detectara la presencia de hoyos dentro de un objeto; es decir, determinar cuándo un contorno interior pertenece a un objeto dentro de la imagen. En este sentido, durante el seguimiento del contorno de la imagen se realizó una marcación de los píxeles que forman parte de éste, de modo que al encontrarse un contorno interior se tiene conocimiento del objeto al cual pertenece.

Finalmente, una vez computados los momentos hasta el orden 3, se obtienen los invariantes 1 y 2 de Hu, que junto con el factor de compactación forman el vector de atributos que identifica a cada objeto. El proceso de reconocimiento se completa con la comparación del vector de atributos de los objetos presentes en la escena con el vector correspondiente a la clase de objeto que se desea manipular y que permanece almacenado en una base de datos, creada previamente durante un proceso de entrenamiento del sistema de visión.

Al detectarse la presencia de un objeto en la imagen, se lleva a cabo esta comparación. Si el objeto detectado es el que se desea manipular, se determina su centro de masa y la orientación en coordenadas imagen de acuerdo a las ecuaciones 2.20, 2.21 y 2.22.

Centroide:

$$m_x = \frac{m_{10}}{m_{00}} \quad (2.20)$$

$$m_y = \frac{m_{01}}{m_{00}} \quad (2.21)$$

Orientación:

$$\theta = \frac{1}{2} \tan^{-1} \frac{2\mu_{11}}{\mu_{20} - \mu_{02}} \quad (2.22)$$

2.3.5. Entrenamiento del sistema de visión.

El objetivo de la etapa de entrenamiento es que el sistema de visión "conozca" de antemano los posibles objetos que se pueden presentar en la escena

En esencia el procedimiento consiste en asociar a cada clase de objetos C_i un vector de atributos X_i , resultando un modelo (C_i, X_i) . Para cada modelo, C_i representa el nombre del objeto y X_i su vector de atributos.

$$(C_i, X_i) = \left\{ \begin{array}{l} \text{Nombre} \\ f_{comp} \\ m_x \\ m_y \end{array} \right\} \quad (2.23)$$

Dadas las imperfecciones de todo sistema de procesamiento de imágenes, entre las que se pueden señalar ruido introducido en la discretización de la imagen, variaciones en las condiciones de iluminación, el fenómeno de perspectiva, etc., la medición del vector de atributos de una misma clase sufre variaciones de una imagen a otra, lo que motiva la imposibilidad de establecer valores constantes a sus componentes, sino una media y su varianza. Además, para aumentar la certidumbre de los datos obtenidos se hace necesario tomar el mayor número de muestras posibles, trasladando, rotando y escalando el objeto bajo aprendizaje por todo el campo visual de la cámara.

Una vez que el objeto que se desea enseñar al sistema de visión ha sido mostrado un número de veces considerable, se obtienen la media y la varianza de cada componente del vector de atributos, que junto con el nombre del objeto, es almacenado en un archivo que conforma una base de datos accesada durante la etapa de reconocimiento.

2.4. Integración Robot-Visión

2.4.1. Estrategias de integración

En un sistema de manipulación asistida por visión, el robot puede necesitar imágenes provenientes de una cámara móvil colocada en su mano o bien, para aplicaciones más simples, pueden bastarle imágenes de una cámara fija. En ambos casos, puede ser suficiente la utilización de imágenes en dos dimensiones 2D (es decir, objetos relativamente planos, que aparecen en una misma posición de equilibrio y que yacen o se mueven a lo largo de trayectorias previsibles en un plano de trabajo) o pueden necesitarse imágenes en tres dimensiones 3D cuando, además de la manipulación de objetos, se requiere su ensamble en sitios o situaciones complejas.

Las diversas estrategias de integración Robot-Visión, se pueden dividir en dos grupos:

Integración con cámara fija (*Fixed-eye*).

Integración con cámara móvil (*Eye-in-hand*).

Integración con cámara fija (*Fixed-eye*). La característica distintiva fundamental de esta forma de integración, radica en el hecho de usar una cámara estática que proporcione una vista completa de la escena, lo cual permite un tratamiento más flexible a las tareas de procesamiento de imágenes. En esta forma, la misma vista de la cámara puede ser usada tanto para el conocimiento del espacio de trabajo del robot, como la fuente primaria de información en el caso en que el sistema de visión sea integrado directamente dentro del lazo de control del robot.

Existen diversas estrategias dentro de esta forma de integración, entre las cuales se pueden mencionar:

Integración en lazo abierto.

Integración en lazo cerrado

Integración en lazo abierto. Constituye la estrategia tradicional de integración y consta de dos fases: ver y mover (figura 2.10), lo cual le da nombre a dicha estrategia (*Look and Move*). La fase *Look* opera como un lazo externo usando información derivada de la cámara, generando las consignas (valores deseados)

a un controlador de espacio articular interno que constituye el centro de la fase *Move* y que se alimenta de la información sensorial propioceptiva de los sensores de posición del robot.

La cadena de procesamiento que esta estrategia involucra en una tarea de mover el robot para tomar una pieza dentro de su espacio de trabajo, se puede resumir de la siguiente manera:

1. El sistema de visión identifica y localiza la pieza dentro del campo de visión de la cámara, obteniendo la posición y orientación de la pieza en coordenadas imagen.
2. A través de un algoritmo de calibración de cámara, se obtienen los parámetros cinemáticos necesarios para realizar la transformación de coordenadas imagen a coordenadas cartesianas.
3. La posición y orientación de la pieza son transformadas de coordenadas imagen a coordenadas cartesianas mediante el modelo cinemático inverso del sistema robot-visión (basado en los parámetros que proporciona el algoritmo de calibración de cámara).
4. Un transformador de coordenadas evalúa el modelo cinemático (geométrico) inverso del robot, obteniéndose la posición y orientación deseadas (consignas) en coordenadas articulares. Con esta información, además del conocimiento de la posición corriente brindada por los sensores propioceptivos del robot, este transformador evalúa la trayectoria a seguir.
5. El controlador aplica los pares demandados a los servomecanismos articulares que anulan el error en posicionado, controlando en todo momento la trayectoria del órgano terminal del robot (posición corriente).

Es evidente que el logro del objetivo final, o sea, llevar el efector final a la posición y orientación de la pieza, sólo es posible si se tiene:

- Un conocimiento exacto de los parámetros cinemáticos usados en las transformaciones.
- Un algoritmo preciso para la transformación de coordenadas imagen a coordenadas del mundo real (basado en el algoritmo de calibración de cámara).

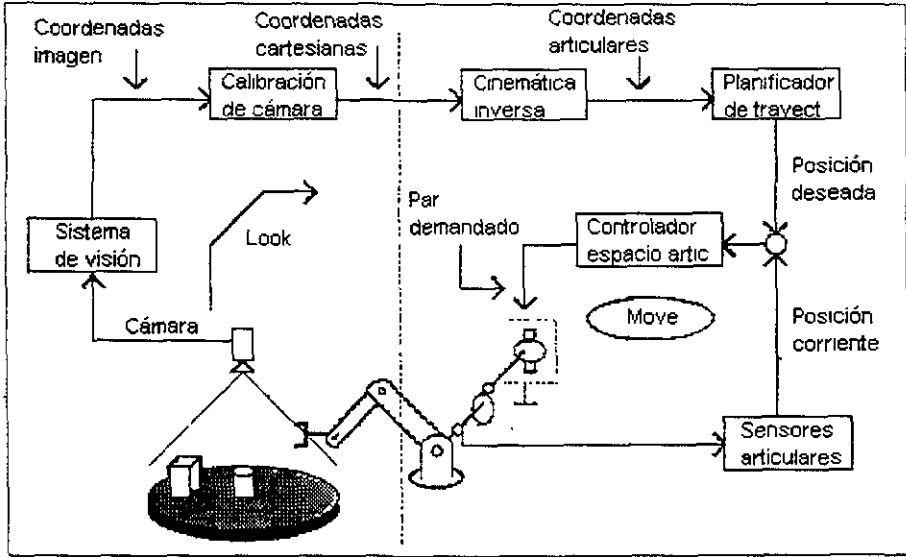


Figura 2.10: Estrategia de integración Look and Move

- Un algoritmo preciso para evaluar el conjunto de posiciones articulares necesarias para colocar el efector final en la posición y orientación deseadas (basado en el modelo cinemático del robot)
- Sensores propioceptivos perfectos.
- Un sistema de cómputo de alta resolución

En la aplicación práctica de esta estrategia, la degradación del comportamiento ideal descrito anteriormente es inevitable, debido a la presencia de imperfecciones generalmente en todos los aspectos mencionados: aproximaciones en los modelos, resolución limitada de los sensores y de la CPU, etc.

La estructura funcional de un sistema basado en esta estrategia se muestra en la figura 2.11.

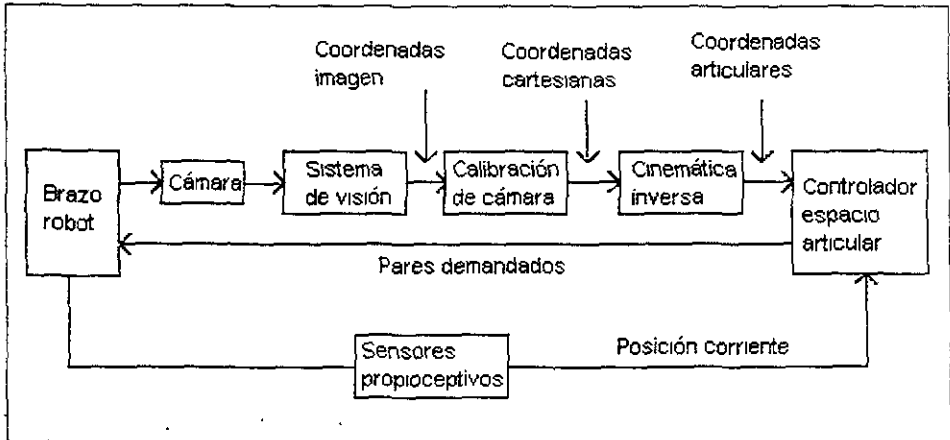


Figura 2.11: Estructura de datos de la estrategia *Look and Move*

Integración en lazo cerrado. Lo que diferencia notoriamente esta estrategia de la anterior, es el uso de la información visual dentro del lazo de control, teniéndose así una retroalimentación visual tanto de la posición del objeto que se desea manipular como de la posición corriente del efector final, quedando resumida la tarea de tomar un objeto en mover el brazo del robot hasta que el sistema de VA "vea" que el efector final esté sobre el componente deseado.

La integración del sistema de visión en el lazo de control potencialmente puede eliminar la necesidad de sensores precisos, toda vez que la localización del efector final no se determina aplicando la transformación cinemática directa a los ángulos articulares derivados de los sensores propioceptivos. De esta manera, el sistema de VA puede utilizarse eficientemente para medir el error de posicionamiento relativo con una resolución ajustable (*zoom*) la cual anula la necesidad de contar con modelos precisos y sensores propioceptivos de alta resolución. La estructura funcional de esta estrategia puede verse en la figura 2.12

En la implementación de esta estrategia, se puede adoptar una de las dos variantes de controlador siguientes:

- Controlador diseñado en el espacio articular

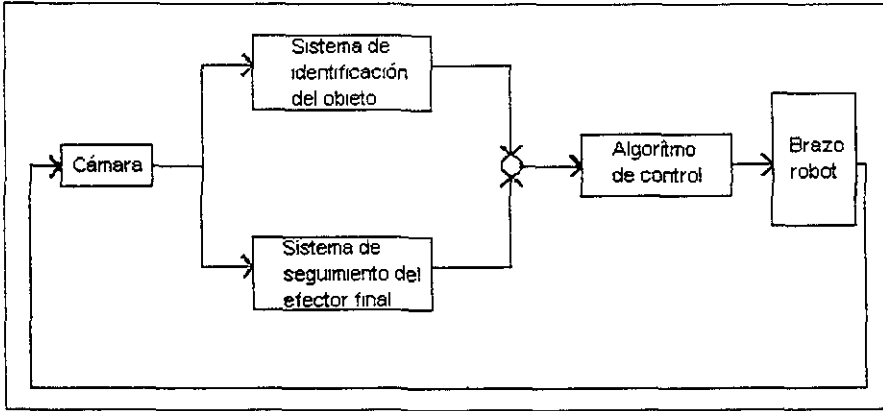


Figura 2.12: Control con retroalimentación visual

- Controlador diseñado en el espacio cartesiano.

Visión integrada a un controlador en el espacio articular.

La integración del sistema de visión a un controlador de espacio articular puede ofrecer un índice de comportamiento muy superior al obtenido con una estrategia *Look and Move*. Los errores causados por una mala calibración del sistema de visión son disminuidos al usar visión como sensor primario para localizar tanto el objeto como el efector final. Los errores causados por una mala estimación de los parámetros del modelo cinemático del robot son reducidos debido a que el modelo inverso es usado no solamente para evaluar los ángulos articulares deseados correspondientes a la posición del objeto, sino también para estimar los ángulos articulares durante el movimiento del efector final que corresponden a su posición medida por la cámara, tendiendo por lo tanto a cancelarse dichos errores.

La figura 2.13 muestra la estructura funcional de esta estrategia de control.

Aunque el uso de esta estrategia de control puede dar una significativa inmunidad a errores en la estimación de los parámetros del modelo cinemático y errores de calibración del sistema de visión, aun adolece de un número de desventajas, entre las cuales destacan las siguientes:

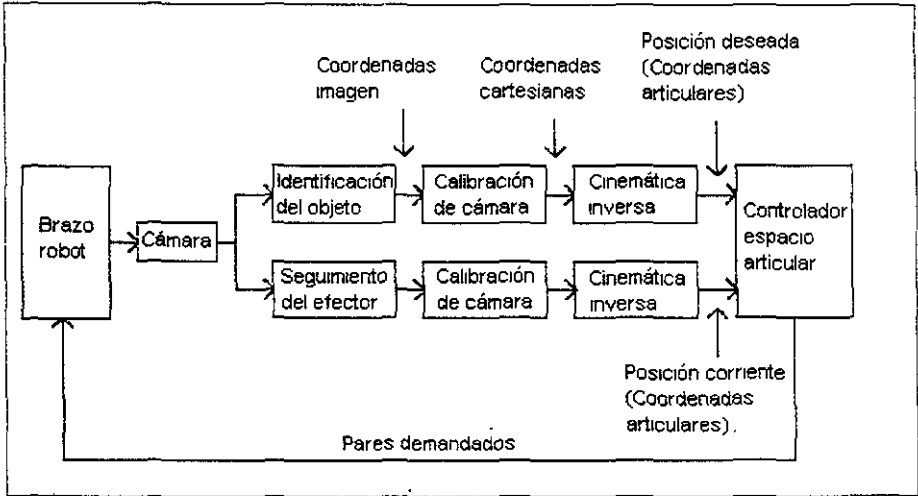


Figura 2.13: Estructura de datos de un controlador en el espacio articular.

- El movimiento del efector final del robot se especifica solamente en los puntos finales de las trayectorias, lo cual produce trayectorias que dependen sólo de los parámetros de los servomecanismos. Esto puede ser un problema en aquellas aplicaciones en las que se necesita sortear obstáculos.
- Para algunas estructuras mecánicas la solución cinemática inversa no se puede encontrar analíticamente.

Visión integrada a un controlador en el espacio cartesiano.

Un controlador de espacio cartesiano puede superar considerablemente las dificultades encontradas en el controlador de espacio articular

El error entre la posición corriente y la deseada sobre cierta trayectoria, es evaluado en el espacio cartesiano y no en el espacio de articulaciones, evitando así la inversión de modelos imprecisos, lo que conduce a una mayor rapidez y precisión. Los pares articulares requeridos para producir la trayectoria demandada son evaluados sobre la base de las posiciones deseada y corriente del efector final especificadas en coordenadas cartesianas. Esto permite tener el control sobre la trayectoria real del efector final.

Las consideraciones de fuerzas a ser aplicadas por el efector final, se hacen más fáciles cuando se emplean coordenadas cartesianas. El control de la tarea propiamente dicho se facilita con esta estrategia pues se mide directamente la variable a controlar (fuerza).

Aunque el sistema de control sigue necesitando estimar los ángulos articulares para hacer un estimado de la dinámica nominal del brazo, éstos no toman parte en la determinación de la precisión del seguimiento del sistema. Por lo tanto, se pueden usar sensores no muy precisos para evaluar la dinámica nominal del brazo, sin necesidad de evaluar el modelo cinemático inverso del robot. La figura 2.14 muestra la estructura funcional de esta estrategia de control y a continuación se describen sus ventajas y desventajas.

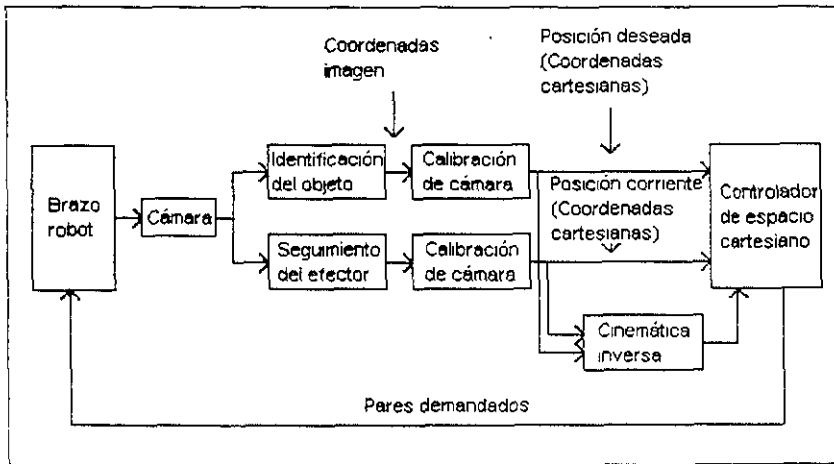


Figura 2.14: Estructura de datos de un controlador en el espacio cartesiano

Ventajas:

- Sencillez de procesamiento y por tanto rapidez.
- Calibración única fuera de línea (*off-line*).

- Una misma escena permite adquirir información sobre el entorno y sus variaciones, así como datos sobre la localización (absoluta o relativa) del robot, útiles en el control de su posición cartesiana y/o articular.

Desventajas:

- El robot puede obstruir el campo de visión útil en relación a la tarea por desarrollar.
- Los objetos pueden aparecer fuera de la zona central de la imagen sufriendo aberraciones y deformaciones.

Integración Robot-Visión con cámara móvil (*Eye-in-hand*). En este modo de integración la cámara es montada en el efector final del robot. La tarea de tomar una pieza se resume en mover el robot hasta que la cámara "vea" la pieza, localizada justo debajo del efector final. Este modo, no obstante de ofrecer ventajas potenciales, sufre de algunas desventajas:

- La cámara no da una vista completa del espacio de trabajo del robot, por lo que no puede usarse fácilmente para la tarea completa de localizar e identificar objetos, siendo necesaria la implementación de un algoritmo de búsqueda eficiente.
- El sistema de visión debe ser recalibrado en cada instante de muestreo, debido a que el referencial visión se encuentra en constante movimiento.
- Complejidad de cálculos.

2.4.2. Sustitución del *Teach Pendant*.

El *Teach-Pendant*, medio de comunicación H-M a través del cual se programa al robot Unimate S-103, realiza una comunicación bidireccional con la CPU (Z-80) del controlador del robot, usando código ASCII en un formato serie asincrónico vía la interfase estándar RS-232. El formato empleado en esta comunicación es el mostrado en la tabla 2.7.

Baud Rate	9600
Bits de datos	8
Paridad	Ninguna
Bits de Stop	1

Tabla 2.7: Formato de comunicación del Teach Pendant.

Para lograr la integración del robot con el sistema de VA, fue necesario sustituir el *teach-pendant* por una PC con el fin de permitir la consideración de información exteroceptiva en la realización de tareas con el robot *Unimate S-103*. Por tal motivo, se requirió conocer el protocolo de comunicación utilizado, así como la codificación de cada uno de los comandos intercambiados entre el *teach-pendant* y la CPU del controlador.

Al conectar la PC al puerto serie del robot y gracias a un programa de lectura e interpretación de datos, se pudo obtener información sobre el intercambio de información, la cual se describe a continuación:

Las 20 teclas multifuncionales que conforman el teclado del *teach-pendant* representan una matriz. Cada vez que se acciona una tecla, se selecciona el código ASCII asociado y se enviado al controlador.

Además de las 20 teclas frontales, el *teach-pendant* cuenta con dos teclas laterales que controlan la velocidad de movimiento y un botón de paro de emergencia.

Al emplear una PC en lugar del *teach-pendant*, se conservó el mismo formato de comunicación proporcionado por el fabricante. Esto trae consigo ciertos inconvenientes, ya que al no contemplar el control en tiempo real desde una PC huésped, se obliga a hacer uso de las funciones asignadas al *teach-pendant* para realizar cualquier tarea. Es decir, la realización de cualquier tarea tiene que hacerse obligatoriamente siguiendo el diagrama de flujo del protocolo C/ROS mostrado en la figura 2.5 y cada vez que la PC envía un comando, debe esperarse a que el controlador responda antes de mandar uno nuevo.

Acceso a los registros y modos de operación. Para lograr una comunicación con el controlador del robot *Unimate S-103* desde la PC, se necesitan dos cosas: por un lado, comunicar al controlador la información exteroceptiva que genera el sistema de visión y por el otro, conocer la información propioceptiva que proporciona el controlador.

El C/ROS proporciona tres registros, uno por cada articulación rotacional, que son usados tanto para lectura como para escritura y que son accedidos dentro del modo de operación *Teach*. Leyendo el contenido de dichos registros se obtiene la posición actual en coordenadas robot; escribiendo en ellos se solicita una consigna a los servomecanismos de posición

Para comunicar información exteroceptiva (en este caso las coordenadas articulares para alcanzar la posición y orientación del efector final deseado), se debe realizar la secuencia de operaciones mostradas en la figura 2.15. Por su parte, la secuencia de operaciones necesarias para la lectura de los sensores propioceptivos (variables articulares), se muestran en la figura 2.16.

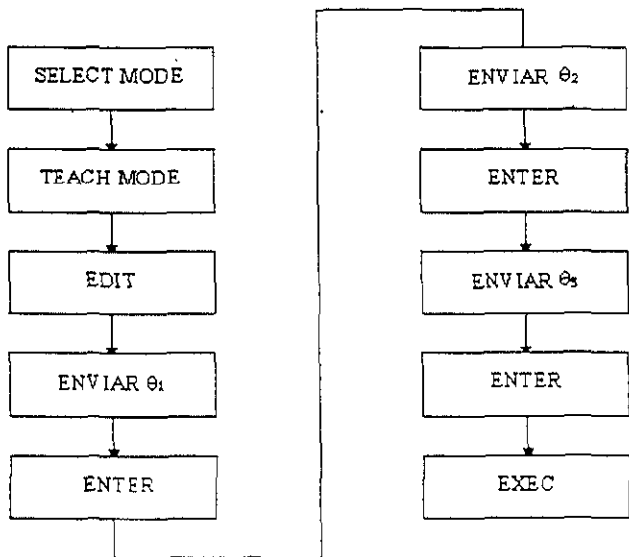


Figura 2.15: Comunicación de información exteroceptiva.

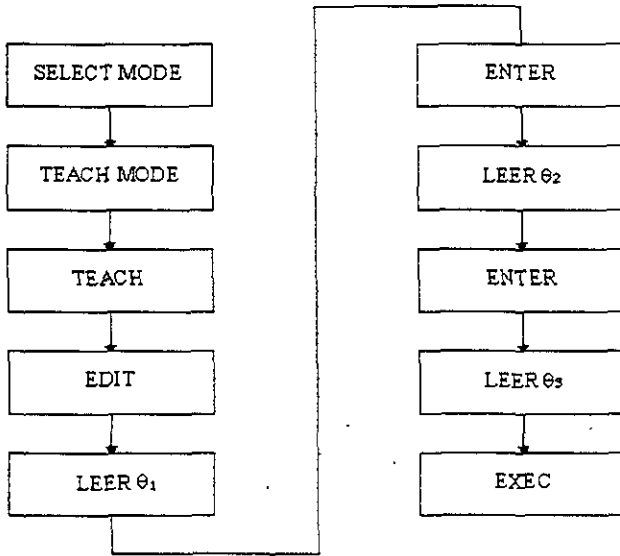


Figura 2.16: Lectura de información propioceptiva.

2.4.3. Calibración del Sistema Robot-Visión.

El objetivo de todo algoritmo de calibración de cámara es la obtención de los parámetros necesarios para la transformación de la posición y orientación en un espacio de coordenadas imagen R_i (bidimensional) al espacio robot o cartesiano R_r (tridimensional). Dadas las características del robot empleado analizadas en capítulos anteriores y dadas las características del puesto de trabajo con que se cuenta, en el cual todo está en un mismo plano horizontal R_m , se utilizó un algoritmo de calibración de cámara para la transformación de coordenadas del plano imagen al plano robot.

La cámara sólo puede ver la parte superior del robot, por lo tanto, en el proceso de calibración se usó la cubierta superior de la tercera articulación, cuya altura es proporcionada por el fabricante. Esto hace que la calibración quede establecida en un plano R_r' , localizado a una altura h_1 del plano robot (figura 2.17). La altura a la cual se encuentra la cámara sobre el plano de trabajo (h_c) también debe ser

conocida de antemano.

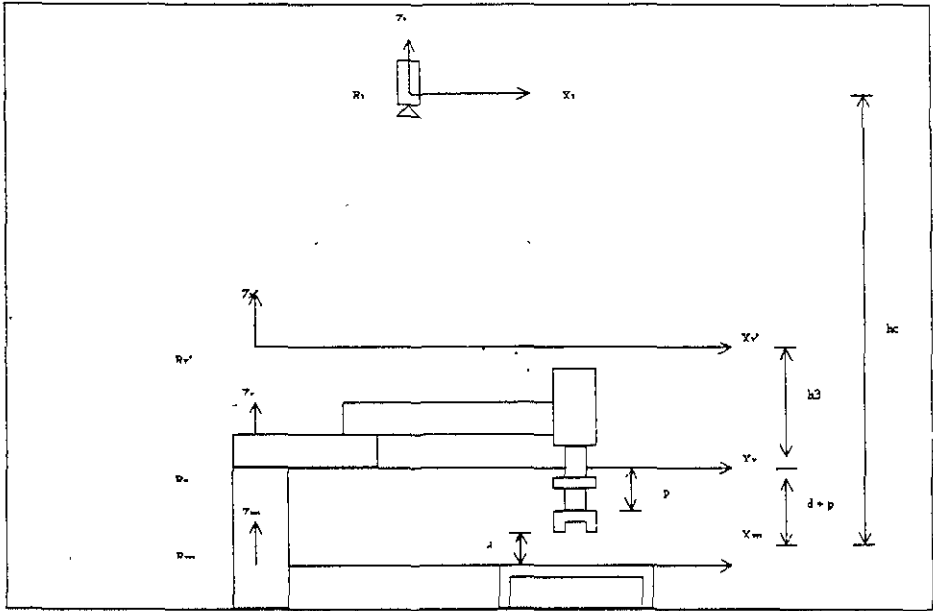


Figura 2.17: Plano de calibración de la cámara.

Proposición 2.1. . Sea $R_r(x_r, y_r, z_r)$ el referencial asociado al robot, cuyo origen se encuentra a la altura de la base de la pinza. Sea $R_m(x_m, y_m, z_m)$ el referencial asociado a la mesa de trabajo, cuyo origen se encuentra $(d + p)$ cm. abajo del referencial robot ($d =$ stroke, $p =$ longitud de la pinza) y sea $R_i(x_i, y_i, z_i)$ el referencial imagen cuyo origen está a h_c cm. sobre el plano de trabajo ($z_m = 0$). Entonces el referencial R_i difiere del referencial R_r por una traslación $T(x_{o_r}, y_{o_r})$, una rotación $R(\alpha, z_r)$ y un escalamiento $S(s_x, s_y)$ (figuras 2.17 y 2.18) según la transformación siguiente:

$$A_r^i = S(s_x, s_y)R(\alpha, z_r)T(x_{o_r}, y_{o_r}) \quad (2.24)$$

De esta manera la transformación de coordenadas imagen a coordenadas robot se realiza mediante la ecuación siguiente:

$$p_r = A_i^r p_i \tag{2.25}$$

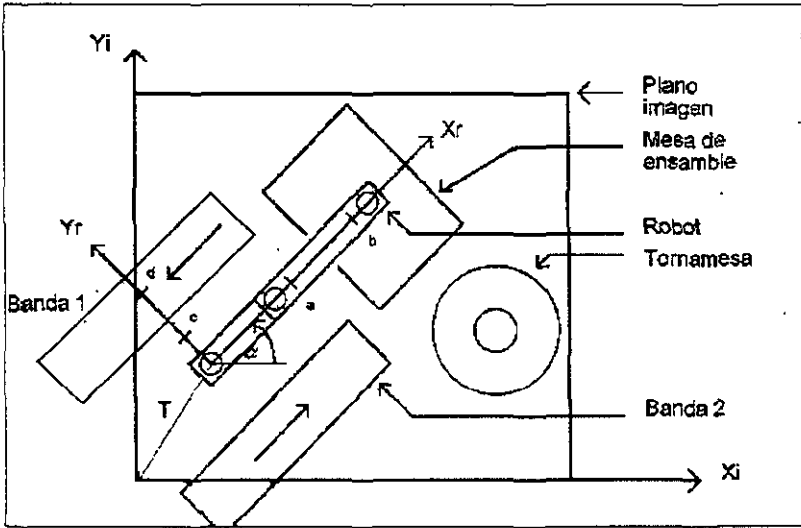


Figura 2.18: Procedimiento de calibración de cámara.

Para calcular los parámetros a utilizar en la transformación $(x_{o_r}, y_{o_r}, s_x, s_y, \alpha)$ se usa un procedimiento indirecto que se describe a continuación.

Procedimiento de calibración Para la calibración del sistema se establecen 4 puntos de referencia sobre los ejes coordenados del plano robot, dos pertenecientes al eje x (puntos a y b) y dos pertenecientes al eje y (puntos c y d) (Figura 2.18), en los cuales se posicionará el efector final del robot para ser observados por el sistema de visión. Al ser observados estos puntos por el sistema de visión, quedan definidos dos vectores por cada punto sobre el plano R'_r : un vector en coordenadas imagen v y un vector en coordenadas robot. Estos vectores constituyen los datos iniciales necesarios para la obtención de los parámetros de calibración del sistema

integrado. Las ecuaciones 2.26 a 2.29 muestran la estructura de estos vectores en coordenadas homogéneas.

$$\begin{aligned} a_r &= \begin{bmatrix} x'_{a_r} & y'_{a_r} & 0 & 1 \end{bmatrix}^T \\ a_i &= \begin{bmatrix} x'_{a_i} & y'_{a_i} & 0 & 1 \end{bmatrix}^T \end{aligned} \quad (2.26)$$

$$\begin{aligned} b_r &= \begin{bmatrix} x'_{b_r} & y'_{b_r} & 0 & 1 \end{bmatrix}^T \\ b_i &= \begin{bmatrix} x'_{b_i} & y'_{b_i} & 0 & 1 \end{bmatrix}^T \end{aligned} \quad (2.27)$$

$$\begin{aligned} c_r &= \begin{bmatrix} x'_{c_r} & y'_{c_r} & 0 & 1 \end{bmatrix}^T \\ c_i &= \begin{bmatrix} x'_{c_i} & y'_{c_i} & 0 & 1 \end{bmatrix}^T \end{aligned} \quad (2.28)$$

$$\begin{aligned} d_r &= \begin{bmatrix} x'_{d_r} & y'_{d_r} & 0 & 1 \end{bmatrix}^T \\ d_i &= \begin{bmatrix} x'_{d_i} & y'_{d_i} & 0 & 1 \end{bmatrix}^T \end{aligned} \quad (2.29)$$

donde el subíndice i indica coordenadas imagen, el subíndice r coordenadas robot y el superíndice " ' " indica referencia al plano R'_i .

Los ejes coordenados del plano robot, referidos al plano imagen, vendrán dados por las ecuaciones 2.30 y 2.31.

$$y'_{z_{r1}} = m_{z1}(x'_i - x'_{a1}) + y'_{a1} \quad (2.30)$$

$$y'_{y_{r1}} = m_{y1}(x'_i - x'_{c1}) + y'_{c1} \quad (2.31)$$

donde:

$m_{z1} = \frac{y'_{b1} - y'_{a1}}{x'_{b1} - x'_{a1}}$ es la pendiente del eje x del robot en el plano imagen

$m_{y1} = \frac{y'_{d1} - y'_{c1}}{x'_{d1} - x'_{c1}}$ es la pendiente del eje y del robot en el plano imagen.

$y'_{z_{r1}}$ es la coordenada y del eje x del robot en el plano imagen.

$y'_{o_{r1}}$, es la coordenada y del eje y del robot en el plano imagen.

Igualando las ecuaciones 2.30 y 2.31 se obtiene la coordenada x del origen del referencial robot en el plano imagen, la que al sustituirse en cualquiera de las ecuaciones 2.30 y 2.31, resulta en la coordenada y de este referencial. Los resultados de estas operaciones están dados por las ecuaciones 2.32 y 2.33 respectivamente.

$$x_{o_{r1}} = \frac{1}{m_{x1} - m_{y1}} (m_{x1}x'_{a1} - m_{y1}x'_{c1} + y'_{c1} - y'_{a1}) \quad (2.32)$$

$$y_{o_{r1}} = m_{x1}(x_{o_{r1}} - x'_{a1}) + y'_{a1} \quad (2.33)$$

donde:

$x_{o_{r1}}$ es la coordenada x del origen del referencial robot en el plano imagen.

$y_{o_{r1}}$ es la coordenada y del origen del referencial robot en el plano imagen

El ángulo de rotación α del referencial robot respecto al plano imagen, estará dado por la ecuación 2.34.

$$\alpha = \text{tg}^{-1}m_{x1} \quad (2.34)$$

Como se mencionó al inicio de esta sección (Proposición 2.1) y como aparece en la figura 2.18, se observa que, para la transformación de coordenadas imagen a coordenadas cartesianas sobre el plano R'_r , son necesarias tres transformaciones:

- Una traslación $-T(x_{o_{r1}}, y_{o_{r1}})$ sobre el plano xy .
- Una rotación $R(\alpha, z_1)$ en $-\alpha$ grados alrededor del eje z .
- Un escalamiento $S(s_x, s_y)$ en x y y , que convierta unidades imagen a unidades robot (dimensiones del pixel).

Expresadas en forma matricial estas transformaciones quedan resumidas por las ecuaciones 2.35 a la 2.37.

$$T = \begin{bmatrix} 1 & 0 & 0 & -x_{o_{r1}} \\ 0 & 1 & 0 & -y_{o_{r1}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.35)$$

$$R_{z-\alpha} = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.36)$$

$$S = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.37)$$

Sustituyendo las ecuaciones 2.35 a la 2.37 en la ecuación 2.24, se obtiene la matriz de transformación total A'_r quedando definida de la siguiente manera:

$$A'_r = \begin{bmatrix} s_x \cos(\alpha) & s_x \sin(\alpha) & 0 & -s_x \left[x'_{o_r} \cos(\alpha) + y'_{o_r} \sin(\alpha) \right] \\ -s_y \sin(\alpha) & s_y \cos(\alpha) & 0 & +s_y \left[x_{o_r} \sin(\alpha) - y'_{o_r} \cos(\alpha) \right] \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.38)$$

Para la determinación de los factores de escala s_x y s_y , se usan los vectores obtenidos inicialmente en los cuatro puntos del espacio. Tomando en cuenta que para todo punto p la transformación de coordenadas imagen a coordenadas cartesianas viene dada por la ecuación 2.25, entonces, resolviendo 2.24 para s_x y s_y se obtienen las ecuaciones 2.39 y 2.40 lo que permiten el cálculo de los factores de escala. En este sentido, es recomendable promediar los factores de escala, obtenidos al evaluar las ecuaciones 2.39 y 2.40 en los cuatro puntos conocidos desde el inicio del proceso de calibración, para tener mayor precisión en los resultados.

$$s_x = \frac{x_p}{x'_{p_i} \cos(\alpha) + y'_{p_i} \sin(\alpha) - x_{o_r} \cos(\alpha) + y_{o_r} \sin(\alpha)} \quad (2.39)$$

$$s_y = \frac{y_p}{-x'_{p_i} \sin(\alpha) + y'_{p_i} \cos(\alpha) + x_{o_r} \sin(\alpha) - y_{o_r} \cos(\alpha)} \quad (2.40)$$

Con la obtención de los factores de escala s_x y s_y se completa el vector de parámetros de calibración del sistema, el cual queda conformado como lo muestra la ecuación 2.41.

$$\text{Parámetros} = \begin{bmatrix} x_{0r1} \\ y_{0r1} \\ s_x \\ s_y \\ \alpha \end{bmatrix} \quad (2.41)$$

La matriz de transformación A_r^1 , dada por la ecuación 2.38, transforma coordenadas imagen a coordenadas robot (cartesianas) en el plano R_r' .

Para lograr la transformación al plano R_m , donde realmente estarán localizados los objetos a manipular, es necesario incluir la transformación mostrada en la figura 2.19.

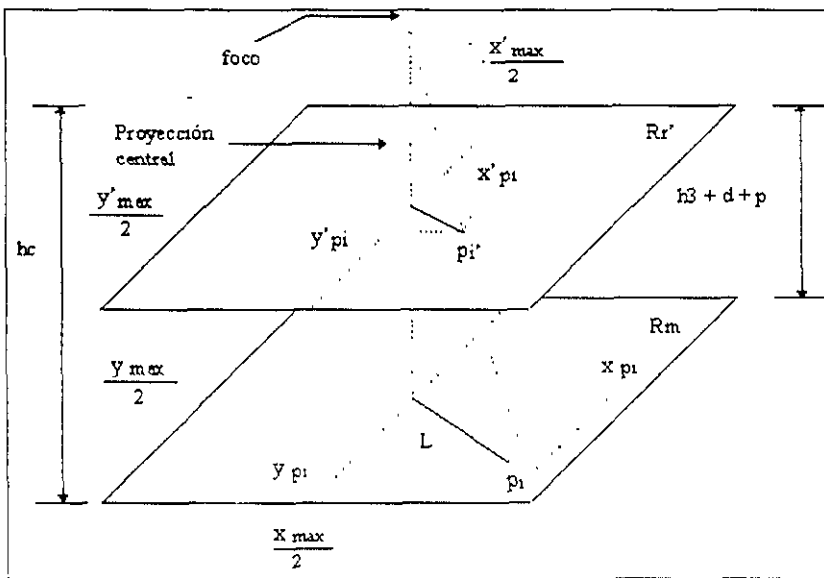


Figura 2.19: Transformación del plano R_r' al plano R_m .

Cualquier punto p_1 , que se encuentre sobre el plano R_m , será visto por la cámara como el punto p_1' en el plano de calibración R_r' , localizado a una distancia l del eje de proyección. Esta distancia está dada por la ecuación 2.42

$$l = \sqrt{\left(x'_{p_i} - \frac{x_{\max}}{2}\right)^2 + \left(y'_{p_i} - \frac{y_{\max}}{2}\right)^2} \quad (2.42)$$

Por relación de triángulos, la distancia del punto p_i al eje de proyección, vendrá dada por la ecuación 2.43.

$$L = \frac{h_c}{h_c - (h_3 + d + p)} l$$

$$L = \frac{h_c}{h_c - (h_3 + d + p)} \sqrt{\left(x'_{p_i} - \frac{x_{\max}}{2}\right)^2 + \left(y'_{p_i} - \frac{y_{\max}}{2}\right)^2} \quad (2.43)$$

Finalmente, las coordenadas de un punto cualquiera sobre el plano R_m , vendrán dadas por las ecuaciones 2.44 y 2.45.

$$x_{p_i} = \frac{h_c}{h_c - (h_3 + d + p)} \left(x'_{p_i} - \frac{x_{\max}}{2}\right) + \frac{X_{\max}}{2} \quad (2.44)$$

$$y_{p_i} = \frac{h_c}{h_c - (h_3 + d + p)} \left(y'_{p_i} - \frac{y_{\max}}{2}\right) + \frac{Y_{\max}}{2} \quad (2.45)$$

En resumen, para la transformación de coordenadas imagen a coordenadas cartesianas, se deben seguir los siguientes pasos:

1. Partiendo de la localización del punto en coordenadas imagen, la cual corresponde a coordenadas del plano R'_r , se debe obtener la proyección del punto sobre el plano R_m a través de las ecuaciones 2.44 y 2.45.
2. Al punto encontrado en el paso anterior, aplicar la matriz de transformación total dada por la ecuación 2.38

3.Introducción a las Redes Neuronales

3.1.Introducción

El cerebro humano está compuesto por aproximadamente 10^{11} neuronas que funcionan a través de impulsos eléctricos a una velocidad de alrededor de $20m/s$; la principal característica de estas neuronas es su poder de interconexión o de comunicación, se calcula que existen alrededor de 10^{15} conexiones, debido a esto es que el cerebro tiene la capacidad de tomar decisiones rápidas.

Cada una de las neuronas, que no son mas que células vivas, está compuesta por un núcleo, un soma, un axón y diversas dendritas, y la forma en cómo funcionan es la siguiente: se recibe la información por medio de las dendritas y se procesa en el núcleo para ser enviada a otras neuronas mediante su axón. Cuando la información proveniente de otras neuronas llega al fin del axón, se transmite a las dendritas de la siguiente neurona a través de una sinapsis, ésta puede ser del tipo positivo (excitatorio) o negativo (inhibitorio)

A diferencia de las neuronas biológicas, los microcomputadores emplean componentes que funcionan a alta velocidad con tiempos de respuesta del orden de los nanosegundos y aunque la comunicación es buena, su poder de interconexión no se compara con el de las neuronas biológicas, debido a que emplean pocos procesadores. Al crearse las Redes Neuronales Artificiales se trató de tomar la principal característica de cada uno de estos casos, es decir, la velocidad de los elementos del microcomputador y la interconexión masiva de las neuronas.

Definición 3.1. *Las RN Artificiales se pueden definir como un sistema computacional de procesamiento en paralelo, cuyos elementos se encuentran altamente interconectados entre si logrando de esta forma proporcionar una respuesta a entradas externas.*

La estructura general de una RN esta determinada por la ecuación.

$$y = f_i(W \cdot x + V \cdot y + \theta) \tag{3.1}$$

donde y es el vector de salidas, f_i es la función de activación, W es la matriz de pesos ponderando las entradas, V es la matriz de retroalimentación de las salidas

, x es el vector de entrada y θ es el vector de polarización. La siguiente figura nos muestra el diagrama de bloques de una Red Neuronal.

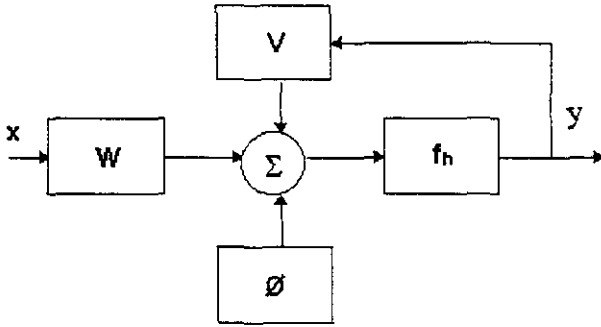


Figura 3.1: Estructura de una Red Neuronal

La forma en que la RN se entrena y aprende es un método muy sencillo que se describe a continuación: la RN está compuesta por Nodos o Neuronas que se encuentran distribuidos en diferentes capas, ya sean de entrada, ocultas o de salida, dichas neuronas están conectadas entre sí y a cada conexión se le asigna un peso que va a estar modificando su valor continuamente de tal forma que se vayan acercando cada vez más los valores de las entradas con los de las salidas deseadas que previamente se debieron haber ingresado. Cuando los valores de los pesos permanezcan estables podremos pensar que el proceso ha terminado y que la RN ha completado su aprendizaje.

3.2. Características

A continuación se describen las principales características de un RN:

a). Topología. Es decir, la estructura de la RN: el número de capas que la componen, el número de neuronas por capa si existe retroalimentación entre sus capas o no ($V \neq 0$) y el grado y tipo de conectividad que existe entre las neuronas.

b). Función de Activación. Así como es necesario una regla que combine las entradas de las neurona con los pesos de las conexiones, también se requiere de una regla que combine las entradas con el estado actual de la neurona para producir

un nuevo estado de activación. El objetivo de la función de activación es producir un nuevo estado de activación a partir del estado actual de la neurona y de la combinación de las entradas con los pesos de las conexiones.

c). Método de Aprendizaje. Se entiende por aprendizaje el ajuste de los pesos (W y V), lo que permiten obtener el comportamiento deseado de la RN. De acuerdo al tipo de aprendizaje que emplean las RN's, podemos mencionar los siguientes:

- Redes neuronales con aprendizaje supervisado.
- Redes neuronales con aprendizaje no supervisado.

La principal diferencia entre ambos tipos, estriba en la existencia o no de un agente externo (supervisor) que controle el proceso de aprendizaje de la red.

d). Tipo de asociación entre la información de entrada y de salida. De acuerdo a la información con la que se entrenó a la RN y a los valores establecidos en los pesos, es capaz de establecer una relación entre la entrada y la salida, lo cual nos es de gran utilidad al ingresar datos diferentes a los empleados en el aprendizaje, siempre y cuando sean tomados de la misma forma.

3.3. Aplicaciones

Las Redes Neuronales inicialmente fueron utilizadas en aplicaciones tales como reconocimiento de patrones, aproximadores de funciones, controladores, predictores y en la actualidad tienen muchas más aplicaciones en diferentes áreas como por ejemplo:

Biología.

Aprendizaje del cerebro y otros sistemas

Obtención de modelos de la retina

Empresas.

Evaluación de probabilidad de formaciones geológicas y petrolíferas.

Identificación de candidatos para posiciones específicas

Explotación de bases de datos

Optimización de plazas y horarios en líneas de vuelo.

Reconocimiento de caracteres escritos

Medio ambiente

Análisis de tendencias y patrones.
Previsión del tiempo.
Finanzas.
Previsión sobre la evolución de los precios.
Valoración del riesgo de los créditos.
Identificación de falsificaciones.
Interpretación de firmas.
Manufactura.
Robots y sistemas de control (visión artificial y sensores de presión, temperatura, gas, etc.).
Control de producción en líneas de proceso.
Inspección de la calidad.
Medicina.
Análisis del habla para mejorar la audición de sordos profundos.
Diagnóstico y tratamiento a partir de síntomas y/o de datos analíticos (electrocardiograma, encefalograma, análisis sanguíneo, etc.).
Monitorización en cirugía. Predicción de reacciones adversas a los medicamentos.
Lectores de rayos X.
Entendimiento de las causas de los ataques epilépticos.
Aplicaciones Militares.
Clasificación de las señales de radar
Optimización del uso de recursos escasos.
Reconocimiento y seguimiento en el tiro al blanco.
Etc.

3.4. Clasificación

Existen muchas formas de clasificar a la Redes Neuronales partiendo siempre de algo de los elementos que las caracterizan como su funcionamiento, su topología, su forma de aprendizaje, etc. A continuación mencionaremos algunas de las clasificaciones más importantes:

3.4.1. Topología.

Redes Monocapa. Las Redes Monocapa sólo cuentan con una capa en la red y las conexiones entre las neuronas se establecen de forma lateral o también pueden

existir conexiones autorrecurrentes (la salida de una neurona se conectada a su propia entrada).

Redes Multicapa. Las Redes Multicapa son aquellas que disponen de diversas neuronas organizadas en varias capas.

3.4.2. Conectividad.

Redes Estáticas. Las RN's estáticas poseen únicamente conexiones entre capas hacia adelante (*feedforward*). Por lo regular las neuronas de determinada capa reciben señales de entrada de la capa anterior y de la misma forma emiten señales de salida para la capa de neuronas posterior. En estas Redes la matriz de ponderación de las salidas es igual a cero ($V = 0$).

Redes Dinámicas. Las RN's dinámicas poseen, además, conexiones hacia atrás (*feedback*). Este proceso es diferente al anterior puesto que aquí la salida de una capa de neuronas se conecta a la entrada de la capa anterior o a una neurona de la misma capa. Aquí la matriz de ponderación de las salidas es diferente de cero ($V \neq 0$).

3.4.3. Aprendizaje.

Supervisado. Aquí el proceso de aprendizaje se realiza mediante un entrenamiento controlado por un agente externo (supervisor), que determina la respuesta que deberá generar la red a partir de de una entrada determinada.

Existen tres formas de llevar a cabo el aprendizaje supervisado, lo que da lugar a los siguientes tipos:

Aprendizaje por corrección de error. Consiste en ajustar los pesos de las conexiones de la red en función de la diferencia entre los valores deseados y los obtenidos en la salida de la red: es decir, en función del error cometido en la salida.

Aprendizaje por refuerzo. Se trata de un aprendizaje más lento que el anterior, que se basa en la idea de no disponer de un ejemplo completo del comportamiento deseado: es decir, de no indicar durante el entrenamiento exactamente

la salida que se desea que proporcione la red ante una determinada entrada. En este aprendizaje la función del supervisor se reduce a indicar el éxito o fracaso mediante una señal de refuerzo (éxito = +1 o fracaso = -1) y en función de esto se ajustan los pesos basándose en un mecanismo de probabilidades.

Aprendizaje estocástico. Este tipo de aprendizaje consiste básicamente en realizar cambios aleatorios en los valores de los pesos de las conexiones de la red y evaluar su efecto a partir del objetivo deseado y de distribuciones de probabilidad.

No Supervisado. No requiere de influencia externa para ajustar los pesos de las conexiones entre sus neuronas. La red no recibe ninguna información por parte del entorno que le indique si la salida generada en respuesta a una determinada entrada es o no correcta; por ello, suele decirse que estas redes son capaces de autoorganizarse.

Dentro de los algoritmos de aprendizaje no supervisado se encuentran los siguientes:

Aprendizaje Hebbiano. En este algoritmo la modificación de los pesos se realiza en función de los estados (salidas) de las neuronas, obtenidos tras la presentación de cierto estímulo (información de entrada a la red), sin tener en cuenta si se deseaba obtener o no esos estados de activación.

Aprendizaje competitivo y cooperativo. Se dice que las neuronas compiten y/o cooperan unas con otras con el fin de llevar a cabo una tarea dada. Con este tipo de aprendizaje se pretende que cuando se presente a la red cierta información de entrada sólo una de las neuronas de salida de la red, o una por cierto grupo de neuronas, se active (alcance su valor de respuesta máximo). Por tanto, las neuronas compiten por activarse, quedando finalmente una, o una por grupo, como neurona vencedora, mientras que el resto son forzadas a sus valores de respuesta mínimos.

3.4.4.Regla de Aprendizaje.

On line, cuando la red puede aprender durante su funcionamiento habitual y *off line* cuando se necesita la desconexión de la red para realizar su aprendizaje

3.4.5. Asociación de información.

Redes Heteroasociativas. Pueden relacionar la información de entrada con diferentes informaciones de salida. Se necesitan al menos 2 capas en la red, una que se encarga de recibir y retener la información de entrada y otra para mantener la salida con la información asociada.

Redes Autoasociativas. Pueden relacionar la información de entrada con la de la salida más parecida almacenada por la red. En este caso la red puede funcionar con una sola capa de neuronas la cual retiene la información de entrada y termina representando la información autoasociada. Si se requiere almacenar la información de entrada y de salida, se deben añadir más capas.

Además de estas clasificaciones podemos mencionar que las redes neuronales se dividen en redes que emplean funciones lineales y redes que emplean funciones de activación no lineales. Las no lineales pueden generar comportamientos más complejos que las lineales.

3.5. Funciones de Activación

Dentro de las funciones de transferencia más empleadas en la actualidad, podemos mencionar las siguientes Lineal, Lineal con Saturación, Escalón, Sigmoidal y Gaussiana. No obstante pueden utilizarse otras, solamente se requiere observar cual es su comportamiento y saber si se obtienen buenos resultados. A continuación se describen cada una de las funciones mencionadas.

3.5.1. Función de Activación Lineal.

Estudios sobre este tipo de función han llegado a un fenómeno que se conoce como Restricción de Predictibilidad Lineal, es decir, existen problemas que no son enteramente reducibles mediante métodos lineales. Una red neuronal compuesta por neuronas lineales, puede resolver a lo más problemas que son linealmente separables. Puesto que la mayoría de los problemas más interesantes involucran no linealidades, estos no pueden ser resueltos por redes con funciones de activación lineales, por lo que se prefiere el uso de funciones no lineales. Una función de activación lineal es aquella cuya salida tiene una pendiente constante a (Figura 3.2).

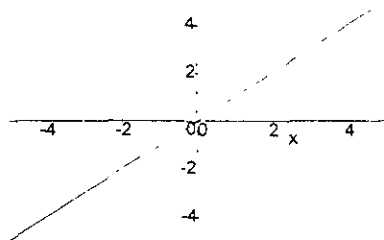


Figura 3.2. Funcion Lineal

$$f(x) = ax + b \quad (3.2)$$

3.5.2. Función Lineal con Saturación.

Es aquella cuya salida es una constante a múltiplo de la entrada sobre algún rango, posiblemente movida hacia la izquierda o derecha por una constante b ; fuera del rango, la función toma un valor constante. La ganancia corresponde a la pendiente de la porción lineal de la curva. Debido a estas características, las RN's con este tipo de funciones experimenta un comportamiento ligeramente mejor comparadas con las redes compuestas de funciones de activación completamente lineales. En los inicios del perceptrón se empleó este tipo de funciones; sin embargo, se ha comprobado que presentan severas limitaciones en cuanto a lo que pueden aprender, por lo que actualmente su uso se ha reducido (Figura 3.3).

$$f(x) = \left\{ \begin{array}{l} ax + b, x_1 < x < x_2 \\ c, x_2 \leq x \\ d, x \leq x_1 \end{array} \right\} \quad (3.3)$$

3.5.3. Función Escalón.

Es aquella cuya salida esta limitada a dos posibles valores. Para una entrada por debajo de un valor de umbral, la salida es siempre baja, mientras que para una entrada sobre el valor de umbral, la salida es alta.

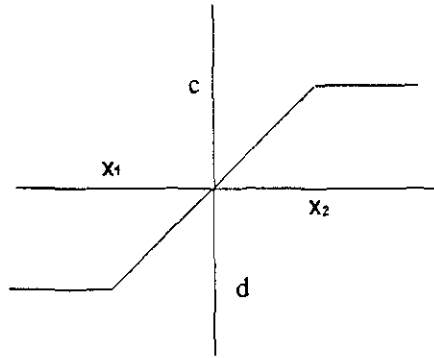


Figura 3.3: Función Lineal con Saturación

Debido a la discontinuidad de esta función es altamente no lineal. Redes con este tipo de función, presentan mejoras en su comportamiento comparadas con las anteriores. Estas funciones fueron usadas en los estudios iniciales de las Redes Neuronales Artificiales, como las construidas por McCulloch & Pitts en 1943, y son también la clase más frecuentes usada en las redes de Hopfield (Figura 3.4).

$$f(x) = \begin{cases} c, & x \geq b \\ d, & x < b \end{cases} \quad (3.4)$$

3.5.4. Función Sigmoidal.

También se le conoce como forma de S, semilineal o función *squashing*; es aquella cuya salida es función continua y monótona creciente de la entrada. Tanto la función como su derivada son continuas en todo punto. Se observa que la función se aproxima asintóticamente por un lado a los valores altos y por el otro a valores bajos. La ganancia es proporcional a la derivada de la función.

Redes con este tipo de función presentan muy buen desempeño, en particular, pueden ser usadas por el algoritmo de *backpropagation* pues se puede hacer la optimización de la función del error de aproximación con respecto a los pesos. Nótese que en altas ganancias ($G \gg 1$), la función llega a ser casi igual a un escalón, mientras que para ganancias bajas ($G \ll 1$ pero $G > 0$), se aproxima a una función lineal. La elección de $G = 1$ da buenos resultados (Figura 3.5).

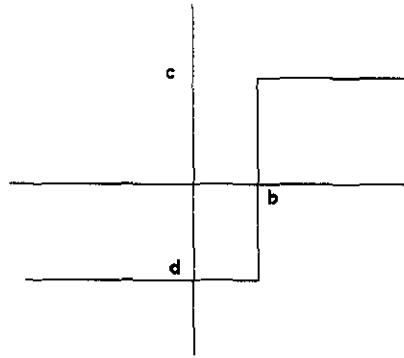


Figura 3.4: Función Escalon

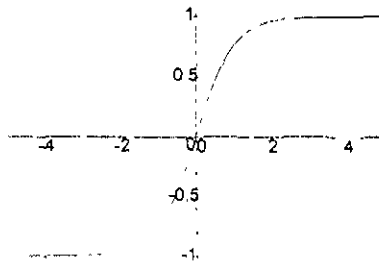


Figura 3.5: Función Sigmoide

$$f(x) = \tanh(Gx) + b \tag{3.5}$$

3.5.5. Función Gaussiana.

También se le conoce como Campana. No es una función monótona, aunque es continua y continua diferenciable. Se puede ver como la unión de dos sigmoideas una creciendo y la otra decreciendo. Su ganancia es proporcional a la desviación estándar de la campana. Esta función permite un buen incremento en la velocidad

de aprendizaje de la RN (Figura 3.6) y debido a su complejidad es adecuada para realizar mapeos no lineales.

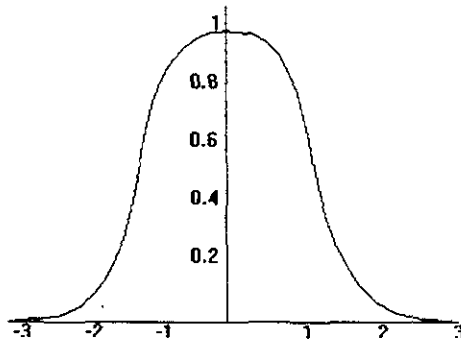


Figura 3.6: Función Gaussiana

$$f(x) = e^{-Gx} \tag{3.6}$$

3.6. Propiedades

A continuación se mencionan las diferentes propiedades ligadas a una RN.

3.6.1. Aprendizaje adaptable.

Es la capacidad que tiene la RN de llevar a cabo alguna tarea determinada a partir de un entrenamiento con ejemplos ilustrativos o experiencias iniciales, es decir, no es necesario elaborar modelos ni especificar funciones para que puedan trabajar correctamente. Una Red Neuronal no necesita un algoritmo para resolver un problema ya que ella puede generar su propia distribución de los pesos mediante el aprendizaje

3.6.2. Sistemas dinámicos autoadaptables.

Las RN's son sistemas dinámicos autoadaptables debido a que sus elementos procesales (neuronas), que componen el sistema, tienen la capacidad de autoajus-

tarse; son dinámicos, pues son capaces de estar constantemente cambiando para adaptarse a las nuevas condiciones.

3.6.3. Autoorganización.

Una Red Neuronal puede crear su propia representación de la información que se le suministra en una etapa de aprendizaje.

3.6.4. Generalización.

Una RN tiene la facultad de responder apropiadamente cuando se le presentan datos o situaciones a las que no había sido expuesta durante el aprendizaje; la RN puede generalizar la entrada para obtener una respuesta. Esta propiedad es muy importante cuando se tienen que solucionar problemas en los cuales la información de entrada es poco clara o está especificada de forma incompleta.

3.6.5. Tolerancia a fallos.

En relación a este punto existen dos aspectos importantes con las RN's: primero, pueden aprender a reconocer patrones con ruido, distorsionados o incompletos, ésta es una tolerancia a fallos de la información; segundo, pueden seguir realizando su función (con cierta degradación) aunque parte de la red no funcione adecuadamente.

La razón por la que las redes neuronales son tolerantes a los fallos es que tiene su información distribuida en las conexiones entre neuronas. existiendo cierto grado de redundancia en este tipo de almacenamiento.

Las Redes Neuronales son los primeros métodos computacionales con la capacidad inherente de tolerancia a fallos, a diferencia de los sistemas computacionales tradicionales, los cuales pierden su funcionalidad en cuanto sufren un pequeño error de memoria.

3.6.6. Operación en tiempo real.

Una de las prioridades de la mayoría de las áreas de aplicación, es la necesidad de realizar procesos con una gran cantidad de datos en forma muy rápida. Las Redes Neuronales se adaptan bien a esto debido a su implementación paralela. Para que la mayoría de la redes puedan operar en un entorno de tiempo real, la necesidad

de cambio en los pesos de las conexiones o entrenamiento es mínima. Por tanto, de todos los métodos posibles, las Redes Neuronales son la mejor alternativa para reconocimiento y clasificación de patrones en tiempo real.

3.7. Perceptron

El Perceptrón generó gran interés cuando inicialmente fue desarrollado debido a su capacidad de aprender a reconocer patrones sencillos. Está compuesto por una capa de entrada con varias neuronas y una capa de salida con una sola neurona (figura 3.7). El Perceptrón forma dos regiones de decisión separadas por un hiperplano o una línea recta en dos dimensiones y es capaz de decidir cuando una entrada pertenece a una de dos clases (A o B); la regla de decisión manda un +1 si pertenece a la clase A o un -1 si pertenece a la clase B.

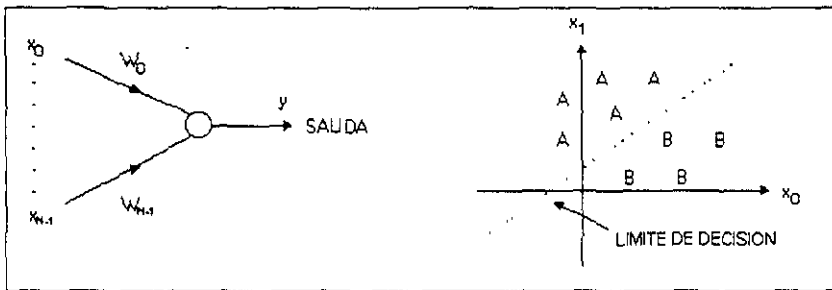


Figura 3.7 Perceptron Monocapa

El procedimiento de convergencia del perceptrón se explica a continuación:

1. Primero se inicializan los pesos (W_i) y el umbral (θ) con valores aleatorios pequeños que sean diferentes de cero.
2. Se presentan las entradas (X_0, X_1, \dots, X_{v-1}) y la salida deseada $d(t)$.
3. Se calcula la salida actual mediante la siguiente fórmula:

$$y(t) = f_h \left(\sum_{i=0}^{v-1} W_i(t) x_i(t) - \theta \right) \quad (3.7)$$

donde f_h es la función de transferencia escalón.

4. Sólo cuando existe un error, los pesos son adaptados por medio de la siguiente fórmula:

$$W_i(t+1) = W_i(t) + \eta[d(t) - y(t)]x_i(t) \quad (3.8)$$

donde $0 \leq i \leq N - 1$

En esta ecuación η representa un factor de ganancia positiva y $d(t)$ es la salida deseada correcta para la actual entrada. Es importante aclarar que los pesos sólo son modificados si la red no hace la decisión correcta, por lo tanto, estamos hablando de una red con aprendizaje supervisado.

5. Si error cuadrático medio (SSE) es menor o igual al nivel máximo tolerado, termina el proceso de entrenamiento, de lo contrario, se repite el procedimiento desde el paso 2.

Ahora hablaremos del Perceptrón Multicapas, que por cierto, pertenece al tipo de redes neuronales con conexiones hacia adelante y el cual cuenta con una o varias capas de neuronas adicionales entre la capa de entrada y la de salida. Esta(s) capa(s) adicional(es) contiene(n) neuronas ocultas que no están directamente conectadas a la entrada o la salida. El Perceptrón multicapas supera muchas de las limitaciones del Perceptrón monocapa descrito anteriormente, por ejemplo, esta red permite establecer regiones de decisión mucho más complejas. La figura 3.8 nos permite ver con mayor claridad las ventajas del perceptrón conforme se le van aumentando capas.

Como podemos observar, no se requieren más de tres capas en una red de tipo Perceptrón, pues como se ha visto, con ese número de niveles se pueden generar regiones de decisión arbitrariamente complejas.

3.8.BackPropagation

BackPropagation es un procedimiento de aprendizaje obtenido a raíz de las investigaciones realizadas en el Perceptron y mediante el cual se puede lograr obtener la relación que existe entre las entradas de la red y la(s) salida(s), y puede ser utilizado en redes que cuenten con dos o mas capas de neuronas. Este método es


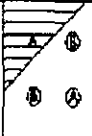







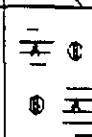


Estructura	Regiones de decisión	Problema de la XOR	Clases con regiones mezcladas	Formas de regiones más generales
1 CAPA 	Medio plano limitado por un hiperplano			
2 CAPAS 	Regiones Cerradas o Convexas			
3 CAPAS 	Arbitraria complejidad limitada por el número de neuronas			

Figura 3.8: Distintas formas de las regiones generadas por un perceptrón multi-nivel

conocido como *backpropagation* (propagación del error hacia atrás o retropropagación).

Una característica importante de este algoritmo es que es capaz de organizar la capa intermedia de las neuronas de tal forma que se pueda conseguir cualquier correspondencia entre la entrada y la salida.

El procedimiento que sigue el mecanismo de aprendizaje *backpropagation* para conseguir establecer una relación entre una serie de patrones con sus salidas correspondientes, consiste en alimentar a la RN con un conjunto de pares entrada-salida deseada previamente seleccionados para la etapa de entrenamiento. Se compara la salida generada por la red con la deseada y se calcula el valor del error en base al resultado obtenido, posteriormente este error se transmite hacia atrás y se estima la contribución al error total de cada una de las neuronas de la capa previa, es decir, se retropropaga el error de salida a la capa oculta previa. Se sigue el mismo proceso con cada una de las capas que constituyen la red y siempre hacia atrás con la finalidad de estimar la contribución al error total de todas y cada una de las neuronas de la red. Enseguida, se ajustan los pesos proporcionalmente al error

producido por la neurona que alimentan y se repite la operacion de tal forma que poco a poco el error vaya disminuyendo y la salida de la red se vaya acercando cada vez más a la salida deseada.

Una vez que el error es mínimo, podemos pensar que la red *backpropagation* ha aprendido la asociación entre los datos de entrada-salida deseados usados en la etapa de entrenamiento y está lista para procesar datos diferentes a los de entrenamiento (siempre y cuando sean obtenidos basándose en la misma regla que los de aprendizaje). Veremos que el perceptrón multicapas podrá aplicar la regla aprendida anteriormente con datos desconocidos y el error que resulte será despreciable.

4. Aplicación

En el presente capítulo se explica el procedimiento llevado a cabo para la obtención de la RN. En primer lugar, se describe la problemática a resolver y la función que tendrá la RN dentro del lazo de control del sistema Robot-Visión. Posteriormente, se seleccionan los elementos que conformarán la estructura de la RN de acuerdo a cada una de sus características. En la fase de entrenamiento, se plantean dos estrategias a seguir: en la primera de ellas es necesario la elaboración de un programa en *MatLab*[®] para simular el sistema Robot-Visión a través de un modelo matemático, lo que permite experimentar con la estructura de la RN hasta llegar a la configuración que entregue los resultados deseados; en la segunda estrategia, el entrenamiento se lleva a cabo con datos obtenidos directamente del robot y del sistema de visión artificial.

4.1. Análisis y acondicionamiento del problema

4.1.1. Sistema Robot - Visión.

El uso de VA en las tareas de ensamble del robot *Unimate S-103*, introduce la necesidad de contar con lo siguiente:

- Un algoritmo de calibración de cámara que proporcione los parámetros cinemáticos.
- El modelo cinemático inverso del sistema Robot-Visión para transformar las coordenadas imagen (x_i, y_i) en coordenadas cartesianas (x_r, y_r) , empleando los parámetros cinemáticos del algoritmo de calibración .
- El modelo cinemático inverso del robot para transformar las coordenadas cartesianas (x_r, y_r) en coordenadas articulares (q_1, q_2) .

Además de lo anterior, se requiere cierta capacidad de cálculo para poder contar con parámetros precisos y modelos cinemáticos exactos. En la figura 4.1 se puede observar un esquema del sistema actual con todos sus elementos.

Al emplear una RN se elimina la necesidad de contar con estos elementos, ya que no requiere del conocimiento de ningún modelo ni de sus parámetros, pues

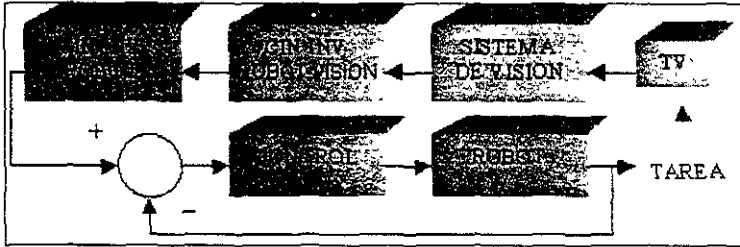


Figura 4.1: Esquema de control actual.

como se mencionó en capítulos anteriores, se basa en el aprendizaje de relaciones entrada-salida.

Al emplear una RN, las coordenadas imagen (x, y) que proporciona el sistema de VA serán transformadas directamente a coordenadas articulares (q_1, q_2) . Dicho de otra manera, la RN deberá aproximar el modelo cinemático inverso del sistema Robot - Visión simplificando así las operaciones computacionales y el tiempo requerido en las tareas de ensamble, prescindiendo de los parámetros cinemáticos del algoritmo de calibración.

En la figura 4.2 se puede observar un esquema del sistema utilizando una RN. Comparando la figura 4.1 con esta última, se aprecia la simplificación obtenida al usar la RN.

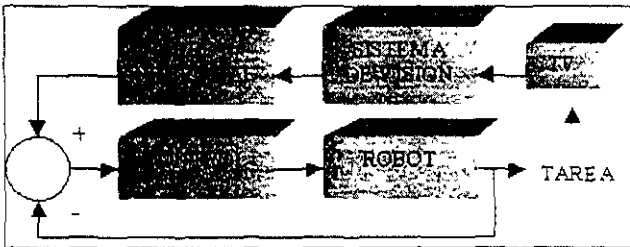


Figura 4.2: Esquema visiomotor propuesto

4.2. Diseño de la RN

4.2.1. Selección del paradigma.

Topología. Como se mencionó anteriormente, la topología o arquitectura de una RN se refiere a la manera en cómo están organizadas las neuronas dentro de la red, esta organización consiste en formar grupos de neuronas los cuales se conocen como capas. Por lo tanto, los parámetros fundamentales que debemos determinar al diseñar la RN son los siguientes.

a) El tipo de conexión entre neuronas. Para resolver el problema que se plantea en este trabajo, es necesario emplear una RN multicapas con conexiones hacia adelante o *feedforward*.

b) El número de neuronas en la capa de entrada y salida. El número de neuronas tanto en la capa de entrada como en la de salida se determinó realizando el siguiente análisis: -

El sistema de VA nos entrega dos coordenadas: x_i y y_i , que corresponden a la posición del objeto a manipular en el plano de la cámara. Por tal motivo, la capa de entrada de la RN tendrá dos neuronas: una para recibir el valor de la coordenada x_i y la otra para recibir el valor de la coordenada y_i .

Por lo que respecta a la capa de salida, de las tres coordenadas articulares que necesita el robot para manipular el objeto, sólo las dos primeras se obtendrán a través de la RN, puesto que la tercera será proporcionada directamente por el sistema de VA. Por lo tanto, la capa de salida tendrá dos neuronas: una para proporcionar el valor de la coordenada q_1 y la otra para proporcionar el valor de la coordenada q_2 .

c) El número de capas ocultas. Para determinar con exactitud el número de capas ocultas y el número de neuronas en cada una de ellas, fue necesario realizar una serie de experimentos hasta encontrar la configuración más adecuada de acuerdo a la complejidad del problema. Esta configuración fue aquella que proporcionó un compromiso entre el menor tiempo de aprendizaje, el menor error en la aproximación del modelo Robot - Visión y el menor número de neuronas.

Los experimentos realizados consistieron en entrenar varias redes con distintas configuraciones, variando tanto el número de capas ocultas como el número de neuronas en cada capa.

La configuración que registró el menor error RMS en el menor tiempo fue la RN con una capa oculta con 10 neuronas, ya que al aumentar el número de neuronas se observó que el error RMS no disminuía considerablemente y por el contrario el tiempo de entrenamiento era cada vez mayor. Esto se debe a que existe un compromiso entre el tamaño de la red y la dimensión del espacio de entrenamiento; es decir, entre mayor sea el espacio de entrenamiento, la RN tiende a ser más grande y emplear más tiempo en su entrenamiento.

Funciones de Activación. La función de activación juega un papel importante en el comportamiento de la RN, ya que es ésta la que determina la aptitud de la RN para aproximar sistemas no lineales.

Para la capa oculta se han seleccionado funciones de activación sigmoidales debido al buen desempeño que presentan en la aproximación de funciones no lineales como la transformación del modelo cinemático inverso del sistema Robot-Visión y en particular con el algoritmo de *backpropagation*.

Por otro lado, la capa de salida contará con funciones de activación lineales, pues sólo se requiere que efectúe la combinación de la información provista por la capa oculta.

Método de Aprendizaje El método de aprendizaje seleccionado fue el método supervisado *backpropagation*, debido a que las RN's que emplean este método son las más adecuadas para trabajos de aproximación de funciones no lineales como el que se requiere en este trabajo.

4.3. Entrenamiento

4.3.1. Entrenamiento en Simulación.

Los modelos matemáticos utilizados en la primer estrategia de entrenamiento, serán los modelos cinemáticos directos y no los modelos cinemáticos inversos. Las razones que motivaron realizar el entrenamiento de esta manera se exponen a continuación:

- Trabajar con el modelo cinemático directo es más sencillo ya que se evita la inversión del modelo y las dificultades introducidas por singularidades del mismo

Articulación	Grados	Radianes
q_1	$-45a + 45$	$-0.7854a + 0.7854$
q_2	$-80a + 80$	$-1.3963a + 1.3963$

Tabla 4.1: Espacio de trabajo

Art.	1	2	3	4	5	6	7	8	9	10
q_1	-0.7854	-0.6283	-0.4712	-0.3142	-0.1571	0.1571	0.3142	0.4712	0.6283	0.7854
q_2	-1.3963	-1.1170	-0.8378	-0.5585	-0.2793	0.2793	0.5585	0.8378	1.1170	1.3963

Tabla 4.2: Puntos de muestreo

- Debido a que la RN basa su aprendizaje en relaciones entrada-salida, podemos emplear indistintamente los vectores x y q en el entrenamiento, es decir, las entradas de la RN pueden ser los vectores x y las salidas los vectores q o viceversa.

Para llevar a cabo el entrenamiento de la RN simulando el sistema Robot - Visión, fue necesario elaborar un programa en el paquete comercial *Matlab*[®]. Este programa se muestra a detalle en el anexo 1 y a continuación se describe brevemente:

1.- Selección de un conjunto de 100 puntos en el espacio articular (sólo para 2 grados de libertad) cubriendo la parte del espacio de trabajo del robot que es vista por el sistema de VA. El volumen total de trabajo del robot *Unimate S-103*, se muestra en la figura 2.4. De éste, sólo se consideró la parte del espacio que corresponde a la mesa de ensamble y que es visto por el sistema de VA. De esta manera se reduce el espacio de trabajo para las dos primeras articulaciones a los rangos que se muestran en la tabla 4.1:

Para obtener los puntos de entrenamiento, se tomaron dentro del espacio reducido 10 puntos de muestreo para cada articulación, siendo éstos igualmente espaciados. La combinación de estos puntos (tabla 4.2) nos da un total de 100 vectores de la forma (q_1, q_2) .

2.- Obtención de los 100 vectores de salida correspondientes (x_i, y_i) aplicando el modelo cinemático directo del sistema Robot - Visión. Para cada uno de los vectores de entrada (q_1, q_2) , es necesario calcular su vector correspondiente en coordenadas cartesianas, utilizando para ello el modelo cinemático directo del robot Unimate S-103 (ecuaciones 2.8 y 2.9).

Con la transformación anterior, se aplica el modelo cinemático directo del sistema Robot-Visión obteniéndose así los 100 vectores de salida correspondientes (x_i, y_i) necesarios para entrenar a la RN.

3. Inicialización de la arquitectura de la RN. En esta parte del procedimiento se inicializa la arquitectura de la RN y se establecen los parámetros de entrenamiento. Además, como parte fundamental del proceso de entrenamiento, se inicializa la matriz de pesos y el vector de polarización.

El número de neuronas en la capa oculta se fija a 10 de acuerdo con los resultados obtenidos en la fase de diseño. Del mismo modo, las dimensiones de la capa de entrada y la capa de salida se inicializan con las dimensiones de la matriz de datos de entrada y de salida respectivamente.

En el proceso de entrenamiento se pretende que la RN aproxime el modelo cinemático inverso del sistema Robot-Visión con un error RMS por abajo del 1 %.

4.- Entrenamiento de la RN con 100 pares entrada-salida. El entrenamiento se llevó a cabo empleando el algoritmo de *backpropagation* en conjunto con los primeros 100 pares de datos entrada-salida.

Al iniciar el entrenamiento, el error rms de la RN supera el 90,000 %. Después de tan sólo 10 épocas, la RN alcanza el error propuesto, terminando de esta forma el entrenamiento.

Los resultados obtenidos se muestran en la figura 4.3, donde puede observarse la evolución del error rms de la salida de la red para el conjunto de pares de entrada-salida.

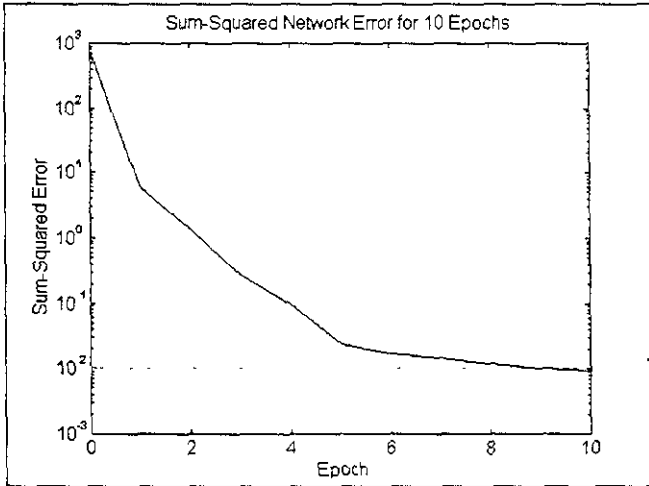


Figura 4.3: Evolución del error RMS.

Para visualizar el error que comete la RN usada como aproximador, se grafican los errores entre los vectores dados por la red y los vectores reales calculados mediante el modelo cinemático directo del sistema Robot-Visión (figura 4.4).

5.- Verificación de la RN con 100 pares entrada-salida. Una vez entrenada la RN, se pueden usar los pesos sinápticos obtenidos en el proceso de entrenamiento - aprendizaje para transformar los 100 vectores restantes q y obtener así sus respectivos vectores x . En la fase de verificación el error rms obtenido con la RN no rebasa el 8 %

Con los resultados obtenidos en simulación, se procede al entrenamiento en la maqueta.

4.3.2. Entrenamiento en la Maqueta.

La segunda estrategia de entrenamiento consistió en entrenar la RN con datos reales obtenidos directamente del Sistema de Visión y del Robot *Unimate S-103*.

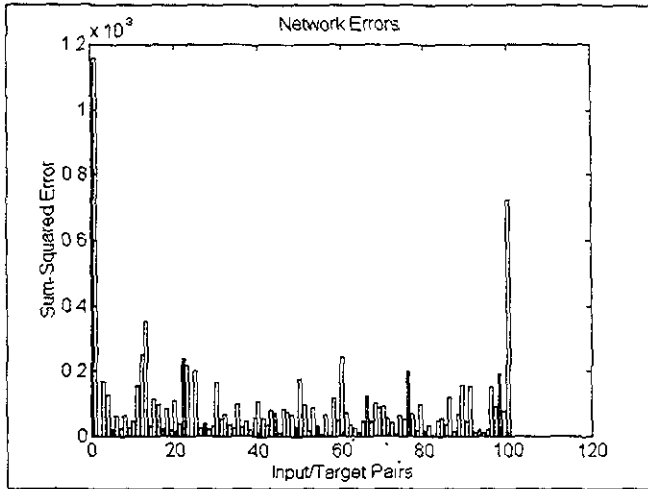


Figura 4.4: Error cometido por la RN usada como aproximador.

Con estos datos y el programa de *Matlab*[®] creado en la primer etapa, se inicia el proceso de entrenamiento-aprendizaje.

Al igual que en la estrategia de simulación, el primer paso es obtener los puntos con los que será entrenada la RN. Por tal motivo, se decidió crear un archivo que contuviera las coordenadas imagen leídas desde el sistema de VA y las coordenadas articulares leídas desde el robot.

El procedimiento para la obtención del archivo se realizó en dos pasos: en el primero, fue necesario cargar los puntos de muestreo en la memoria del robot a través de un procedimiento manual; en el segundo, se ingresó un programa en la memoria del robot con la finalidad de llevar el objeto a cada uno de los puntos para que fuera visto por el sistema de VA y así obtener la posición de dicho objeto tanto en coordenadas imagen como en coordenadas articulares.

A continuación se describen las tareas realizadas en cada uno de los pasos:

1. **Carga de los puntos de muestreo en la memoria del robot.** El espacio de trabajo donde el robot lleva a cabo el ensamble y que es visto por el sistema

de VA, es una mesa de 35 cm. de ancho por 40 cm. de largo, ubicada sobre la base del robot. El objeto a manipular por el robot es un cubo de aluminio de 2.5 cm en cada uno de sus lados. Por tal motivo, el área de la mesa fue dividida de tal manera que cada punto de muestreo quedará separado 5 cm. uno del otro, obteniéndose así un total de 72 puntos en los cuales el robot tendrá que posicionarse.

Debido a que los movimientos del robot están determinados por el contenido de la memoria del controlador, es necesario cargar los puntos de muestreo en dicha memoria. Recordemos que la localización de todos los puntos están almacenados en la memoria del controlador como una tabla de valores representando la posición del brazo con respecto a la posición de referencia (*Home*). Para cargar en la memoria los 72 puntos de muestreo, se llevó a cabo el siguiente procedimiento:

1. 1.1 Se enciende el robot y se lleva a la posición de inicio *Home*.

1.2 Usando el *teach-pendant*, se posiciona el efector final del robot (pinza) en el punto que nos interesa.

1.3 Se selecciona el modo de enseñanza de puntos *Teach*.

1.4 Por medio de las teclas *Prev* y *Next*, elegimos la dirección de memoria en la cual queremos que se guarde la posición del robot.

1.5 Se selecciona la función TCH (*teach*) para que de esta manera el robot memorice la posición en que se encuentra, asignándole la dirección de memoria que escogimos previamente.

1.6 Para grabar los puntos restantes, se repiten las acciones descritas en los puntos 1.2, 1.3, 1.4 y 1.5.

2. Obtención de la posición del objeto con el robot y el sistema de VA. En la figura 4.5 se muestra la secuencia de pasos necesaria para obtener la posición del objeto en coordenadas imagen y coordenadas articulares. Esta secuencia consiste básicamente en tomar el objeto y llevarlo al siguiente punto de muestreo, soltarlo y mover el brazo del robot para que el sistema de VA pueda ver el objeto y determinar su posición y orientación.

A continuación se describe cada uno de los pasos:

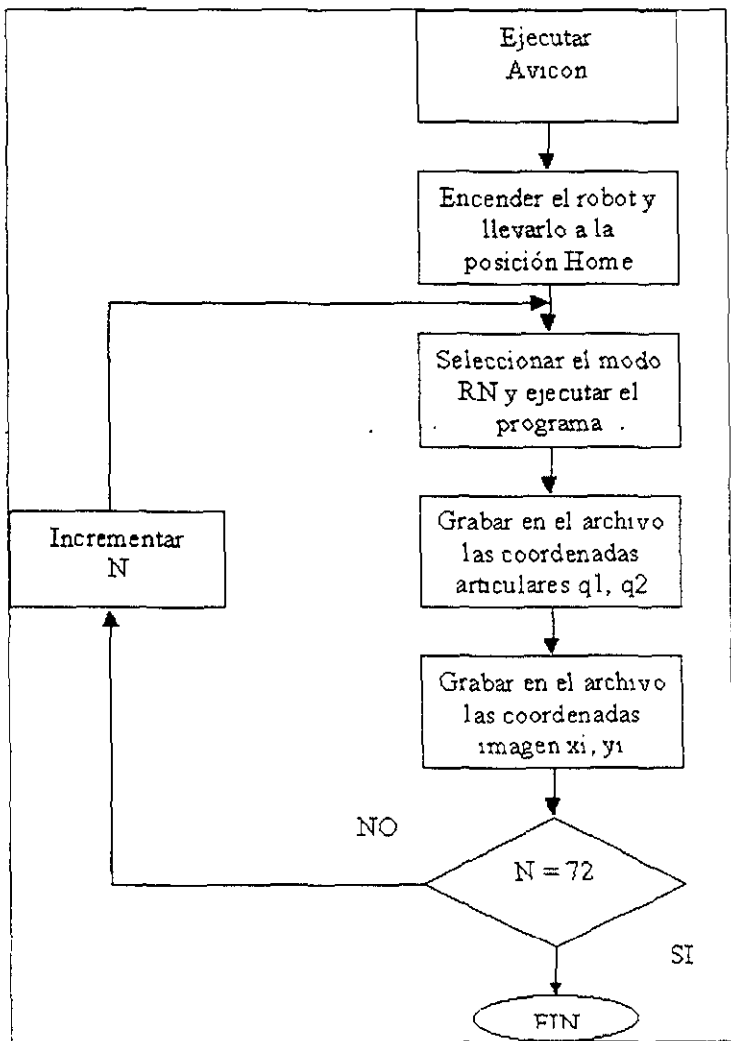


Figura 4.5:

- Se ejecuta el programa Avicon.
- Se enciende el robot y se lleva a la posición *Home*.
- Se selecciona el modo RN (Ejecutar Programa) y se ejecuta el programa cargado previamente en memoria. La secuencia de instrucciones que realiza el robot, se lista a continuación tal y como fue ingresada a la memoria del controlador. La primer columna denota el número de instrucción, la segunda el nombre del comando y la tercera el argumento necesario para ejecutar la instrucción.

{PROGRAMA PROG01.DAT}

000: SP 4 {Velocidad de trabajo 4}
 001: GR 2 {Sube pinza}
 002: WT 5 {Espera que la pinza este arriba}
 003: GR 0 {Abre pinza}
 004: CS 1 {Ejecuta la subrutina 1} Agarra el objeto
 005: CS 2 {Ejecuta la subrutina 2} Deja el objeto
 006: MP 94 {Muevete al punto 94}
 007: GO 3 {Muevete a la etiqueta 3}

{SUBRUTINA 1}

008: LB {Se define la etiqueta 1}
 009: MP {Muevete al punto 1}
 010: GR 3 {Baja pinza}
 011: WT 6 {Espera que la pinza este abajo}
 012: GR 1 {Cierra pinza}
 013: GR 2 {Sube pinza}
 014: WT 5 {Espera que la pinza este arriba}
 015: RE {Retorna de la subrutina}

{SUBRUTINA 2}

016: LB 2 {Se define la etiqueta 2}
 017: MP 2 {Muevete al punto 2}
 018: GR 3 {Baja pinza}

019: WT 6 {Espera a que la pinza este abajo}
020: GR 0 {Abre pinza}
021: GR 2 {Sube pinza}
022: WT 5 {Espera a que la pinza este arriba}
023: RE {Retorna de la subrutina}

- Una vez que el objeto se encuentra en la mesa de ensamble, se graban en el archivo de datos las coordenadas correspondientes a las dos primeras articulaciones siguiendo el procedimiento de lectura de información propioceptiva descrito en el capítulo 2.
- Al ver el objeto que está sobre la mesa de ensamble, el sistema de VA proporciona automáticamente su posición y su orientación. En este momento, las coordenadas x , y y , se graban en el archivo de datos con el par de coordenadas articulares que les corresponde.
- Si el número de puntos leídos es menor a 72, se incrementa el punto; si no, el proceso de lectura termina completandose así el archivo de datos que servirá para entrenar a la RN.

3. Entrenamiento de la RN con un subconjunto de pares de datos. Con la RN obtenida en simulación en conjunto con las primeras 42 parejas de puntos (x_i, q_i) , se inicia el proceso de entrenamiento, estableciendo como datos de entrada las coordenadas imagen x , y como datos de salida las coordenadas articulares q .

La aproximación inicial del modelo cinemático inverso del sistema Robot-Visión que produce la RN, se puede ver en la figura 4.6. El error RMS obtenido en este punto supera el 30,000 %.

Al cabo de 5 épocas el algoritmo de *backpropagation* produce los resultados mostrados en la figura 4.7 donde puede observarse la evolución del error RMS de la salida de la red para el conjunto de pares de entrada-salida. La línea punteada representa el umbral fijado que la RN debía cumplir y que en estas pruebas fue de 1×10^{-3} . De igual manera, en la figura 4.8 se presenta la aproximación final lograda por la RN una vez concluido el entrenamiento.

Para visualizar el error que comete la RN usada como aproximador, la figura 4.9 muestra el error entre los vectores dados por la red y los vectores reales obtenidos directamente del robot y del sistema de visión.

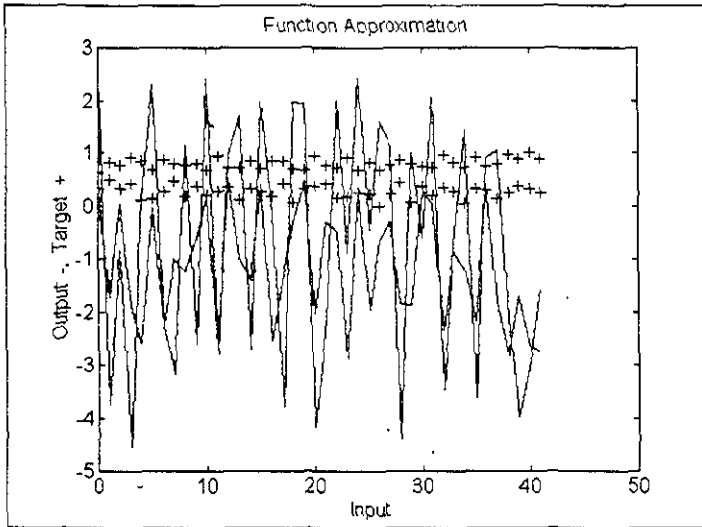


Figura 4.6: Aproximación inicial de la RN.

4. Verificación de la RN con el subconjunto de puntos restante. Una vez entrenada la RN, se pueden usar los pesos sinápticos obtenidos en el proceso de entrenamiento-aprendizaje para transformar los 30 vectores restantes x_i y obtener así sus respectivos vectores q . La figura 4.10 muestra la aproximación que logra la RN en esta fase de verificación.

De la misma manera en la figura 4.11 se puede observar el error obtenido entre los vectores dados por la RN y los vectores de verificación.

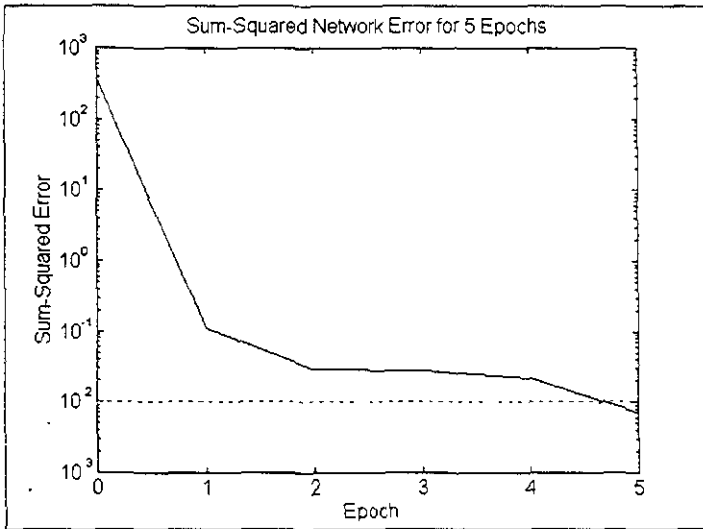


Figura 4.7: Evolución del error RMS.

5. Resultados y Conclusiones

5.1. Resultados

El error final de aproximación que se obtuvo en la fase de entrenamiento utilizando 42 pares de entrada-salida, no rebasó el 0.5 %, mientras que el error individual para cada par (x, q) nunca fue mayor a 0.03 %.

En la fase de verificación, el error de aproximación cometido por la RN en cada par de entrada-salida fue menor al 0.7 %, lo que equivale en milímetros a un error de 2.8 como máximo.

Si tomamos en cuenta que el robot *Unimate S-103* tiene una resolución de 0.5 mm, al emplear la RN se obtiene una resolución total de 3.3 mm.

Debido a que el tamaño de las piezas por manipular es de 25 mm x 25 mm, el error cometido por la RN no representa problemas de manipulación, ya que la distancia que existe entre el objeto por manipular y las pinzas del robot es de

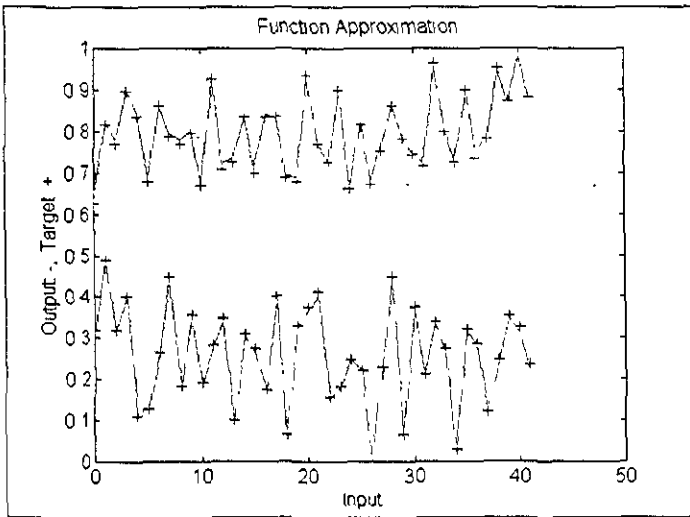


Figura 4.8: Aproximación final de la RN

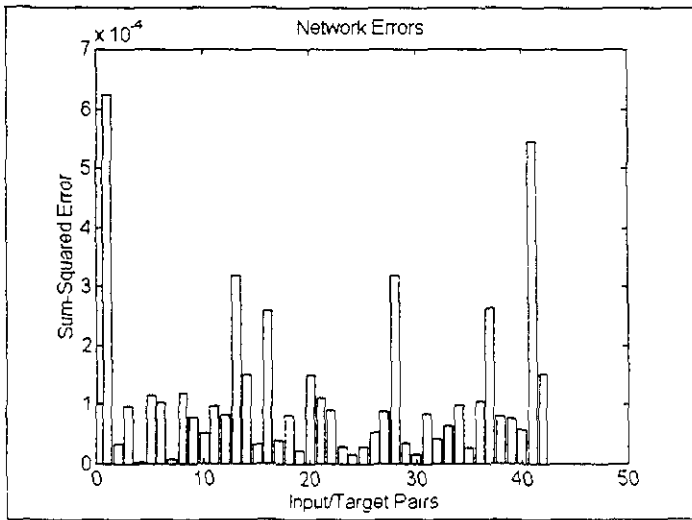


Figura 4.9 Error cometido por la RN usada como aproximador.

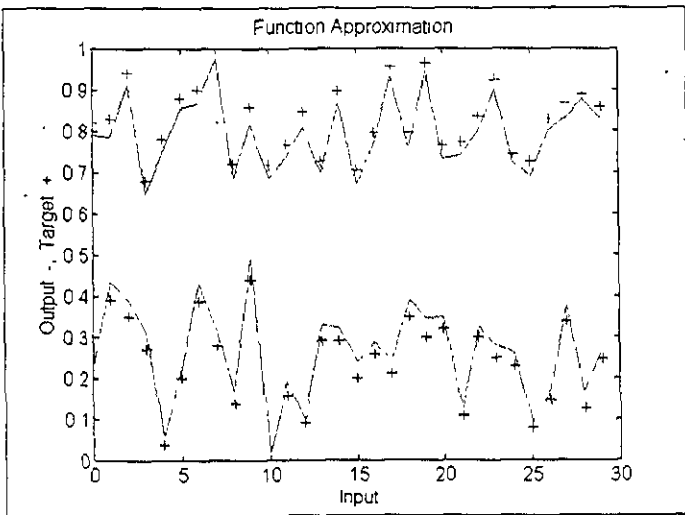


Figura 4.10: Aproximación final de la RN en la fase de verificación.

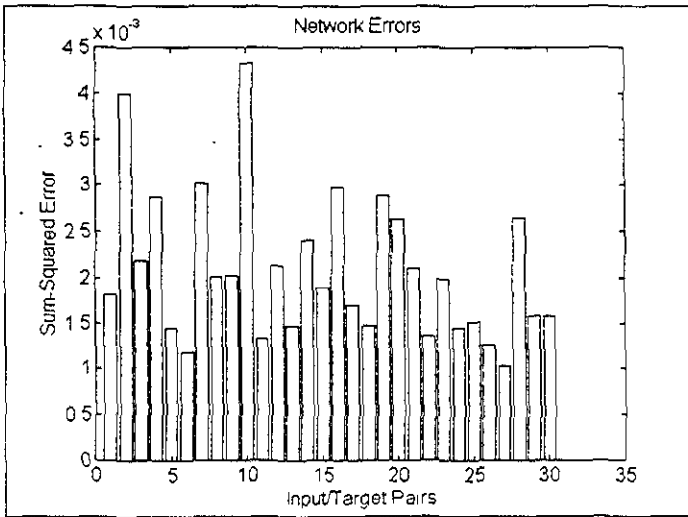


Figura 4 11: Error cometido por la RN usada como aproximador en la fase de verificación.

aproximadamente 10 mm y los 3.3 mm de resolución que tiene el robot al emplear la RN es equivalente a tener un objeto más grande.

5.2. Conclusiones

Los logros alcanzados en el presente trabajo de tesis constituyen un conjunto de ventajas que se resumen a continuación:

- Al emplear la RN en el esquema de control del sistema Robot-Visión, se elimina la necesidad de contar con un algoritmo de calibración de cámara ya que no es necesario conocer los parámetros cinemáticos del robot ($L1$ y $L2$) ni los parámetros del sistema de visión (x_{ori} , y_{ori} , S_x , S_y , α).
- De igual manera, no se requiere el uso de la cinemática inversa del sistema Robot-Visión, puesto que la transformación de coordenadas se realiza directamente a través de la RN.
- Por último, no se requiere el uso de la cinemática inversa del robot *Unimate S-103* para la transformación de coordenadas cartesianas a coordenadas articulares.

5.3. Mejoras

El logro de un mejor desempeño podría obtenerse si se realiza lo siguiente:

- Usar más puntos para el entrenamiento de la RN, ya que entre mayor sea el número de puntos el error de aproximación será menor puesto que la relación entre la información de entrada y salida será más precisa.
- Focalizar el área de entrenamiento ya que el error es relativo al tamaño; es decir, si empleáramos un objeto más chico los puntos de muestreo estarían más cerca uno del otro.

6. Apendice A. Programas de Matlab

6.1. Entrenamiento en Simulación

% Programa: SIMULA1.M
% Autor: V. BAHENA, H. MENDOZA
% Descripción: Entrenamiento de una RN multicapas para la aproximación del modelo cinemático directo del sistema Robot-Visión.

```
clear
clf reset
pausetime = 0.1;

% DEFINICION DEL PROBLEMA
%=====
% Parámetros
nd=180;
np=100;
ntr=0;
va=3;
% Parámetros del robot
% -----
L1 = 0.25;
L2 = 0.35;
% Definición de alfa (alfa = 15 grados)
% -----
alfa = 0.2618;
% Definición de Cx v Cy (coordenadas del origen de Rm en el plano de la
imágen)
% -----
Cx = 0.1;
Cy = 0.05;
h = 0.03;
% Definición de 100 vectores de entrada para entrenamiento (de dos elementos
cada uno)
% -----

Q = [-.7854 - 6283 - 4712 -.3142 - 1571 1571 .3142 .4712 6283 .7854;
```

```

-1.3963 -1.1170 -.8378 -.5585 -.2793 2793 .5585 .8378 1.1170 1.3963 ];
% Definición de 100 vectores de entrada para verificación (de dos elementos
cada uno)
% -----
Q2 = [ -0.7068 -0.5490 -0.3927 -0.2356 -0.0785 -0.0785 -0.2356 -0.3927 -0.5490
-0.7068;
-1.2567 -0.9774 -0.6981 -0.4188 -0.1395 0.1395 0 4188 0.6981 0.9774 1.2567 ];
s = 0;
for j = 1:10
for i = 1:10
s = s + 1;
P(1,s) = Q(1,j);
P(2,s) = Q(2,i);
E(1,s) = Q2(1,j);
E(2,s) = Q2(2,i);
end
end
% Definición de los 100 vectores de salida correspondientes
% -----
A = [ cos(alfa) -sin(alfa);
sin(alfa) cos(alfa) ];
for j = 1:100
for i = 1:2
V(i) = P(i,j);
V2(i) = E(i,j);
end
% se calculan las coordenadas cartesianas de los datos de entrada
% (Xr y Yr) utilizando la cinemática directa del robot.
H(1,1) = L1*cos(V(1)) + L2*cos(V(1)+V(2));
H(2,1) = L1*sin(V(1)) + L2*sin(V(1)+V(2));
H2(1,1) = L1*cos(V2(1)) + L2*cos(V2(1)+V2(2));
H2(2,1) = L1*sin(V2(1)) + L2*sin(V2(1)+V2(2));
% se calculan las coordenadas imagen (Xi y Yi).
H(1,1) = H(1,1) - Cx;
H(2,1) = H(2,1) - Cy;
Ci = h*A*H;
H2(1,1) = H2(1,1) - Cx.

```

```

H2(2,1) = H2(2,1) - Cy;
C2i = h*A*H2;
for i = 1:2
T(i,j) = Ci(i);
S(i,j) = C2i(i);
end
end
[l,np]=size(T);
Ix=0:1:np-1;
Ox=0:1:np-1;
% Normalizacion de los vectores de entrada (P y E) y de salida (T y S).
Pmin=min(min(P));
Pmax=max(max(P));
P=(P-Pmin)/(Pmax-Pmin);
Emin=min(min(E));
Emax=max(max(E));
E=(E-Emin)/(Emax-Emin);
Tmin=min(min(T));
Tmax=max(max(T));
T=(T-Tmin)/(Tmax-Tmin);
Smin=min(min(S));
Smax=max(max(S));
S=(S-Smin)/(Smax-Smin);

% GRAFICA DE LOS VECTORES DE ENTRENAMIENTO
%=====
plot(Ix,P,'+');
title('Vectores de Entrenamiento');
xlabel('Vector de Entrada P');
ylabel('Vector de Salida T');
pause
plot(Ox,E,'+');
title('Vectores de Verificación');
xlabel('Vector de Entrada E');
ylabel('Vector de Salida S');
pause

% INICIALIZACION DE LA ARQUITECTURA DE LA RED

```

```

%=====
% Ajusta el vector de entrada R, el número de neuronas por capa S1 & S2.
(batch size Q.)
[R,Q] = size(P); S1 = 10; [S2,Q] = size(T);
% Inicializa los pesos y el bias
z = menu('Inicializa los Pesos & Bias con:', ...
'Archivo de Entrenamiento'. ...
'Valores Aleatorios [Default]');
disp('')
if z == 1
load cs_b1.dat
load cs_b2.dat
load cs_w1.dat
load cs_w2.dat
W10=cs_w1;
W20=cs_w2;
B10=cs_b1;
B20=cs_b2;
else
[W10,B10,W20,B20] = initff(P,S1,'tansig',T,'purelin');
end

% GRAFICA DE LA APROXIMACION INICAIL
%=====
plotfa(Ix,T,Ix,purelin(W20*tansig(W10*P,B10),B20));
h = get(gca,'Children');
h = h(1);
hold on
pause2(pausetime).

% RUTINA DE ENTRENAMIENTO DE LA RN
%=====
TP(1)=25;
TP(2)=1000;
TP(3)=0 01;
TP(4)=1e-6;
TP(5)=0 001;
TP(6)=10.

```

```

TP(7)=0.1;
TP(8)=1e10;
[W1,B1,W2,B2,TE] = trainlm(W10,B10,'tansig',W20,B20,'purelin',P,T,TP);
totalflops = flops;

```

```

% GRAFICA DE LA APROXIMACION FINAL

```

```

%=====
%CLF
pause;
plotfa(Ix,T,Ix,purelin(W2*tansig(W1*P,B1),B2));
h = get(gca,'Children');
h = h(1);
hold on
pause;
delete(h);
%Ox=0:1:9;
plotfa(Ox,S,Ox,purelin(W2*tansig(W1*E,B1),B2));
hold off
pause

```

```

% GRAFICA DE LOS ERRORES ASOCIADOS A CADA VECTOR DE
SALIDA

```

```

%=====
A1=tansig(W1*P,B1);
A2=purelin(W2*A1,B2);
barerr(A2,T);
pause
A1=tansig(W1*E,B1);
A2=purelin(W2*A1,B2);
barerr(A2,S);
pause

```

```

% RESUMEN DE RESULTADOS

```

```

%=====
fprintf('Verificación')
SSE = sumsq(S-purelin(W2*tansig(W1*E,B1),B2))
fprintf('Entrenamiento')
SSE = sumsq(T-purelin(W2*tansig(W1*P,B1),B2))

```



```

fprintf('\nRESULTADOS DEL ENTRENAMIENTO:\n')
W1
B1
W2
B2
save cs_w1.dat W1 /ascii /double
save cs_b1.dat B1 /ascii /double
save cs_w2.dat W2 /ascii /double
save cs_b2.dat B2 /ascii /double
fprintf('Entrenado en %.0f epochs.\n',TE)
fprintf('El entrenamiento tom6 %.0f flops.\n',flops);
fprintf('Suma del cuadrado del error propuesto %g.\n',TP(3));
fprintf('Suma del cuadrado del error obtenido %g.\n',SSE);
fprintf('El entrenamiento de la red fue: ');
if SSE < TP(3)
disp('Adecuado.')
else
disp('No adecuado, no llego al error propuesto.')
end
drawnow

```

6.2. Entrenamiento en la Maqueta

```
% Programa: MAQUETA1.M
% Autor: V. BAHENA, H. MENDOZA
% Descripción: Entrenamiento de una RN multicapas para la aproximación
del modelo cinemático inverso del sistema Robot-Visión.
```

```
clear
clf reset
pausetime = 0.1;

% DEFINICION DEL PROBLEMA
%=====
% Parámetros
nd=180;
np=100;
ntr=0;
va=3;
% Lectura de los datos de entrenamiento (entrada y salida).
%-----
load c:\tesis\prog_m\entrada3.dat;
load c:\tesis\prog_m\salidas.dat;
P=entrada3';
T=salidas';
[l,np]=size(T);
Ix=0:1:np-1,
% Normalizacion del vector P y el vector T
Pmin=min(min(P));
Pmax=max(max(P));
P=(P-Pmin)/(Pmax-Pmin);
Tmin=min(min(T));
Tmax=max(max(T));
T=(T-Tmin)/(Tmax-Tmin);
% Lectura de los datos de verificación (entrada y salida).
%-----
load c:\tesis\prog_m\img_x.dat;
load c:\tesis\prog_m\img_y.dat;
Vx=img_x';
```

```

Vy=img_y';
load c:\tesis\prog_m\ent_pru.dat;
load c:\tesis\prog_m\sal_pru.dat;
E=ent_pru';
S=sal_pru';
[l,nd]=size(S);
Ox=0:1:nd-1;
% Normalizacion del vector E y el vector S
Emin=min(min(E));
Emax=max(max(E));
E=(E-Emin)/(Emax-Emin);
Smin=min(min(S));
Smax=max(max(S));
S=(S-Smin)/(Smax-Smin);

% GRAFICA DE LOS VECTORES DE ENTRENAMIENTO
%=====
plot(Vx,Vy,'+');
title('Vectores de Entrenamiento');
xlabel('Vector de Entrada P');
ylabel('Vector de Salida T');
pause

% INICIALIZACION DE LA ARQUITECTURA DE LA RED
%=====
% Ajusta el vector de entrada R, el número de neuronas por capa S1 & S2,
(batch size Q.)
[R,Q] = size(P); S1 = 10; [S2,Q] = size(T);
% Inicializa los pesos y el bias
z = menu('Inicializa los Pesos & Bias con:', .
'Archivo de Entrenamiento', .
'Valores Aleatorios [Default]');
disp("")
if z == 1
load c:\tesis\prog_m\optimos\cs_b1.dat
load c:\tesis\prog_m\optimos\cs_b2.dat
load c:\tesis\prog_m\optimos\cs_w1.dat
load c:\tesis\prog_m\optimos\cs_w2.dat

```

```

W10=cs_w1;
W20=cs_w2;
B10=cs_b1;
B20=cs_b2;
else
[W10,B10,W20,B20] = initff(P,S1,'tansig',T,'purelin');
end

% GRAFICA DE LA APROXIMACION INICIAL
%=====
plotfa(Ix,T,Ix,purelin(W20*tansig(W10*P,B10),B20));
h = get(gca,'Children');
h = h(1);
hold on
% pause2(pausetime);
pause

% RUTINA DE ENTRENAMIENTO
%=====
TP(1)=25;
TP(2)=1000;
TP(3)=0.01;
TP(4)=1e-6;
TP(5)=0.001;
TP(6)=10;
TP(7)=0.1,
TP(8)=1e10;
[W1.B1,W2.B2,TF] = trainlm(W10.B10,'tansig',W20.B20,'purelin',P,T,TP).
totalflops = flops;

% GRAFICA DE LA APROXIMACION FINAL
%=====
%CLF
pause;
plotfa(Ix.T,Ix.purelin(W2*tansig(W1*P B1).B2));
h = get(gca,'Children');
h = h(1),
hold on

```

```

pause;
delete(h);
%Ox=0:1:9;
plotfa(Ox,S,Ox,purelin(W2*tansig(W1*E.B1),B2));
hold off
pause

% GRAFICA DE LOS ERRORES ASOCIADOS A CADA VECTOR DE
SALIDA
% =====
A1=tansig(W1*P,B1);
A2=purelin(W2*A1,B2);
barerr(A2,T);
pause
A1=tansig(W1*E,B1);
A2=purelin(W2*A1,B2);
barerr(A2,S);
pause

% RESUMEN DE RESULTADOS
% =====
fprintf('Verificación')
SSE = sumsq(S-purelin(W2*tansig(W1*E.B1),B2))
fprintf('Entrenamiento')
SSE = sumsq(T-purelin(W2*tansig(W1*P.B1),B2))
fprintf('\nRESUMEN FINAL DE RESULTADOS DE LA RED.\n')
W1
B1
W2
B2
save cs_w1.dat W1 /ascii /double
save cs_b1.dat B1 /ascii /double
save cs_w2.dat W2 /ascii /double
save cs_b2.dat B2 /ascii /double
fprintf('Entrenado en %.0f epochs.\n',TE)
fprintf('El entrenamiento tomó %.0f flops \n',flops);
fprintf('Suma del cuadrado del error propuesto %g.\n',TP(3)).
fprintf('Suma del cuadrado del error obtenido %g.\n',SSE);

```

7. Referencias

[1] J. R. Hilera, Victor J. Martinez "Redes Neuronales Artificiales. Fundamentos, Modelos y Aplicaciones". Ed. Addison-Wesley Iberoamericana

[2] Richard P. Lippman "An Introduction to Computing with Neural Nets" IEEE ASSP Magazine. Abril de 1987.

[3] R. C. Eberhart and R. W. Dobbins, "Neural Networks PC Tools A Practical Guide". Academic Press Inc. 1990.

[4] J.M. Ibarra Zannatha, I. Mazaira. C.A. Campos. "Desarrollo de un sistema de manipulación robotizada de objetos". Memoria del Congreso. Morelia, Mich. marzo 1996.

[5] J.M. Ibarra Zannatha, R. Kelly, D. Bassi. "Visual connectionist control of robots: Formulation and practical issues". Workshop on Fuzzy Intelligent Neural Systems FINS '94. Fort Worth, Texas, USA. December 1994.

[6] J.M. Ibarra Zannatha, D. Bassi, R.A. García. "Position and differential kinematic neural control of robot manipulators: A comparison between two schemes". Proc. of IEEE Int. Conf. on SMC. Vol. 4, pp479-484. Le Touquet, France, October 1993.

[7] D. Bassi, J.M. Ibarra Zannatha. "Sensorial robot control based on connectionist models for assembly tasks" Proc of AMCA-IEEE Int. Workshop on Neural Networks Applied to Control and Image Processing NNACIP'94. México, D.F. November 1994.

[8] Bart Kosko (Ed.). "Neural Networks for Signal Processing". Prentice Hall, Englewood Cliffs, NJ, USA. 1992.

[9] S.W. Wijwsoma, D.F.H. Wolfe, R.J. Richards. "Eye to hand coordination for vision guided robt control applications". The International Journal of Robotics Research. Vol. 12 No 1 pp. 65-78. Febraury 1993.

[10] John J. Craig "Introduction to Robotics: Mechanics and Control" Addison Wesley Publishing Co. Reading, MA, USA. 1986

[11] A. J. Malo Tamayo. "Manua de usuario del robot Unmation S-103". Departamento de Publicaciones Técnicas del Cinvestav-IPN. Serie Azul

[12] H. Demuth, M. Beale. "Neural Network Toolbox User' Guide". The MathWorks Inc., 1992.

[13] I Mazaira Tesis de Maestría, Cinvestav 1995