Control de un Robot Basado en Retroalimentación Visual.

Sebastian Ibarra Rojas.

División de Estudios de Posgrado de la Facultad de Ingeniería UNAM.

Maestro en Ingeniería Eléctrica

Índice General

Capítulo 1

Introducción

1.1 Motivación del trabajo

Actualmente se puede considerar que la presencia de los robots (manipuladores) es indispensable casi en todos los campos de la vida moderna: el campo industrial, ensamble, selección y soldadura en plantas de fabricación de semiconductores, en ensamble de automóviles, ensamble y verificación de tarjetas electrónicas; en el campo de la investigación, manipuladores operados a control remoto en plantas químicas y nucleares, vehículos operados a control remoto capaces de tomar multiples datos de su ambiente; en el campo doméstico sistemas de control de accesos, juguetes, etc. Debido a la gran cantidad de aplicaciones que existen y que siempre está en aumento, además de que el desempeño requerido de estos robots es cada día mayor en cuanto a precisión y velocidad se hace por tanto necesario un estudio detallado de las técnicas de control de manipuladores robóticos más recientes.

El diseño de los robots industriales actuales se basa en servo-mecanismos simples (codificadores ópticos) asignados a las diferentes juntas de la estructura. Esto resulta en una respuesta reducida en velocidad, además de limitar la precisión y exactitud del efector final lo cual produce un desempeño sub-óptimo. Las tareas incrementalmente sofisticadas requeridas de los manipuladores robóticos demandan mejores técnicas de control que mejoren la velocidad y exactitud del efector final operando en ambientes inciertos. Se han desarrollado muchos esquemas de control para mejorar el desempeño de los manipuladores robóticos, pero el desempeño de estos métodos es altamente dependiente de la exactitud del modelado dinámico del robot.

Una de las ventajas del uso de las redes neuronales como método de control de los manipuladores robóticos, es precisamente que no se requiere del conocimiento a priori de un modelo del manipulador. Esta técnica de control es genérica en el sentido de que los parámetros del controlador no dependen de alguna estimación, lo que es opuesto a los métodos convencionales de control adaptable. Por otro lado, el hecho de utilizar sensores independientes de la estructura del manipulador como son las cámaras de video, proporciona varias ventajas. Utilizando sensores externos, un robot puede tener un comportamiento adaptable: el robot

será flexible a cambios en su ambiente y podrá ejecutar tareas inteligentes. También se debe mencionar que la principal desventaja al utilizar sensores de visión es la enorme potencia computacional requerida.

El campo de la visión de máquina [23], [21], [22] también resulta en una buena área de aplicación para las redes neuronales, por supuesto que no es un nuevo paradigma dentro de las redes neuronales, debido a que el problema de la visión de máquina no es un problema reciente.

1.2 Revisión literaria

El tema de las redes neuronales fue introducido desde los años cuarenta, después de un auge inicial se detuvó el desarrollo, esto debido a la aparición del libro [18] "Perceptrons" que presenta las limitaciones del perceptrón de "Rumelhart" [29], en cuanto a la capacidad de aprendizaje de la unidad básica de las redes neuronales, el perceptrón. Esto provocó un desánimo en los investigadores y de esta forma se retrasó varios años el desarrollo de algoritmos de aprendizaje y nuevas arquitecturas de redes.

Desde mediados de los años ochenta ha retomado un nuevo auge el desarrollo y aplicación de las redes neuronales [27], esto debido principalmente a la aparición de algoritmos de aprendizaje eficientes y al aumento en la capacidad de cálculo de las computadoras. En el año de 1990 [1]se propuso que las redes neuronales pueden también utilizarse como componentes en sistemas dinámicos y que un estudio de tales sistemas debe formar parte de la estructura teórica, unificada de sistemas. En particular se propone [21] que los procedimientos adoptados exitosamente para el control práctico de sistemas lineales, combinando modelado matemático, cálculo y experimentación también deben ser aplicados en control no lineal basado en redes neuronales. Se ha tenido un gran progreso desde entonces, tanto en la teoría [26] como en la práctica del control basado en redes neuronales

Extensivas simulaciones por computadora han sido realizadas por los investigadores y muchas han sido implementadas, en el campo de la industria aerospacial y manipuladores industriales, principalmente en los campos de la aproximación de funciones y el reconocimiento de patrones [27], [28].

1.2.1 Redes neuronales

A pesar de que existen numerosos métodos de aproximación de funciones, basados en polinomios, funciones trigonométricas, funciones ortogonales, aproximaciones de Padé, las redes neuronales son preferidas a tales métodos en muchas situaciones prácticas y particularmente para control en línea. En vista de su arquitectura éstas son implementables más fácilmente en "hardware". El trabajo de Barron [3] proporciona justificación teórica parcial para el uso de redes neuronales sobre sus competidores.

En problemas complejos de control las dificultades que se encuentran pueden clasificarse en términos generales bajo las siguientes categorías: (i) complejidad, (ii) no linealidad (iii) incertidumbre.

El método de las redes neuronales es muy conveniente para cubrir las tres categorías de dificultades. Su habilidad para aproximar mapas arbitrarios no lineales en un conjunto compacto y la disponibilidad de métodos para ajustar sus parámetros sobre la base de datos de entrada y de salida, hace de las redes neuronales particularmente atractivas cuando no linealidades desconocidas están presentes en el sistema.

En problemas de control, también es su capacidad de aproximación de las redes neuronales lo que las hace atractivas en identificadores y controladores. Desde un punto de vista teórico, la existencia de mapeos no lineales para la representación del proceso a ser controlado y en el controlador se tiene que establecer antes de la aplicación de redes neuronales.

De las características más importantes de las redes neuronales podemos mencionar su capacidad de aproximar funciones no lineales y su capacidad de procesamiento paralelo, éstas resultan por demás atractivas en el campo del control, ya que, por un lado no es indispensable obtener el modelo del sistema, pues el mapeo entrada salida se puede obtener con el entrenamiento de una red neuronal y por otro lado la capacidad de procesamiento paralelo permite crear aplicaciones en tiempo real.

1.2.2 Aplicaciones de las redes neuronales en el control de manipuladores robóticos.

Los robots industriales actuales están extremadamente limitados en sus capacidades. El método de las redes neuronales tiene la capacidad de proporcionarles la habilidad de aprender, adaptar e integrar información de múltiples entradas. Las redes neuronales han encontrado aplicaciones prácticas en el campo de la robótica. Los problemas en robótica que están siendo atacados actualmente son:

Programación y control de trayectorias de estructuras robóticas multi - ejes [27].
Navegación robótica [8].
Coordinación de robot - cámara coordinación ojo - mano [24].
Fusión tacto/visual para reconocimiento de objetos [8].

Estos problemas han sido abordados usando varios modelos de redes neuronales, incluyendo: perceptrones multicapa y redes de Hopfield, "acercamientos" simulados. En algunos casos, las redes neuronales son una nueva forma de implementar una idea existente en robótica, mientras en otros casos son el núcleo de nuevos métodos en robótica.

1.3 Problema

En el presente trabajo se pretende comprobar el funcionamiento de las redes neuronales en el campo de la robótica, específicamente "El Control Visual de un Robot por medio de una Red Neuronal". Para poder utilizar las redes neuronales en aplicaciones prácticas, debemos realizar el proceso de aprendizaje de la red, este será el proceso clásico de retropropagación de error, en el siguiente capítulo se detalla este.

El presente trabajo tiene por objetivo controlar un robot para realizar seguimiento de una trayectoria definida dentro del espacio de trabajo, utilizando retroalimentación visual y aplicando redes neuronales para aproximar el mapeo de coordenadas de imagen a coordenadas espaciales.

1.4 Contribuciones

Las contribuciones principales del presente trabajo son:

- Implementación de un algoritmo "algoritmo de retropropagación rápida" para el entrenamiento de una red neuronal.
- Comprobación del diseño de una red neuronal propuesta, por medio de simulación y por medios experimentales.
- Implementación de un sistema de control de un manipulador robótico basado en "Labview".
- Comparación del comportamiento del algoritmo de aprendizaje de la red neuronal propuesto, con "ANFIS" (Adaptive - Network - Based Fuzzy Inference System) caja de herramientas de Matlab y con la caja de herramientas de redes neuronales "NN" de Matlab.
- Obtener los primeros resultados de control del manipulador de 5 grados libertad,del laboratoriode robótica de la Sección de Eléctrica.de la DEPFI.

1.5 Organización

En la primera parte de este trabajo se presenta una introducción a las redes neuronales; definiciones, arquitectura y formas de entrenamiento utilizadas. Se presenta también la forma general de la ecuación de un robot y algunas estrategías básicas de control de posición.

En la segunda parte se presentan los problemas de visión robótica, se define visión artificial y se comenta el proceso de adquisición de imágenes por medio de cámaras CCD[1]. Se aborda uno de los temas principales de este trabajo, la utilización de las redes neuronales

[1] CCD Dispositivos de acoplamiento de carga

para realizar retroalimetación visual, se presentan resultados de simulación, así mismo se presenta una comparación con las cajas de herramientas de Matlab[2].

En la última parte se presentan los pasos necesarios para realizar control de manipuladores robóticos con retroalimentación visual, se presentan los resultados experimentales obtenidos y finalmente las conclusiones.

[2]Cajas de herramientas NN y ANFIS.

Capítulo 2

Redes neuronales (NN)

2.1 Introducción

Quizá una de las más excitantes e inovadoras técnicas desarrolladas en los pasados diez años en el campo del control ha sido la introducción de métodos de las redes neuronales para modelado, identificación y control. Como cualquier tema nuevo, este sufre aún de una gran cantidad de elogios en el sentido de considerar este método como el mejor de todos sus predecesores, por otro lado también sufre de ataques de sus críticos, quienes insisten en que este método no es nada nuevo y no tiene nada que ofrecer. Una red neuronal es una estructura computacional modelada sobre procesos biológicos [8]. En su nivel más fundamental, el interés en las redes neuronales se resalta por dos hechos:

(a) El sistema nervioso, hasta del animal más pequeño puede resolver problemas que son muy difíciles para la computadora convencional, incluyendo la mejor computadora disponible en este momento.

(b) La habilidad de modelar el funcionamiento del sistema nervioso utilizando máquinas hechas por el hombre incrementa el entendimiento de esta función biológica.

Las neuronas artificiales, son análogas a sus inspiradoras biológicas. Aquí las neuronas se hacen elementos de procesamiento, los axones y dendritas se hacen cables y las sinápsis se hacen resistores variables, llevando entradas "pesadas" que representan datos o la suma de pesos de aún otros elementos de procesamiento.

Estas entradas de voltajes, que son proporcionales a los pesos que han sido establecidos, son sumadas por medio de las resistencias. Las resistencias son conectadas a un amplificador operacional sobre el cual ha sido conectado un umbral, tal que cuando la suma de estos voltajes alcanza un umbral predefinido, la neurona o el elemento de procesamiento se activa. La figura asume que este elemento de procesamiento tiene dispositivo de limitación fuerte: esto es, cuando la suma de voltajes está por abajo del valor de umbral el dispositivo tendrá una salida de -1.0 y no se activa; cuando la suma alcanza el valor de umbral su salida será +1.0 y el dispositivo se activa.

Los elementos de procesamiento pueden interactuar en muchas formas, en virtud de la forma que estos estén interconectados, los elementos de procesamiento son únicamente prealimentados; los elementos de procesamiento tienen un lazo de retroalimentación. Elementos de procesamiento que son completamente conectados a todos los elementos de procesamiento; elementos de procesamiento que son conectados sólo a algunos otros elementos de procesamiento.

La naturaleza y número de estos lazos de retroalimentación y conexiones dependen del modelo, o arquitectura, utilizada para construir la red neuronal. El diseño de los lazos de retroalimentación de una red neuronal tiene implicaciones para la naturaleza de su adaptabilidad/entrenabilidad, mientras el diseño de las interconexiones tiene implicaciones para su paralelismo.

2.2 Definiciones

Definición 2.1 ***Red Neuronal (NN):*** *Una red neuronal es un sistema de procesamiento de datos, que consiste en un gran número de elementos simples de procesamiento altamente interconectados en una arquitectura inspirada por sistemas nerviosos biológicos. Una red neuronal es un dupla.*

$$NN \triangleq (ARCH, REGLA)$$

Donde ARCH es la arquitectura de la red, la cual representa características espaciales de una red neuronal y REGLA denota las reglas para el procesamiento de información, la cual determina la característica temporal de una NN.

Se debe decir que se pueden construir redes neuronales estáticas y dinámicas. En las estáticas no existen conexiones de retroalimentación ni retardos en las mismas, hecho que sí se dá en las dinámicas.

Definición 2.2 ***Arquitectura de una red neuronal:*** *La arquitectura de una red neuronal, es una quinteta que define la topología de la red como:*

$$ARCH \triangleq (x(t), y(t), a(t), w(t), f)$$

Donde: $x(t) \in X \subseteq \Re^n$, un vector columna con n componentes de variables de entrada externas; $y(t) \in Y \subseteq \Re^m$, un vector columna con m componentes de variables de salida externas: $a(t)$ es un vector columna con N de niveles de activación correspondientes a N neuronas; $w(t) \in A \subseteq \Re^N$, $W \subseteq \Re^M$, un vector columna con M componentes de parámetros adaptables los cuales pueden descomponerse en tres vectores: $w(t) = [w_{nn}^T(t), w_{in}^T(t), w_{th}^T(t)]^T$, los cuales corresponden a, pesos entre neuronas, pesos de entradas externas a neuronas y umbrales respectivamente, $f \in F \subseteq C$, es un vector columna de N funciones de activación o de transferencia continuas.

Observación 2.1 *: La arquitectura de una red neuronal es completamente definida por cinco elementos. Específicamente, x(t), y(t); la entrada y salida externa de una NN, a(t) determina los estados internos, w(t) determina la topología de conectividad y el comportamiento funcional y f determina las características de nodo de una NN.*

Para que por una red neuronal transite y se procese información se deben especificar un conjunto de reglas.

Definición 2.3 ***Regla de una NN****: La regla de una NN es un dueto*

$$REGLA \triangleq (PR, LR)$$

Definición 2.4 ***Regla de Propagación****: Una regla de propagación (PR), es una regla para el tránsito de información de entrada externa a la salida de una NN, esto es:*

$$PR = X \times Y \times A \times W \times F \longrightarrow Y$$

Definición 2.5 ***Regla de aprendizaje LR****: Una regla de aprendizaje, es una regla para determinar los parámetros w basado en la información disponible, esto es,* $LR = X \times Y \times A \times W \times F \times E \longrightarrow W$ *donde E es el conjunto de las funciones de evaluación de desempeño.*

Observación 2.2 *LR (regla de aprendizaje) está relacionada con la codificación del conocimiento contenido en los presentes datos de entrenamiento y el conocimiento que poseen los entrenadores. LR es de importancia primordial para diseñar una NN. Una LR buena debe ser efectiva y eficiente. La efectividad significa que una NN sintetizada es representable para cierto tipo o clase de procesos; eficiencia significa que el tiempo computacional del proceso de aprendizaje es relativamente corto.*

2.3 Modelo de una neurona

Se muestra en la Fig. 2.1 el modelo de una neurona, la entrada escalar p se transmite a través de una conexión que multiplica su magnitud por el peso escalar w para formar el producto wp, también un escalar que es sumado con el factor de desplazamiento b, éste resultado de la suma se presenta como el argumento de la función de activación o función de transferencia, que proporcionará el escalar a.

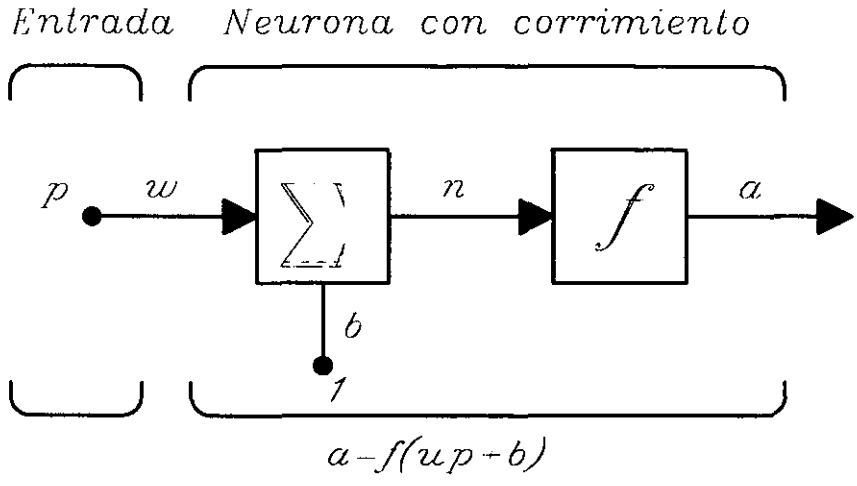

Figura 2.1: Neurona artificial básica

2.4 Funciones de transferencia

Se presentan enseguida varias funciones de transferencia que se utililizan en la construcción de redes neuronales

Función de transferencia de límite fuerte, ésta función de transferencia limita la salida de la neurona a cero, si el argumento de entrada n es menor que cero ó uno, si n es mayor o igual a cero. Esta función de transferencia se utiliza para crear neuronas que toman desiciones de clasificación.

Función de transferencia lineal, esta función como su nombre lo dice limita la salida de la neurona hasta menos uno en forma lineal, si el argumento de la función n es menor que cero y hasta más uno en forma lineal si n es mayor o igual a cero, este tipo de función de transferencia se utiliza en la construcción de neuronas en filtros lineales adaptables.

Función de transferencia sigmoidal, esta función de transferencia toma la entrada, la cual puede tomar valores entre más infinito y menos infinito, realiza un suavizamiento y que como salida un valor entre cero y uno, este tipo de función de transferencia se utiliza en construcción de redes por retropropagación.

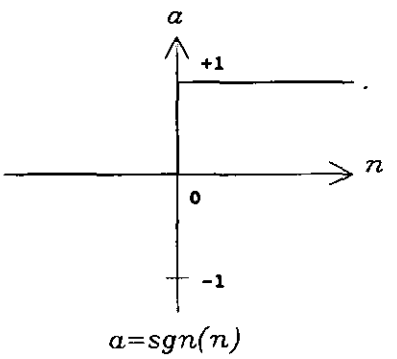

Figura 2.2: Función de transferencia limite fuerte

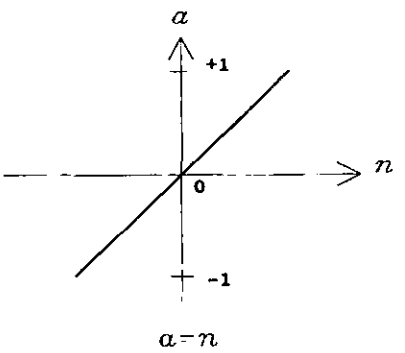

Figura 2.3: Función de transferencia lineal

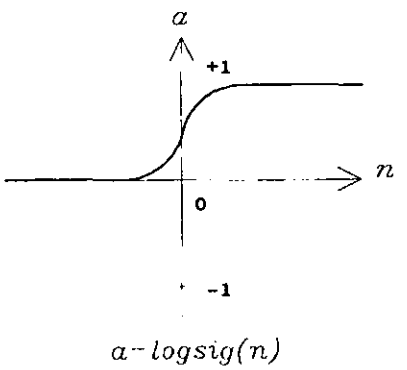

Figura 2.4: Función de transferencia no lineal

2.5 Neurona con vector de entrada

En la Fig 2.5 se presenta una neurona con un vector de entrada de R-elementos, aquí los elementos de entrada p_1, p_2, ... p_R son multiplicados por los pesos $w_{1,1}$, $w_{1,2}$, ... $w_{1,R}$ y los valores pesados (ponderados) son alimentados al nodo de suma. Esta suma es simplemente Wp, el producto punto de la matriz W (de un sólo renglón) y el vector p.

La neurona tiene un factor de corrimiento b el cual es sumado con las entradas ponderadas para formar la entrada n de la red. Esta suma, n, es el argumento de la función de transferencia f

$$n = w_{1,1}p_1 + w_{1,2}p_2 + ... + w_{1,R}p_R + b$$

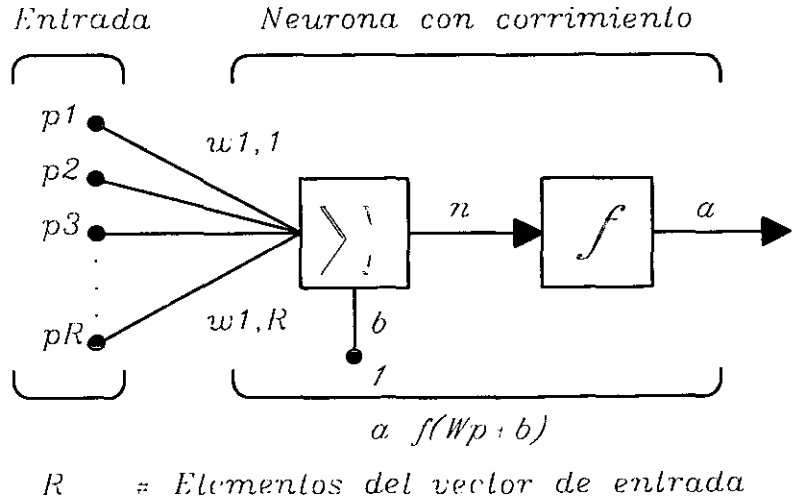

Figura 2.5: Neurona con vector de entrada

2.6 Arquitecturas de red

2.6.1 Una capa de neuronas

En la Fig. 2.6 se presenta una red de una capa con R elementos de entrada y S neuronas.

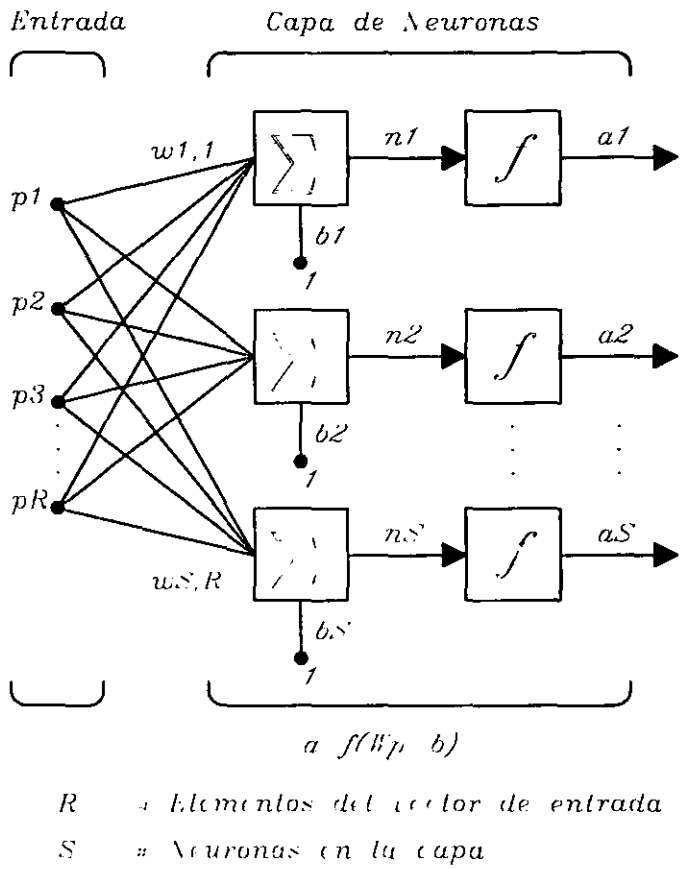

Figura 2.6: Capa simple de neuronas

En esta red, cada uno de los elementos del vector de entrada p es conectada a cada una de las entradas de la neurona a través de la matriz de pesos W. La i-ésima neurona tiene un sumador, que suma sus entradas pesadas y factores de corrimiento, para formar su propia salida escalar $n(i)$. Los valores $n(i)$ tomados juntos forman un elemento S finalmente, las salidas de la capa neuronal forman el vector a.

Observación 2.3 *Observe que es común que el número de entradas a la red neuronal sea diferente que el número de neuronas. También se pueden crear redes neuronales con diferentes funciones de transferencia, simplemente poniendo (conectando) las redes en paralelo. éstas tendrán las mismas entradas y cada una de las redes podrían crear algunas de las salidas.*

Los elementos del vector de entrada entran a la red a través de la matriz de pesos:

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & \vdots & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}$$

Observe que los índices de renglón en la matriz de pesos W indican la neurona destino de los pesos y los índices de columna indican cual es la fuente de entrada para estos pesos. De esta forma, para el elemento $w_{1,2}$ se dice que es el peso para la entrada dos y está conectada a la neurona uno.

2.6.2 Multiples capas de neuronas

Una red neuronal puede tener varias capas. Cada una de las capas tiene una matriz de pesos W, un vector de corrimientos b y un vector de entradas a. Para distinguir las matrices de pesos, vectores de salida, etc., para cada una de las capas en la siguiente figura, se anexará el número de la capa como superíndice a la variable de interés.

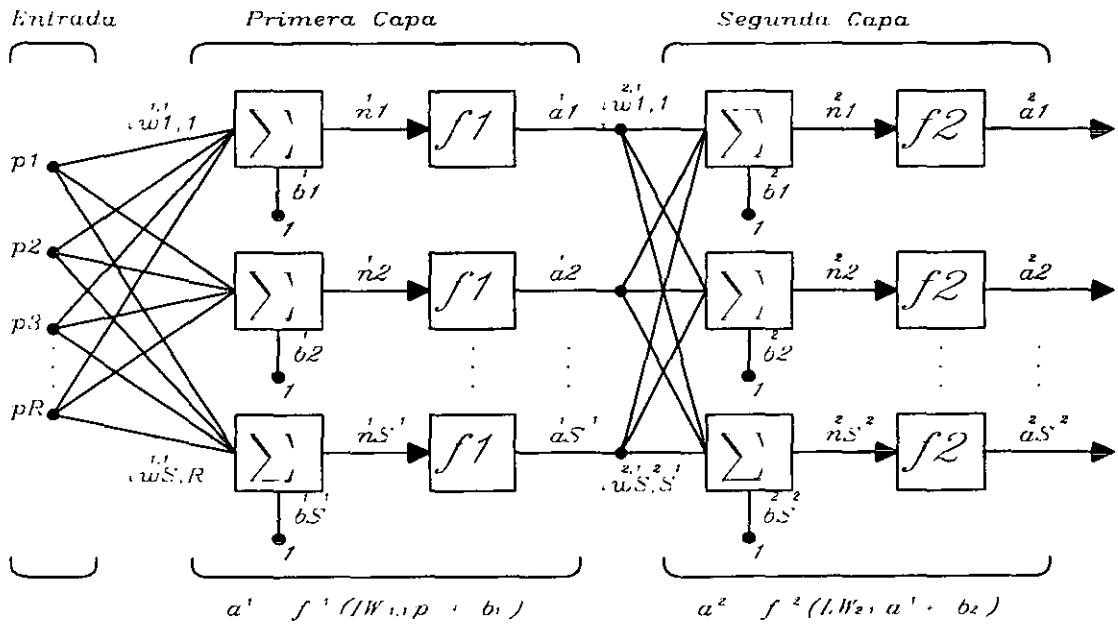

Figura 2.7: Multiples capas de neuronas

Observando la figura anterior se puede concluir que las salidas de cada capa intermedia son las entradas a la siguiente capa. De esta forma, la capa dos puede analizarse como una

red de una capa con S^1entradas, S^2 neuronas y una matriz de pesos W^2 con dimensión $S^1 \cdot \times S^2$. La entrada a la capa 2 es a^1, la salida es a^2.

Las capas de una red neuronal multicapa juegan diferentes roles. La capa que produce la salida de la red es llamada la capa de salida, todas las demás capas son llamadas capas ocultas.

Las redes multicapa son muy poderosas. Por ejemplo, una red de dos capas, donde la primera capa es sigmoidal y la segunda capa es lineal, puede ser entrenada para aproximar cualquier función (con un número finito de discontinuidades) arbitrariamente bien, en un conjunto compacto. Esta clase de red neuronal de dos capas se usa extensivamente en retropropagación [1].

2.6.3 Algunas topologías importantes de redes neuronales

Se ha mencionado anteriormente que existen redes neuronales estáticas y dinámicas, a pesar de esto en el presente trabajo sólo se analizarán las redes neuronales estáticas completamente conectadas y con prealimentación.

Enseguida se presentan algunas arquitecturas importantes.

De acuerdo con la función de transferencia utilizada en la creación de la red neuronal, ésta tendrá diferente capacidad de procesamiento y por consiguiente su aplicación será diferente, a continuación se presentan algunos topologías importantes.

Perceptrón

Rosemblatt [9] creó muchas variaciones del perceptrón. Una de las más simples es la red neuronal de una capa, en donde sus pesos y factores de corrimiento pueden entrenarse para proporcionar un vector de salida correcto, presentando un vector correspondiente de entrada. La técnica de entrenamiento usada es llamada la regla de aprendizaje del perceptrón.

Los perceptrones son especialmente convenientes para tareas simples de clasificación de patrones, pues son rápidos y confiables. En suma, un entendimiento de las operaciones del perceptrón proporciona una buena base para la comprensión de redes más complejas.

En la Fig. 2.8 se presenta una neurona de perceptrón en la cual se puede apreciar la función de transferencia (límite "fuerte").

En la Fig. 2.9 se presenta la arquitectura de una red neuronal perceptrón, la cual consiste de una capa simple de S neuronas de perceptrón, conectadas a R entradas a través de un conjunto de pesos $W_{i,j}$.

Una nota adicional con respecto a la regla de aprendizaje del perceptrón, es que, sólo es capaz de entrenar una capa. Los percetrones tienen severas limitaciones. Dentro las que se destacan: los valores de salida están limitados a 0 ó 1, debido a su función de transferencia. Los perceptrones, sólo pueden clasificar conjuntos de vectores de entrada

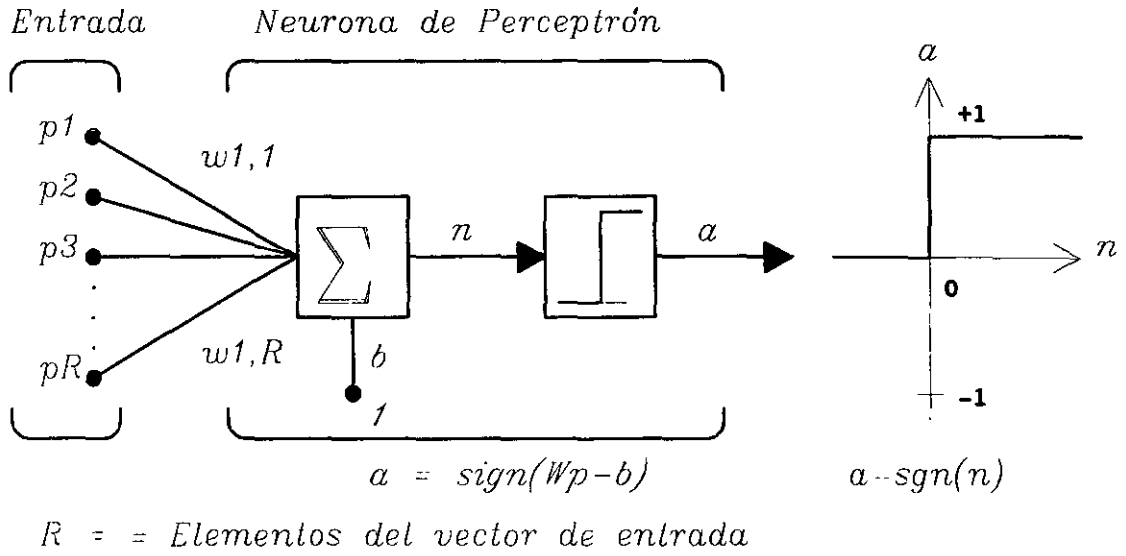

Figura 2.8: Neurona básica de perceptrón

linealmente separables (si se puede dibujar un plano o una línea recta, para separar los vectores de entrada en sus respectivas categorias, los vectores de entrada son linealmente separables). Si los vectores no son linealmente separables, el aprendizaje nunca alcanzará un punto donde todos los vectores son clasificados adecuadamente.

Redes neuronales lineales adaptables (ADALINE)

El trabajo pionero en este tipo de redes neuronales se debe a Widrow y Hoff, quienes dieron el nombre de ADALINE para elementos adaptables lineales [10].

Las redes ADALINE son similares a los perceptrones, pero con una función de transferencia lineal. Esta función de transferencia permite que la salida de la red neuronal tome cualquier valor entre 0 y 1. La red ADALINE al igual que el perceptrón sólo resuelve problemas linealmente separables, por otro lado este tipo de red utiliza la regla de aprendizaje LMS ("Least mean square"), la cual es mucho más poderosa que la regla de aprendizaje del perceptrón. La regla de aprendizaje utilizada, es la regla de aprendizaje de Widrow-Hoff [4]. Podemos decir también de esta regla de aprendizaje que sólo puede entrenar una capa de redes lineales.

En la Fig. 2.10 se presenta una red neuronal lineal, la cual tiene una capa de S neuronas conectadas a R entradas a través de una matriz de pesos W. Esta red es llamada también MADALINE por múltiple ADALINE. Este tipo de redes neuronales puede entrenarse para aprender una función afín de sus entradas, o para encontrar una aproximación lineal a una función no lineal. Una red lineal no puede, por supuesto hacer cálculo no lineal.

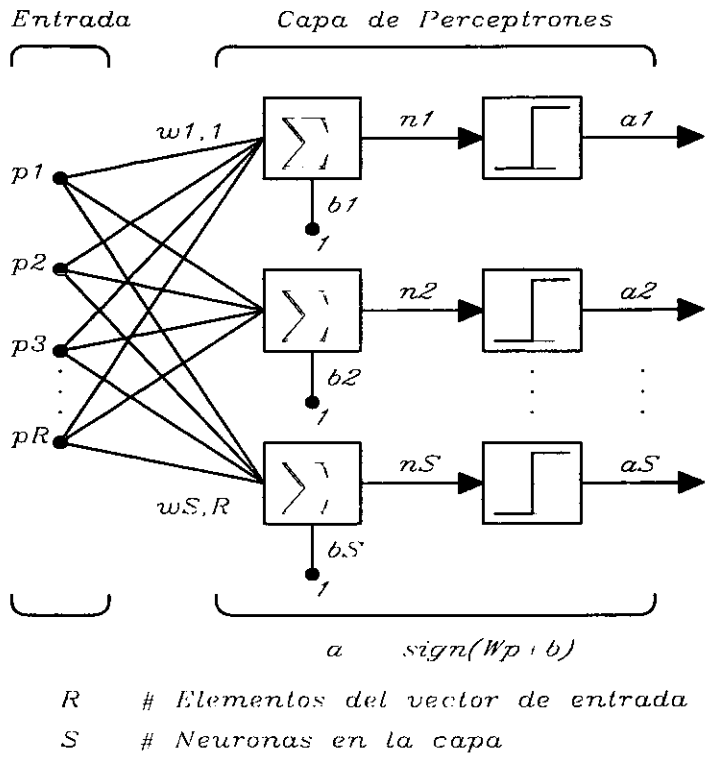

Figura 2.9: Red de una capa de perceptrones

Limitaciones de las ADALINE: estas sólo pueden aprender relaciones lineales de vectores de entrada y salida. De esta forma las ADALINEs no pueden encontrar soluciones a ciertos problemas. Por otro lado aún si la solución perfecta no existe, la ADALINE puede minimizar el error cuadrático si la velocidad de aprendizaje es suficientemente pequeña.

Retropropagación

Retropropagación fue creada por la generalización de la regla de aprendizaje de Widrow - Hoff para redes neuronales multicapa y funciones de transferencia diferenciables no lineales vectores de entrada y vectores de salida para el entrenamiento de la red neuronal, hasta que ésta pueda aproximar una función.

Las redes con factores de corrimiento, una capa sigmoidal y una capa lineal de salida son capaces de aproximar cualquier función con un número finito de discontinuidades en un conjunto compacto.

La retropropagación estándar, es un algoritmo de gradiente descendiente. El término retropropagación se refiere a la manera en la cual es calculado el gradiente para redes no lineales multicapa.

Las redes entrenadas adecuadamente por retropropagación tienden a dar respuestas razonables, cuando les son presentadas entradas que nunca antes se le han aplicado. Típica-

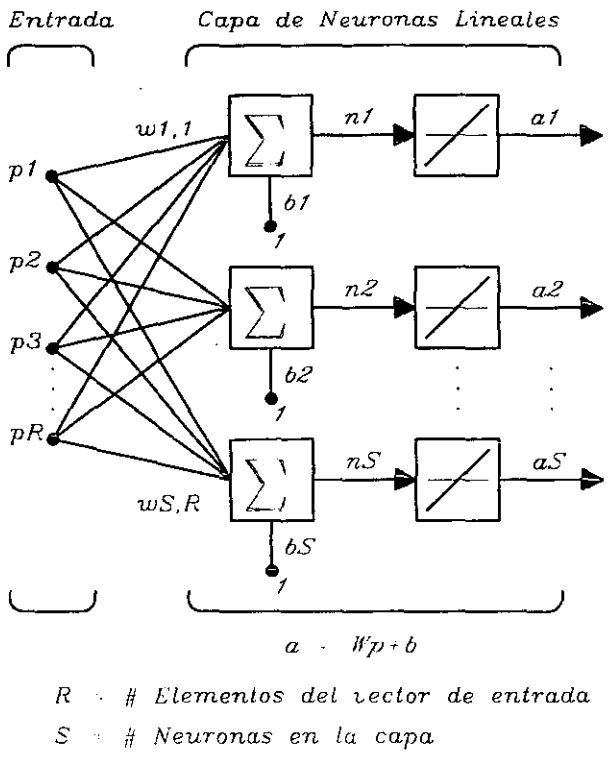

Figura 2.10: Capa de neuronas lineales

mente, una entrada nueva puede llevar a obtener una salida similar a la salida correcta para vectores usados en el entrenamiento que son similares a la nueva entrada presentada. Esta propiedad de generalización hace posible entrenar una red sobre un conjunto representativo de pares entrada salida y proporcionar un buen resultado, sin tener que entrenar la red para todos los posibles pares de entrada salida.

En la Fig. 2.11 se presenta una neurona elemental con R entradas. Cada una de las entradas es pesada con una w apropiada. La suma de las entradas pesadas y el factor de corrimiento forman la entrada a la función de transferencia f. Las neuronas pueden usar cualquier función de transferencia diferenciable f para generar su salida. También se muestran las tres funciones de transferencia más comunmente utilizadas para la retropropagación.

Red con prealimentación Una red de una capa de S neuronas con funciones de transferencia no lineales, con R entradas se muestra en la siguiente Fig. 2.12 en detalle.

Las redes neuronales con prealimentación frecuentemente tienen una o más capas ocultas de neuronas sigmoidales, seguidas por una capa de neuronas lineales. Múltiples capas de neuronas con funciones de transferencia no lineales, permiten a la red aprender relaciones

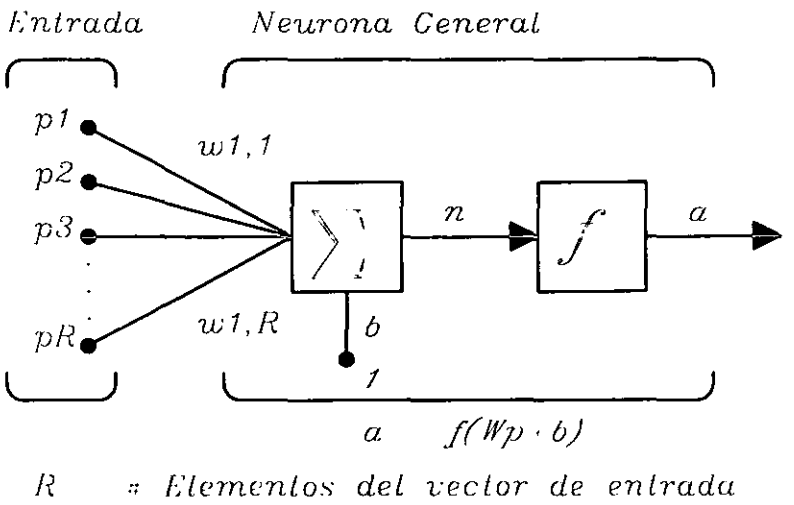

Figura 2.11: Neurona básica con vector de entrada

no lineales entre vectores de entrada y salida. La capa de neuronas de salida lineal produce valores entre el rango de -1 a +1.

Entrenamiento de redes con prealimentación con el algoritmo de retropropagación de error Existen muchas variaciones del algoritmo de retropropagación. La implementación más simple de algoritmo de aprendizaje de retropropagación, actualiza el valor de los pesos y los factores de corrimiento en la dirección en la cual la función de desempeño decrece más rápidamente el negativo del gradiente. Una iteración de este algoritmo se puede escribir como:

$$X_{k+1} = X_k - \alpha_k g_k$$

Donde X_k es un vector de valores actuales y factores de corrimiento, g_k es el gradiente y α_k es la velocidad de entrenamiento.

Existen dos formas diferentes en las cuales el algoritmo de gradiente descendiente se puede implementar [4]: modo incremental y modo por lote. En el modo incremental, el gradiente se calcula y los pesos se actualizan después de que cada entrada es aplicada a la red. En el modo por lote todas la entradas son aplicadas a la red antes de que los pesos sean actualizados.

El algoritmo de gradiente descendiente es generalmente muy lento, debido a que este requiere velocidades de entrenamiento muy pequeñas para aprendizaje estable. Redes multicapa son capaces de realizar aproximación de funciones no lineales, con un error de aproximación arbitrariamente pequeño, en un conjunto compacto.

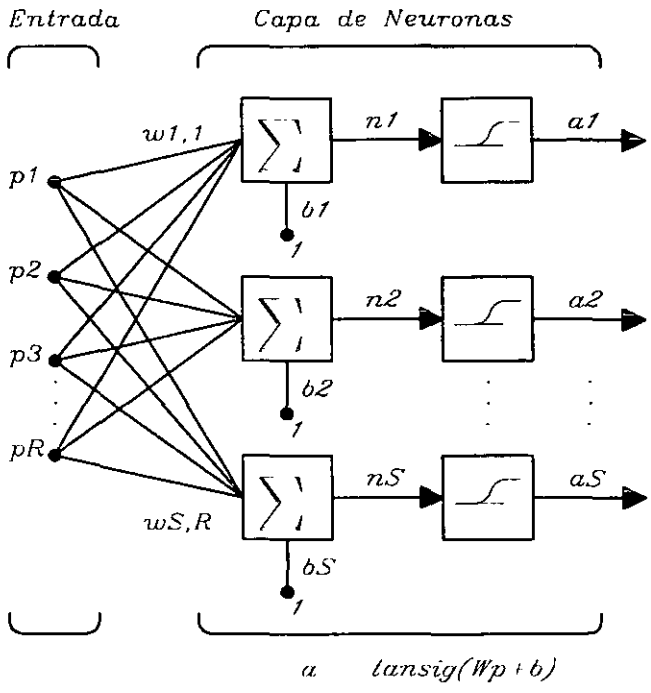

Figura 2.12: Red de neuronas no lineales

Redes neuronales "Mapas auto-organizables"

La auto-organización en redes es uno de los temas más importantes en el campo de las redes neuronales. Tales redes pueden aprender a detectar regularidades y correlaciones en sus entradas y adaptar sus futuras respuestas, de acuerdo a estas entradas. Las neuronas de redes competitivas aprenden a reconocer grupos de vectores de entrada similares.

Aprendizaje competitivo.

Las neuronas en una capa competitiva se distribuyen para reconocer vectores de entrada frecuentemente presentados (la neurona que con el vector de entrada tenga una distancia más pequena, resulta la ganadora y se le asigna un valor de uno, todas las demás se hacen cero).

En la Fig. 2.13 se presenta la arquitectura de una red competitiva.

El rectángulo $\| \; ndist \; \|$ en esta figura acepta el vector de entrada p y la matriz de pesos de entrada $IW^{1,1}$ y produce un vector de S^1 elementos. Los elementos son las distancias negativas entre el vector de entrada y el vector $IW^{1,1}$ formado por los renglones de la matriz de pesos de entrada.

El nodo de entrada n^1 de una capa competitiva se calcula encontrando la distancia negativa entre el vector de entrada p, el vector de pesos y sumando los factores de corrimiento

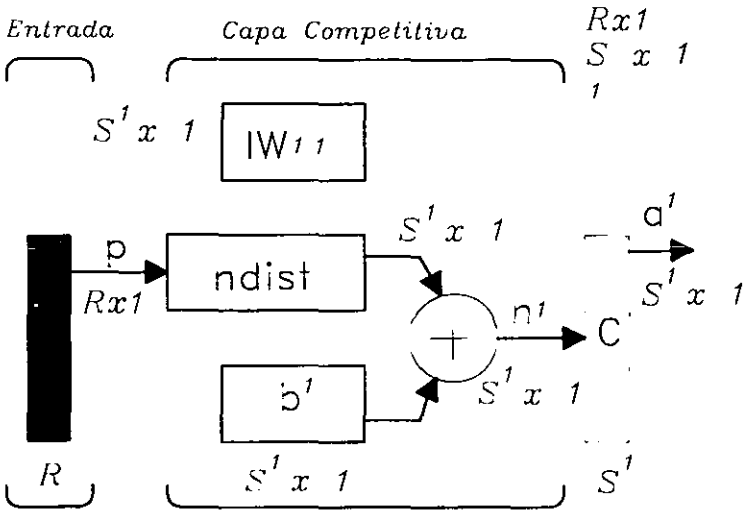

Figura 2.13: Red neuronal mapa auto-organizable

b. Si todos los factores de corrimiento son ceros, el máximo nodo de entrada que puede tener una neurona es cero. Esto ocurre cuando el vector de entrada p iguala el vector de pesos de la neurona.

La función de tranferencia competitiva acepta un vector de entrada para una capa y regresa la salida de la neurona de cero para todas las neuronas excepto para la ganadora, es decir la neurona asociada con el elemento más positivo de la entrada n^1. La salida ganadora es uno. Si todos los factores de corrimiento son ceros entonces la neurona cuyo vector de pesos es más cercana al vector de entrada tiene la entrada menos negativa, y por lo tanto gana la competencia y su salida será uno.

2.7 Entrenamiento

Una característica que distingue a las redes neuronales de otras técnicas de aproximación de funciones, es la entrenabilidad o la adaptabilidad. Esta característica permite a las redes neuronales realizar un mapeo de datos de entrada-salida, ajustando en este proceso sus pesos sinápticos. La arquitectura de red seleccionada en el presente trabajo es la de retropropagación. por lo tanto, a continuación se analizan las dos formas de entrenamiento básicas para este tipo de redes, cabe mencionar que la red utilizada en el presente trabajo, también será estática.

2.7.1 Entrenamiento fuera de línea

Este método de entrenamiento se utiliza tanto para redes neuronales estáticas como dinámicas. Para este método de entrenamiento de redes neuronales, los pesos sinápticos y los factores de corrimiento son actualizados sólo después que se ha presentado todo el lote de pares de entrenamiento. A este método de entrenamiento también se le conoce como entrenamiento por lote, para aplicar dicho método es necesario conocer un lote de pares entrada-salida.

Este método de entrenamiento es el que se utilizará en el presente trabajo

2.7.2 Entrenamiento en línea

Este tipo de entrenamiento se utiliza tanto para redes neuronales estáticas o dinámicas, aunque es más comúnmente utilizado en redes neuronales dinámicas tales como filtros adaptables. En este tipo de entrenamiento los pesos y los factores de corrimiento son actualizados cada vez que se presenta un dato de entrada a la red, cuando el sistema está en operación, es decir, no se conoce el correspondiente dato de salida. A este tipo de entrenamiento también se le conoce como, entrenamiento no supervisado, puesto que para éste, no se necesita contar con un lote de datos (pares de entrada salida).

Capítulo 3

Control de manipuladores robóticos

3.1 Ecuaciones dinámicas de un robot

Las ecuaciones de movimiento de un manipulador son un conjunto de ecuaciones matemáticas que describen su conducta dinámica. Tales ecuaciones son útiles para la simulación por computadora del movimiento del robot, el diseño de ecuaciones de control y la evaluación del diseño de la estructura del brazo.

En general, el rendimiento dinámico depende directamente de la eficacia de los algoritmos de control y de su modelo dinámico. El problema de control consiste en obtener modelos dinámicos del brazo robot y a continuación especificar leyes de control para conseguir la respuesta y rendimiento del sistema deseado.

El modelo dinámico de un robot se puede obtener a partir de leyes físicas conocidas, tales como las leyes de la mecánica clásica (de Newton) y la mecánica lagrangiana. Esto conduce al desarrollo de ecuaciones de movimiento dinámico para las diversas articulaciones del manipulador en términos de los parámetros geométricos e inerciales de los elementos. Métodos convencionales como las formulaciones de Euler-Lagrange (E-L) y Newton-Euler (N-E) se pueden aplicar sistemáticamente para desarrollar las ecuaciones dinámicas del robot. Estas ecuaciones de movimiento son equivalentes unas a otras en el sentido de que describen la conducta dinámica del mismo robot, sus estructuras pueden diferir porque se obtienen por diversas razones y objetivos. Algunas se obtienen para reducir el tiempo de cálculo, otras se obtienen para facilitar el análisis y diseño de la ley de control y otras se obtienen para mejorar la simulación en computadora.

La obtención del modelo dinámico de un manipulador basado en la formulación de Euler-Lagrange es simple y sistemática. Suponiendo el movimiento de cuerpo rígido, las ecuaciones de movimiento resultante, excluyendo la dinámica de los dispositivos de control electrónico y la fricción de los engranes, son un conjunto de ecuaciones diferenciales no lineales acopladas y de segundo orden.

Las ecuaciones de E-L proporcionan ecuaciones de estado explícitas para la dinámica del robot y se pueden utilizar para analizar y diseñar estrategias de control avanzadas en el

espacio de las variables de articulación. En menor medida se utilizan para resolver el problema *dinámico directo*, esto es, dadas las fuerzas/pares deseadas, se utilizan las ecuaciones dinámicas para resolver las aceleraciones de las articulaciones, que se integran para obtener las velocidades y las coordenadas generalizadas; o para el problema *dinámico inverso*, esto es, dadas las coordenadas generalizadas deseadas y sus primeras derivadas respecto del tiempo, se calculan las fuerzas/pares generalizados. En ambos casos es necesario calcular los coeficientes de la matriz simétrica de inercia y los coeficientes del vector de fuerzas de Coriolis y centrífuga. Desgraciadamente, el cálculo de estos parámetros requiere de un gran cantidad de operaciones aritméticas. Por tanto se considera que las ecuaciones de E-L son muy dificiles de utilizar en aplicaciones de control en tiempo real al menos que se simplifiquen.

Como alternativa para derivar ecuaciones de movimiento más eficientes existen algoritmos para calcular las fuerzas/pares generalizados basados en las ecuaciones de N-E (Newton-Euler) [11]. Las ecuaciones dinámicas resultantes, excluyendo la dinámica del dispositivo de control, juego y rozamiento de los engranes son un conjunto de ecuaciones recursivas hacia adelante y hacia atrás. Este conjunto de ecuaciones se puede aplicar secuencialmente a los elementos del robot. El resultado más significativo de esta formulación, es que el tiempo de cálculo de las fuerzas/pares generalizados se encuentra que es linealmente proporcional al número de articulaciones del manipulador e independiente de la configuración del mismo. Con este algoritmo se puede realizar el control en tiempo real en el espacio de las variables de las articulaciones. Cabe hacer una observación importante en este momento y es que, las ecuaciones recursivas destruyen la "estructura" del modelo dinámico y éste es indispensable para darnos comprensión para el diseño del controlador en el espacio de estados.

$$\tau(t) = D(q(t))\ddot{q}(t) + C(q(t), \dot{q}(t))\dot{q}(t) + g(q(t)) \tag{3.1}$$

donde

$\tau(t)$ = Vector de par generalizado de n x *1* aplicado en las articulaciones.

$q(t)$ = Vector de variables de articulación del brazo de n x *1*.

$\dot{q}(t)$ = Vector de velocidades de articulación de n x *1*.

$\ddot{q}(t)$ = Vector de aceleraciones de articulación de n x*1*.

$D(q)$ = Una matriz simétrica inercial de n x n relacionada con la aceleración.Con elementos

$$D_{kj} = \Sigma_j^n d_{kj}(q)\ddot{q}_j$$

para $i, k = 1, 2, ..., n$

$C(q(t).\dot{q}(t))\dot{q}(t)$ = Vector de fuerzas de Coriolis y centrifuga no lineal de *nx1*. cuyos elementos son

$$C = \Sigma_{i,j}^n c_{ijk}(q)\dot{q}_i\dot{q}_j$$

$$c_{ijk} = \frac{1}{2}\left\{\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\right\}$$

para i, k, $m = 1, 2, ..., n$
g(q) = Un vector de la fuerza de la carga gravitatoria de n x 1, con elementos

$$g_i = \frac{\partial V}{\partial q_k}$$

$$V = g^T r_c m$$

m es la masa del elemento, r_c es la distancia de la articulación al centro de masa y g es la constante de gravitación.

De esta ecuación se puede decir:

1. Los coeficientes g_i representan la carga gravitatoria debido a los elementos.

2. Los coeficientes D_{kj} están relacionados con la aceleración de las variables de articulación. En particular, para $j=k$ D_{jj} está relacionado con la aceleración de la articulación i donde el par motor τ_i actúa, mientras que para j$\neq k$, D_{kj} está relacionado al par de reacción inducido por la aceleración de la articulación k y actuando en la articulación i o viceversa. Como la matriz de inercia es simétrica y $Tr(A) = Tr(A^T)$, se puede demostrar que $D_{jk} = D_{kj}$.

3. Los coeficientes c_{ijk} están relacionados con la velocidad de la variables de las articulaciones. Los dos últimos índices, km, cuya interelación dinámica induce un par (o fuerza) de reacción en la articulación i. El primer índice se relaciona siempre con la articulación donde se "sienten" las fuerzas de reacción inducidas por la velocidad.

La ecuación del manipulador, tal como se mencionó anteriormente, es un conjunto de ecuaciones diferenciales ordinarias no lineales y de segundo orden. A causa de su estructura matricial, las ecuaciones de E-L son interesantes desde el punto de vista de control en lazo cerrado, porque dan un conjunto de ecuaciones de estado. Esta estructura permite el diseño de una ley de control que compensa fácilmente todos los efectos no lineales.

Propiedades del modelo dinámico

El modelo dinámico del robot, presentado anteriormente, tiene las siguientes propiedades:

1. La matriz de inercia $D(q(t))$ es simétrica y positiva definida, la cual verifica

$$\lambda_m I \leq D(q(t)) \leq \lambda_M I$$

donde $\lambda_m, \lambda_M < \infty$ son los valores propios de la matriz $D(q(t))$ estrictamente positivos, mínimo máximo, respectivamente para todas las configuraciones de q.

2. La matriz $N(q(t), \dot{q}(t)) = \dot{D}(q(t)) - 2C(q(t), \dot{q}(t))$ es antisimétrica para una matriz $C(q(t), \dot{q}(t))$ particular (lo que siempre es posible), esto es.,

$$z^T N(q(t), \dot{q}(t)) z = 0$$

para cualquier vector z de dimensión n x *1*.

3. El vector de torques de entrada τ se puede escribir como:

$$\tau = Y(q(t), \dot{q}(t), \ddot{q}(t))\rho$$

donde $Y(q(t), \dot{q}(t), \ddot{q}(t))$ es una matriz de *(n x r)* y ρ es un vector de parámetros dinámicos de base de dimensión *(r x 1)*.

4. Dados cualquiera dos vectores x e y de dimensión *(n x 1)*, entonces

$$C(q(t), x)y = C(q(t), y)x$$

5. Los términos de Coriolis y centrífugos $C(q(t), \dot{q}(t))\dot{q}(t)$ verifican

$$\| C(q(t), \dot{q}(t))\dot{q}(t) \| \leq c_o \| \dot{q}(t) \|^2$$

para alguna cota constante $c_o > 0$.

6. La matriz $C(q(t), \dot{q}(t))$ verifica

$$\| C(q(t), \dot{q}(t)) \| \leq k_c \| \dot{q}(t) \|$$

para alguna cota constante $k_c > 0$.

7. El vector de gravedad g$(q(t))$ satisface

$$\| g(q(t)) \| \leq g_o$$

para alguna cota constante $g_o > 0$.

Observación 3.1 *El acotamiento de las entradas de la matriz de inercia, esto es, $\lambda_M < \infty$, así como también el acotamiento de la norma del vector de gravedad se cumple sólo para manipuladores de juntas de revolución.*

Observación 3.2 *Se puede probar que la segunda propiedad es esencial en el desarrollo de cierta clase de algoritmos de control y está íntimamente relacionado a las propiedades de pasividad del manipulador.*

Observación 3.3 *La medición de $\dot{q}(t)$ se puede relajar vía la introducción de observadores no lineales en el lazo de control.*

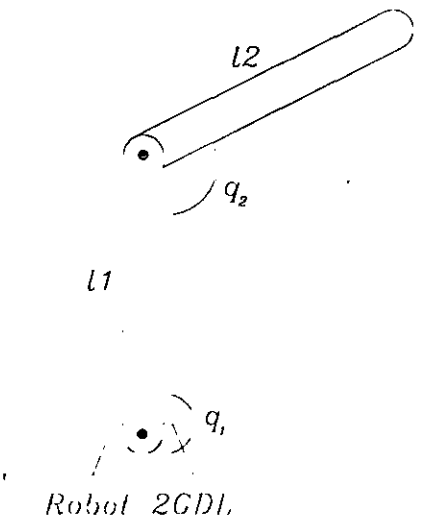

Figura 3.1: Manipulador de dos grados de libertad

3.1.1 Ejemplo de ecuación dinámica de un manipulador de dos elementos

En la Fig.3.1 se presenta un esquema del manipulador de dos elementos giratorios. .

Enseguida se presenta la ecuación dinámica de un manipulador de dos elementos giratorios.

$$\tau(t) = D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) \tag{3.2}$$

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{3}m_1l^2 + \frac{4}{3}m_2l^2 + m_2C_2l^2 & \frac{1}{3}m_2l^2 + \frac{1}{2}m_2l^2C_2 \\ \frac{1}{3}m_2l^2 + \frac{1}{2}m_2l^2C_2 & \frac{1}{3}m_2l^2 \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix}$$

$$+ \begin{bmatrix} -\frac{1}{2}m_2S_2l^2\dot{q}_2 - m_2S_2l_1^2\dot{q}_1\dot{q}_2 \\ \frac{1}{2}m_2S_2l^2\dot{q}_1^2 \end{bmatrix} + \begin{bmatrix} \frac{1}{2}m_1glC_1 + \frac{1}{2}m_2glC_{12} + m_2glC_1 \\ \frac{1}{2}m_2glC_{12} \end{bmatrix}$$

donde el subíndice se refiere al elemento, la ecuación está en función de los ángulos de las articulaciones, la longitud de los elementos es la misma y los términos C_i, C_{ij}, S_i, S_{ij}, se refieren a las funciones coseno y seno de los ángulos de articulación.

El modelo del robot analizado se obtiene de la formulación de Euler-Lagrange y tiene la siguiente forma

$$\tau(t) = D(q(t))\ddot{q}(t) + C(\dot{q}(t).q(t))\dot{q} + g(q(t)) \tag{3.3}$$

3.2 Tipos de controladores

Controladores del tipo PID (Proporcional Derivativos e Integrales), y PD más compensación, son diseñados para resolver el problema de regulación y seguimiento, respectivamente. Estos tienen la ventaja de no requerir de conocimiento ni de la estructura del modelo ni de los parámetros del mismo.

3.2.1 Control PD

Un controlador PD tiene la siguiente forma básica:

$$\tau = K_P(q_d - q(t)) - K_D\dot{q}(t) \tag{3.4}$$

donde q_d y q denotan las variables de junta, la deseada y la actual respectivamente, K_P y K_D son matrices constantes y positivas definidas, de dimensiones convenientes.

Considere el problema de regulación, con q_d constante e introduzcamos el vector de error, $\tilde{q} = q - q_d$, con$\dot{\tilde{q}} = \dot{q}$ y $\ddot{\tilde{q}} = \ddot{q}$, con esta definición, aplicamos la ley de control a la ecuación dinámica del robot, esto es,

$$D(q(t))\ddot{\tilde{q}}(t) + C(\dot{\tilde{q}}(t), q(t))\dot{\tilde{q}}(t) + K_P\tilde{q}(t) + K_D\dot{\tilde{q}}(t) + g(q(t)) = 0 \tag{3.5}$$

Observe que el vector de gravedad se da como una función del estado real y no como una función del error de regulación. Para poder expresar la ecuación de error anterior en forma autónoma, podemos expresar g$(q(t))$ como $g(\tilde{q}(t)+q(t))$ y usando relaciones trigonométricas para eliminar la dependencia sobre el vector constante q_d en c. Estamos ahora en posición de determinar el posible punto de equilibrio, o conjunto de equilibrio. Considere que las condiciones iniciales son: $\ddot{\tilde{q}}(0) = 0$, $\dot{\tilde{q}}(0) = 0$, $\tilde{q}(0) = \tilde{q}_0$. También $\tilde{q}_0$ y $\tilde{q}_d$ satisfacen las siguiente relaciones

$$K_P\tilde{q}_0 + g(\tilde{q}_0 + q_d) = 0$$

y entonces no es posible el movimiento, debido a que $\ddot{\tilde{q}}(0) = \dot{\tilde{q}}(0) = 0$ para todo tiempo. De acuerdo a la definición de conjuntos invariantes, el conjunto

$$S = \left\{(\tilde{q}, \dot{\tilde{q}}) : K_P\tilde{q} + g(\tilde{q}_0 + q_d) = 0,\ \dot{\tilde{q}} = 0\right\}$$

describe el conjunto de todos los posibles puntos de equilibrio, para la ecuación de error.

Generalizando ahora el análisis de estabilidad para completar la dinámica del manipulador, condiderando un punto de equilibrio

$$\xi^* = (q^*, \dot{q}^*)^T \in S$$

Considere la función candidata de Lyapunov

$$V = \frac{1}{2}\dot{\tilde{q}}^T D(q)\dot{\tilde{q}} + \tilde{q}^T K_P\tilde{q} + U(q) + U_o$$

donde $U(q)$ representa la energía potencial y U_o es una constante conveniente tal que V satisface los requerimientos de la función candidata de Lyapunov (en particular $V(\xi^*) = 0$). Derivando la función candidata de Lyapunov y evaluando en las trayectorias de la ecuación de error, se tiene [17]

$$\dot{V} = \tilde{q}^T K_D \dot{\tilde{q}} + \tilde{q}^T (\frac{1}{2}\dot{D}(q) - C(\dot{q}(t), q(t))\dot{\tilde{q}} - \tilde{q}^T g(q) + \tilde{q}^T (\frac{\partial U(q)}{\partial q})^T \leq -\lambda_{\min}(K_D) \parallel \dot{\tilde{q}} \parallel^2$$

donde $\lambda_{\min}(K_D)$ denota el valor característico más pequeño de la matriz K_D, para obtener este resultado, se utilizó la propiedad 2 de la ecuación dinámica del robot además del hecho que $(\frac{\partial U(q)}{\partial q})^T = g(q)$. Entonces el análisis anterior establece que $(\tilde{q},\ \dot{\tilde{q}})$ son estables en el sentido de Lyapunov. Se puede aplicar el Teorema de La Salle para concluir que el sistema, al menos localmente, converge a S con S definido como en (conjunto invariante). El siguiente Lema resume este resultado [17]

Teorema 3.1 *Considere un controlador PD (3.9) aplicado a la dinámica del manipulador (3.1); entonces la ecuación de lazo cerrado tiene un punto de equilibrio si K_P es suficientemente grande y varios puntos de equilibrio si K_P es suficientemente pequeña. El equilibrio está descrito por un conjunto invariante*

$$S = \left\{ (\tilde{q},\ \dot{\tilde{q}}) : K_P \tilde{q} + c(\tilde{q}_0 + q_d) = 0,\ \dot{\tilde{q}} = 0 \right\}$$

Además si K_P y K_D son matrices positivo definidas, entonces todas las soluciones $\xi(t)$ convergen asintótica y globalmente (si K_P es grande) o localmente (si K_P es pequeña) - a S cuando $t \rightarrow \infty$

El error en estado estable puede de esta forma, en principio, ser arbitrariamente reducido, incrementando K_P ; a pesar de esto, el ruido de medición y la dinámica no modelada limitará el uso práctico de altas ganancias.

3.2.2 Control PD más compensación

La estrategía de control utilizada en el presente trabajo es la técnica de control descentralizado [19].

El método de control descentralizado se desarrolló basado en los sistemas mecánicos no lineales interconectados el controlador resultante es extremadamente simple. Para la aplicación de esta técnica de control no es necesario conocer el modelo del robot, por lo que resulta muy conveniente para la presente aplicación.

La ley de control es la siguiente:

$$\tau = -Ks + w \tag{3.6}$$

donde:

$$s = [s_1 \; s_2 \; , \ldots \; s_n]^T \tag{3.7}$$

$$s \triangleq \begin{bmatrix} \dot{q}_1 - \dot{q}_1^d \\ \dot{q}_2 - \dot{q}_2^d \end{bmatrix} + \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} q_1 - q_1^d \\ q_2 - q_2^d \end{bmatrix} \tag{3.8}$$

$$w_1 = -(\delta S_1)^2 \frac{s_1}{\delta S_1 \mid s_1 \mid + \epsilon_1} \tag{3.9}$$

$$w_2 = -(\delta S_2)^2 \frac{s_2}{\delta S_2 \mid s_2 \mid + \epsilon_2} \tag{3.10}$$

$$S_1 = 1 + \mid s_1 \mid + \mid s_1 \mid^2 \tag{3.11}$$

$$S_2 = 1 + \mid s_2 \mid + \mid s_2 \mid^2 \tag{3.12}$$

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} K_1 & 0 \\ 0 & K_2 \end{bmatrix} \left[\begin{bmatrix} \dot{q}_1 - \dot{q}_1^d \\ \dot{q}_2 - \dot{q}_2^d \end{bmatrix} + \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} q_1 - q_1^d \\ q_2 - q_2^d \end{bmatrix} \right] + \begin{bmatrix} -(\delta S_1)^2 \frac{s_1}{\delta S_1 \mid s_1 \mid + \epsilon_1} \\ -(\delta S_2)^2 \frac{s_2}{\delta S_2 \mid s_2 \mid + \epsilon_2} \end{bmatrix} \tag{3.13}$$

$K_1, \; K_2 \; > 0$ forman la matriz K de ganancia constante, $\lambda_1, \; \lambda_2 > 0$ forman la matriz λ de ganancia constante y $\delta, \; \epsilon_i > 0, \; i = 1, 2;$ son parámetros de diseño. Como un resultado importante de la aplicación del esquema de control utilizado [19], que garantiza la estabilidad del sistema se presenta el siguiente teorema.[19]

Teorema 3.2 *Considere el controlador descentralizado robusto 6.3 aplicado a la planta 3.3. El sistema de lazo cerrado resultante es globalmente estable en el sentido que todas las señales son globalmente acotadas y el error de seguimiento $\tilde{q}(t)$ converge al conjunto residual*

$$B(0, r) \triangleq \{\tilde{q} \in \mathcal{R}^n \mid\mid \tilde{q} \mid\mid \leq r\}$$

donde $\mid\mid \; . \; \mid\mid$ denota la norma Euclidiana y para una $h > 0$ arbitraria,

$$r^2 \triangleq m_r \frac{\epsilon}{\min_i \{\lambda_i\}^2 \min_i \{k_i\}} + h, \quad m_r \triangleq \frac{\bar{m}}{\underline{m}}, \quad \epsilon \triangleq \Sigma_{k=1}^n \epsilon_k$$

Capítulo 4

Retroalimentación visual

4.1 Visión artificial

Al igual que sucede en el ser humano, la capacidad de visión dota al robot con un sofisticado mecanismo de percepción que permite a la máquina responder a su entorno de una forma inteligente y flexible.

Así como la percepción de la proximidad de contacto juega un papel significativo en el perfeccionamiento del comportamiento del robot, la visión se considera como la más potente capacidad sensorial de un robot. Como es de suponer, los sensores, conceptos y "hardware" de proceso asociados con la visión artificial son considerablemente más complejos, que los asociados con cualquier otra capacidad robótica.

La visión artificial puede ser definida como el conjunto de procesos de obtención, caracterización e interpretación de información de imágenes tomadas de un mundo tridimensional. Estos procesos, también habitualmente llamados *visión por computadora* pueden a su vez ser subdivididos en seis áreas principales: 1). captación, 2) preprocesamiento, 3) segmentación, 4) descripción, 5) reconocimiento, 6) interpretación. La captación es el proceso a tráves del cual se obtiene una imagen visual. El preprocesamiento incluye técnicas tales como la reducción de ruido y realce de detalles. La segmentación es el proceso que divide una imagen en objetos que sean de nuestro interés. Mediante los procesos de descripción se obtienen características (por ejemplo, tamaño, forma) convenientes para diferenciar un tipo de objeto de otro. El reconocimiento es el proceso que identifica estos objetos. Finalmente, la interpretación le asocia un significado a un conjunto de objetos reconocidos.

Es conveniente agrupar estas diversas áreas de acuerdo con la complicación y delicadeza que lleva aparejada su implementación. Se pueden considerar entonces, tres niveles de procesamiento: visión de bajo, medio y de alto nivel. Así se asocia a la visión de bajo nivel aquellos procesos que son primarios en el sentido de que pueden ser considerados "reacciones automáticas" sin requerir ningún tipo de inteligencia, en esta categoria se pueden incluir la captación y el preprocesamiento. En la visión de nivel medio se pueden incluir la segmentación, la descripción y el reconocimiento de objetos individuales. La visión de alto nivel se refiere a procesamientos que tratan de emular la cognición.

Las categorías y subdivisiones antes presentadas están basadas en la forma en que son implementados la gran mayoria de los sistemas de visión artificial. Esto no implica que estas subdivisiones sirvan de modelo para la visión humana ni que puedan llevarse a cabo unas independientes de otras.

Aunque la visión real es una actividad inherentemente tridimensional, la mayoría el trabajo de visión artificial se lleva a cabo usando imágenes de una vista tridimensional, obteniéndose la información relativa a la profundidad, de técnicas de procesamiento de imagen especiales, tales como la iluminación estructurada o el uso de la información de imágenes estéreo.

4.1.1 Adquisición de imágenes

La información visual se convierte en señales eléctricas por los sensores visuales. Cuando estas señales eléctricas se muestrean espacialmente y se cuantifican en amplitud, obtendremos de ellas una *imagen digital.*

Cámaras de video

Los principales dispositivos usados para la visión artificial son las cámaras de televisión, que están compuestas por un sensor basado en un dispositivo de estado sólido o válvulas electrónicas y electrónica asociada. Los sensores de estado sólido son representados por los dispositivos de acoplamiento de carga (CCD). Los dispositivos de tratamiento de imagen de estado sólido nos ofrecen un gran número de ventajas sobre las camáras de tubos, como son, su menor peso, menor tamaño, más larga vida y menor consumo de potencia. No obstante, la resolución de algunos tubos está todavía por encima de las capacidades de las camáras de dispositivos de estado sólido.

Dispositivos CCD

En el campo de los dispositivos CCD, se puede dividir los sensores en dos categorias: sensores de exploración de líneas y sensores de exploración de área. El componente básico de un sensor CCD de exploración de línea es una fila de elementos de silicio llamados photosites. Los fotones de la imagen pasan a través de una estructura transparente policristalina de silicio y son absorbidos en el cristal de silicio, creando así pares electrón-hueco. Los fotoelectrones que así se obtienen son recogidos por los photosites, siendo la cantidad de carga acumulada en cada photosite proporcional a la intensidad lumínica en ese punto.

Un sensor de exploración de línea típico está compuesto por una fila de elementos de imagen, dos puertos de transferencia usados para registrar los contenidos de los registros de transporte a un amplificador cuya señal de salida es una señal de tensión proporcional a los contenidos de la fila de photosites.

Los sensores de área son similares a los de exploración de línea, con la diferencia de que los photosites están organizados en forma de matriz y existe un conjunto de registros de

transporte y puertos para cada columna de photosites. Los contenidos de los photosites impares son secuencialmente transferidos a los registros verticales de transporte y posteriormente al registro horizontal de transporte. El contenido de este registro es enviado a un amplificador cuya salida es una línea de video. Repitiendo este procedimiento para las líneas pares completamos el segundo campo de un cuadro de televisión. Este "mecanismo de exploración" se repite 30 veces por segundo.

De las cámaras de exploración de línea se obtendrá obviamente una línea por cada imagen de entrada. Estos dispositivos se ajustan muy bien para aplicaciones en las cuales los objetos se mueven enfrentándose al sensor (como en las cintas transportadoras). El movimiento de un objeto en la dirección perpendicular al sensor produce una imagen bidimensional. La resolución de los sensores de exploración de líneas normales oscilan entre los 256 y los 2048 elementos. La resolución de los sensores de área varía entre 32 x 32 elementos para un sensor de baja resolución y 256 x 256 elementos para un sensor de resolución media. Los dispositivos de resolución más alta actualmente en el mercado incluyen del orden de 480 x 480 elementos y los sensores CCD experimentales son capaces de obtener resoluciones de 1024 x 1024 elementos. La salida de la cámara es el resultado de leer secuencialmente la carga de las celdas elementales explorando todos estos y los voltajes de carga son convertidos a señales analógicas (existen alguna cámaras cuyas salidas son señales digitales en forma directa), codificadas por algún formato estándar como CCIR, RS-170 o en el caso de color NTSC PAL. La velocidad de la toma está limitada por el lento desempeño de los registros de corrimiento y amplificadores de salida..

De acuerdo con el teoría de muestreo de Nyquist, una cámara CCD captura un arreglo discreto de imágenes, éstas pueden distinguir sólo cierto grado de detalle en la escena observada. Como la señal analógica de entrada es remuestrada, existe pérdida de información.

4.1.2 Problemas en visión robótica

Para dotar de la capacidad de visión a los robots existen varios métodos, la selección de alguno de estos no es una tarea simple, pues la selección del método no sólo afecta la salida de un paso del proceso, sino que afecta todos los pasos subsecuentes del procedimiento, en la Fig. 4.1 se presenta un sistema típico de visión de máquina.

El proceso inicia cuando la iluminación es reflejada de las superficie del objeto. En este punto los rayos de luz entran al sensor (que puede ser una cámara de video) como una intensidad de luz bidimensional. Esta es función de dos factores, la luminancia de la fuente de luz y la reflectancia de los objetos observados. La salida del sensor es una señal analógica, la cual puede ser muestreada y digitalizada. Después de crear la imagen digital, puede ser apropiado un proceso de filtrado de los datos obtenidos, para filtrar posible ruido. La imagen limpia resultante está lista para la búsqueda del objeto, por algún algoritmo de búsqueda. Después de esto se implementa un procedimiento de retroalimentación y reconocimiento. Todos los pasos mencionados pueden realizarse en formas diferentes.

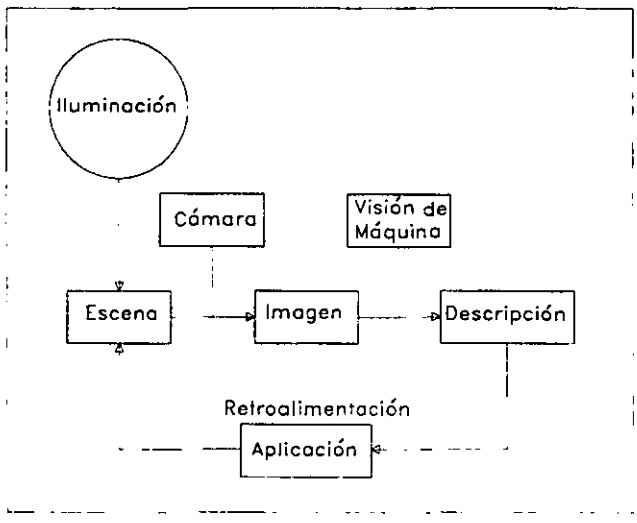

Figura 4.1: Visión de máquina

Digitalización y muestreo

La señal analógica tiene que transformarse en su equivalente forma digital de tal manera que ésta pueda ser procesada en una computadora digital.

Procesamiento de imágenes digitales

Este proceso es realizado sobre la salida del digitalizador, éste puede ser realizado en circuitería("hardware"), programación ("software") o una combinación de ambas.

La importancia de este paso es considerar el hecho de que en éste se puede remover el ruido introducido a la imagen por el sensor, digitalizador y los canales de transmisión.

Histogramas

Los histogramas de niveles de gris son funciónes que dan la frecuencia de ocurrencia de cada uno de los niveles de gris en la imagen. Estos pueden definirse como una función que muestra para cada uno de los niveles de gris, el número de pixels en la imagen que tienen este nivel de gris [14]. Esta frecuencia de ocurrencia es graficada en función del nivel de gris de los pixels. La importancia de la obtención de éste histograma de imagen, después de la digitalización de la foto, es determinar si la imagen fué adecuadamente escalada dentro del rango disponible de niveles de gris [14].

Adopción de umbrales

La adopción de umbrales de forma automática usa la función de densidad de probabilidad como un índice estadístico de las propiedades de la imagen. El problema es encontrar el punto exacto de niveles de gris de umbral que separa el objeto del fondo

de la foto. De esta forma pues, el proceso de adopción de umbral se refiere básicamente a separar el objeto de interés de todos los demás objetos de la foto.

Orillas y detección de frontera

Este es el método por medio del cual se reduce la imagen digital a un objeto de forma simple. Una orilla local es una pequeña área de la imagen donde los niveles de gris cambian drásticamente. La detección de frontera es quizá la unión más importante entre los datos y la interpretación de la imagen. Los dos métodos generales de detección son: algoritmo de detección de alto conocimiento y el algoritmo de detección de bajo conocimiento, la selección de alguno de éstos depende del conocimiento disponible de la forma general de la imagen.

Métodos de búsqueda

Este procedimiento es la comparación de la imagen encontrada a la base de datos pre-existente. El resultado esperado de este procedimiento es el reconocimiento de la imagen detectada, comparando ésta con imágenes conocidas. Este procedimiento debe ser extremadamente rápido debido al tiempo requerido para comparar la imágenes.

Lugar y ambiente

El lugar donde se instalará el sistema y bajo que restricciones ambientales opera juegan un papel muy importante en el ajuste del ancho y contraste de las fronteras legibles de la imagen. También esto puede determinar la inclusión de ruido y velocidad de una operación óptima, especialmente cuando se trata de ambientes críticos.

Otras consideraciones

El otro criterio a considerar es el tipo de cámara usada y su exactitud, también si los objetos son móviles o estacionarios, tipo de digitalizador y efectos de ruido.

4.2 Realización de la retroalimentación visual

De acuerdo con el apartado anterior, ante la complejidad inherente en la visión de máquina se justifica la aplicación de las redes neuronales como medio de realizar un mapeo entre las coordenadas de imagen y la posición de efector final del robot, o entre las coordenadas de imagen y los ángulos de las articulaciones del robot.

Este mapeo para retroalimentación visual se puede hacer básicamente de dos formas:

Visión estéreo

En este método se utilizan dos cámaras que forman una imagen estéreo del espacio de trabajo, por lo tanto para el entrenamiento de la red neuronal se contará con cuatro coordenadas de imagen, dos por cada una de las cámaras, la salida estará formada ya sea por las coordenadas espaciales del efector final, o por los ángulos de las articulaciones. Utilizando este método se puede identificar cualquier punto en coordenadas espaciales del espacio de trabajo, con tal

que la red neuronal sea entrenada con un número considerable de puntos y con un margen de error "suficientemente" pequeño.

Visión planar

En este método de visión de máquina se utiliza una cámara, la cual proporciona dos coordenadas de imagen, con este método se puede identificar un punto en un plano del espacio de trabajo. Entonces para el entrenamiento de la red neuronal se tendrá como entradas dos coordenadas de imagen y como salidas se tendrá, ya sea, las coordenadas espaciales de un punto en un plano del espacio de trabajo o los ángulos de articulación que definan un plano en el espacio de trabajo.

Preprocesamiento

El objetivo de control visual del robot nos marca la necesidad del tratamiento de imágenes, para este caso la cámara que se está utilizando es una cámara CCD en blanco y negro, el espacio de trabajo no tendrá una iluminación especial, además si se considera que la toma de imágenes se hará a una frecuencia máxima de 30 (Hz) por lo tanto es indispensable aplicar un preprocesamiento a la imagen captada. El preprocesamiento aplicado a la imagen será de forma simple, la aplicación de un umbral a la imagen tomada y de esta forma se tiene una imagen binaria libre de fallas en la imagen original tomada. Esto se realizó utilizando Labview, Matlab e Imaq Vision el programa (VI) forma parte de programa general realizado en Labview[1]. Este procedimiento es suficiente para la aplicación en el presente trabajo, debido a que se trata de una operación de visión de bajo nivel.

Cinemática directa del robot

Para cumplir los objetivos del presente trabajo, se fijarán el cuarto y quinto grado de libertad, de esta forma se define la cinemática en relación a la figura 4.2.

De la Fig. 4.2 a es el ángulo que forma un vector que une el origen de la articulación 2 con el efector final, tomando como referencia el eje de la articulación 2, B es al ángluo formado por el mismo vector tomando como referencia el eje horizontal x. En forma directa, geométricamente de la Fig. 4.2 se pueden obtener las ecuaciones que definen la posición del efector final en coordenadas espaciales en función de los ángulos de articulación.

$$Z = l_1 + l_2 \sin(\theta_2) + l_3 \sin(\theta_3) \tag{4.1}$$

$$X = [l_2 \cos(\theta_2) + l_3 \cos(\theta_3)] \cos(\theta_1) \tag{4.2}$$

$$Y = [l_2 \cos(\theta_2) + l_3 \cos(\theta_3)] \sin(\theta_1) \tag{4.3}$$

[1] El autor ofrece una copia electrónica bajo petición expresa del lector

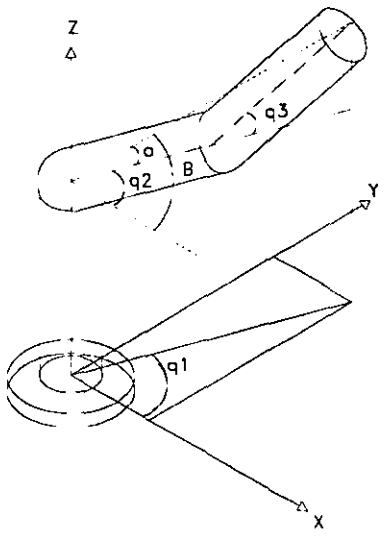

Figura 4.2: Esquema de robot utilizado

Donde, $q1 = \theta_1$, $q2 = \theta_2$, $q3 = \theta_3$, son los ángulos de articulación y $l1$, $l2$, $l3$ son las dimensiones de los elementos del robot.

Cinemática inversa del robot

Observando la figura anterior, se tiene:

$$\theta_1 = \tan^{-1}\left(\frac{Y}{X}\right) \tag{4.4}$$

Aplicando el Teorema de Pitágoras

$$r^2 = X^2 + Y^2$$

Considerando ahora únicamente los elementos 2 y 3, que están situados en un plano, utilizando la ley de los cosenos y definiendo $Pz = Z - l_1$ se tiene:

$$r^2 + Pz^2 = l_2^2 + l_3^3 + 2l_2l_3\cos(\theta_3)$$

$$\cos(\theta_3) = \left(\frac{X^2 + Y^2 + Pz^2 - l_2^2 - l_3^2}{2l_2l_3}\right) \tag{4.5}$$

Utilizando identidades trigonométricas, se puede obtener:

$$\sin(\theta_3) = \pm\sqrt{1 - \cos^2(\theta_3)} \tag{4.6}$$

Con las ecuaciones(4.5), en (4.6), se puede encontrar el valor de θ_3,con la función tangente inversa en lugar de la función coseno inverso, debido a que la última permite recuperar la configuración codo arriba y codo abajo utilizando el signo adecuado:

$$\theta_3 = \tan^{-1}\left(\frac{\pm\sqrt{1-\cos^2(\theta_3)}}{\cos(\theta_3)}\right) \tag{4.7}$$

donde es claro que $\cos(\theta_3)$ (4.5) queda en función de las dimensiones de los elementos del robot y las coordenadas espaciales, así como $Pz = Z - l_1$. Las dos posibles soluciones corresponden; la positiva con la configuración codo abajo y la negativa con la configuración codo arriba.De la Fig. (4.2) se puede calcular B, como:

$$B = \tan^{-1}\left(\frac{Pz}{r}\right) = \tan^{-1}\left(\frac{Pz}{\pm\sqrt{X^2+Y^2}}\right) \tag{4.8}$$

Aplicando nuevamente el Teorema de Pitágoras, se puede calcular a:

$$a = \tan^{-1}\left(\frac{l_3\sin(\theta_3)}{l_2 + l_3\cos(\theta_3)}\right) \tag{4.9}$$

De la Fig. (4.2), también se observa que:

$$\theta_2 = B - a$$

sustituyendo (4.9) y (4.8) en la ecuación anterior se tiene:

$$\theta_2 = \tan^{-1}\left(\frac{Pz}{\pm\sqrt{X^2+Y^2}}\right) - \tan^{-1}\left(\frac{l_3\sin(\theta_3)}{l_2 + l_3\cos(\theta_3)}\right) \tag{4.10}$$

Nuevamente para θ_2 se observan 2 posibles soluciones, que corresponden también (signo positivo configuración codo abajo, signo negativo configuración codo arriba).Las ecuaciones (4.4), (4.10) y (4.7), permiten la solución al problema de cinemática inversa para el robot de 3 grados de libertad, con articulaciones de revolución.

4.3 El modelo de la cámara

El proceso de transformación de perspectiva (transformación de imagen) proyecta puntos tridimensionales sobre un plano. Las transformaciones de perspectiva juegan un papel clave en el procesamiento de imagen, ya que nos suministran una aproximación a la manera en que una imagen se forma al observar un mundo tridimensional [7].

Enseguida se presenta en forma resumida el proceso de determinación del modelo de la cámara.

Los cuatro pasos para la transformación de coordenadas espaciales a coordenadas de la cámara.

La Fig. 4.3 ilustra la geometría básica del modelo de la cámara [15] donde (x_w, y_w, z_w) son las coordenadas de un punto de un objeto en el sistema de coordenadas espaciales absolutas (coordenadas de mundo), (x, y, z) son las coordenadas en 3D del punto del objeto en el sistema de coordenadas de la cámara, el cual es centrado en O (el centro óptico), con el eje z igual que el eje óptico. (X, Y) es el sistema de coordenadas de imagen, centrado en O_i (intersección del eje óptico z y el plano de imagen) y paralelo a los ejes x e y (del sistema de coordenadas de la cámara),. f (longitud focal efectiva) es la distancia entre el plano de imagen y el centro óptico. (X_u, Y_u) son las coordenadas de imagen del punto (x, y, z) sin distorsión (el lente se considera perfecto). (X_d, Y_d) son las coordenadas distorsionadas de imagen del punto (x, y, z). (X_f, Y_f) son las coordenadas de la foto tomada por la computadora (dadas en pixels).

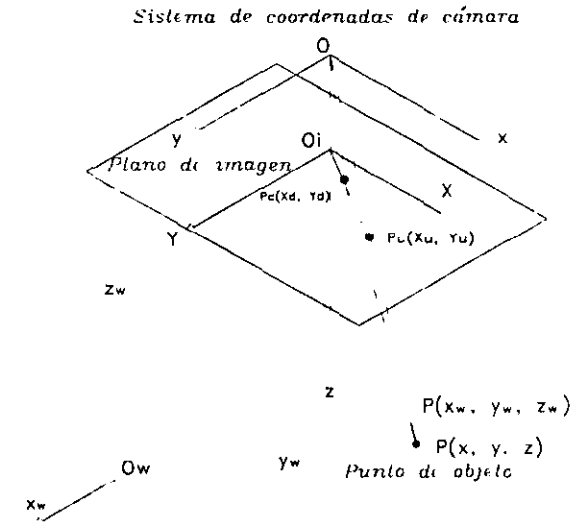

Figura 4.3: Transformación de coordenadas

Paso 1: Transformación de cuerpo rígido del sistema de coordenadas de mundo del objeto (x_w, y_w, z_w) al sistema de coordenadas de 3D de la cámara.(x, y, z)

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + T \tag{4.11}$$

donde R es una matriz de 3 × 3 puede medirse o calibrarse [15]

$$R = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} \tag{4.12}$$

y T es el vector de traslación puede medirse o calibrarse [15]

$$T = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \tag{4.13}$$

Paso 2: Transformación de las coordenadas de la cámara (x, y, z) a las coordenadas de imagen ideales (sin distorsión) (X_u, Y_u) utilizando proyección de perspectiva con geometría de la cámara

$$X_u = f\frac{x}{z} \tag{4.14}$$

$$Y_u = f\frac{y}{z} \tag{4.15}$$

donde f longitud focal efectiva párametro a calibrarse.

Paso 3: Distorsión radial del lente

$$X_d + D_x = X_u \tag{4.16}$$

$$Y_d + D_x = Y_u \tag{4.17}$$

donde (X_d, Y_d) son las coordenadas reales (con distorsión) del plano de imagen y k_i son los coeficientes de distorsión a calibrar

$$r = \sqrt{X_d^2 + X_d^2}$$

$$D_x = X_d(k_1 r^2 + k_2 r^4 + ...)$$

$$D_y = Y_d(k_1 r^2 + k_2 r^4 + ...)$$

Paso 4: Transformación de coordenadas de imagen real (X_d, Y_d) a las coordenadas de imagen en la computadora (X_f, Y_f)

$$X_f = s_x d_x^{'-1} X_d + C_x \tag{4.18}$$

$$Y_f = d_y^{-1} Y_d + C_y \tag{4.19}$$

donde s_x es el factor de incertidumbre de escala a ser calibrado y

(X_f, Y_f) números de renglón y columna de los pixels de imagen en la foto de la computadora,

$(C_x,\ C_y)$ números de renglón y columna del centro de la foto de la imagen,

$$d_x' = d_x \frac{N_{cx}}{N_{fx}} \tag{4.20}$$

d_x distancia de centro a centro entre elementos sensores en la dirección X (línea de exploración),
d_y distancia de centro a centro entre elementos sensores en la dirección Y (línea de exploración),
N_{cx} número de elementos sensores en la dirección X,
N_{fx} número de pixels en una línea, como es muestreada por la computadora.

Combinando los últimos tres pasos, se puede relacionar las coordenadas de imagen (en la computadora) $(X_f,\ Y_f)$ con las coordenadas de 3D del punto del objeto en el sistema de coordenadas de la cámara de la siguiente forma:

$$s_x^{-1} d_x' X + s_x^{-1} d_x' X k_1 r^2 = f\frac{x}{z} \tag{4.21}$$

$$d_y' Y + d_y Y k_1 r^2 = f\frac{y}{z} \tag{4.22}$$

donde:

$$r = \sqrt{(s_x^{-1} d_x' X)^2 + (d_y Y)^2}$$

Sustituyendo (4.11) en (4.21) y (4.22) se tiene

$$s_x^{-1} d_x' X + s_x^{-1} d_x' X k_1 r^2 = f\frac{r_1 x_w + r_2 y_w + r_3 z_w + T_x}{r_7 x_w + r_8 y_w + r_9 z_w + T_z} \tag{4.23}$$

$$d_y' Y + d_y Y k_1 r^2 = f\frac{r_4 x_w + r_5 y_w + r_6 z_w + T_y}{r_7 x_w + r_8 y_w + r_9 z_w + T_z} \tag{4.24}$$

Se puede catalogar los parámetros, en intrínsecos y extrínsecos:

Parámetros Extrínsecos: Estos son los elementos de la matriz de rotación R (en función de los ángulos de Euler,3) y los elementos (3) del vector de traslación T.

Parámetros intrínsecos: f longitud focal efectiva, k_1 coeficiente de distorsión del lente, s_x factor de escala para x debido a la exploración de la cámara, $(C_x,\ C_y)$ coordenadas del origen de imagen de la computadora.

Como se puede observar, el número de incógnitas, que corresponden con el número de parámetros a encontrar son 11. El procedimiento de calibración de la cámara consiste en

determinar de forma sistemática estos parámetros. Tsai [15] propone un algoritmo de calibración de la cámara de forma no-coplanar, utilizando 11 puntos en coordenadas de imagen, de los cuales se conocen sus coordenadas espaciales.

Después de realizar el proceso de calibración de la cámara, será posible entonces, a partir del conocimiento de las coordenadas de imagen, determinar a que punto del espacio (en coordenadas de mundo) cooresponden éstas. Es necesario mencionar que si la posición de la cámara cambia será necesario determinar de nueva cuenta los parámetros extrínsecos, los parámetros intrínsecos sólo será necesario determinarlos una vez para una cámara.

En el presente trabajo se utilizarán las redes neuronales para determinar las coordenadas de un punto en un plano en coordenadas espaciales, por lo que la derivación del modelo de la cámara y el método de calibración de la cámara, se presenta como un método alternativo para calcular las mismas coordenadas.

Capítulo 5

Retroalimentación visual por medio de una NN

En el capítulo anterior se presentó la teoría de operación de la cámara como sensor visual del espacio de trabajo, así como también la derivación de un modelo de la cámara que nos permite determinar las coordenadas espaciales de un punto del objeto de trabajo, lo cual representa el método clásico para sistemas de visión artificial, en el presente capítulo se propone la utilización de las redes neuroales (NN) como método de aproximación no lineal para el mapeo de coordenadas de imagen a coordenadas espaciales.

5.1 Diseño de la red neuronal

Los siguientes comentarios son en relación a las capacidades de aproximación de las redes neuronales. Considere los mapeos en $C(R^n, R)$ y $C(R^n, R^m)$. Por el Teorema de Stone-Weierstrass [30], se sabe que cualquier función de esta clase puede ser aproximada hasta cualquier grado de exactitud por un polinomio[1]

$$f(x_1, x_2, \ldots x_n) \approx \alpha_0 + \Sigma_i \alpha_i x_i + \Sigma_i \Sigma_j \alpha_{ij} x_i x_j + \ldots \tag{5.1}$$

donde α_1, α_i, $\alpha_{ij} \in R$. Similarmente, si $\phi_i(x_1, x_2, \ldots x_n)$, $(i = 1, 2, \ldots)$ son elementos de un conjunto ortonormal completo, f puede expresarse como

$$f(x_1, x_2, \ldots x_n) = \Sigma_{i=1}^{N} c_i \phi_i(x_1, x_2, \ldots x_n) \tag{5.2}$$

La primera cuestión es saber si una red neuronal multicapa (3 capas) puede usarse o no para aproximar una función $f \in C(R^n, R)$ para cualquier grado de exactitud deseado.

Se ha demostrado [31], usando el Teorema de Stone-Weierstrass, que cualquier función continua $f \in C(R^n, R)$ sobre un conjunto compacto de R^n.se aproximar por una red de dos capas con un número de nodos arbitrariamente grande en la capa de oculta.

[1] Debe tomarse en cuenta que el grado de exactitud depende del grado del polinomio

Como los polinomios (5.1) y expansiones ortonormales (5.2) pueden también aproximar funciones en $C(R^n, R)$ para cualquier grado de exactitud, las ventajas de las redes neuronales son menos obvias y se debe justificar la utilización de estas, sobre la base de consideraciones prácticas. En particular, las siguientes cuestiones se deben tomar en cuenta cuando se utilicen las redes neuronales como método de aproximación de funciones no lineales

- ¿Son las redes neuronales representaciones más parsimoniosas de clases especiales de funciones continuas, en que éstas necesitan sólo algunos parámetros?. Si esto es así, ¿Cuales son las características de tales funciones?
- Dado un mapeo no lineal f el cual tiene que ser aproximado, ¿Que dicta el número de capas y el número de nodos en cada capa de la red?.

De acuerdo con lo anterior, en el diseño de las redes neuronales no existe una metodología precisa que permita determinar la estructura de la red neuronal, así como el número de neuronas y el número de capas que deben formar la red neuronal, se debe hacer uso por tanto del conocimiento que se tenga del sistema o proceso bajo estudio y del manejo mismo de las redes neuronales. En el presente trabajo se utilizará una red neuronal del tipo estática con prealimentación y totalmente conectada, esto considerando que la aplicación que se hará en este trabajo es del tipo estático (mapeo de coordenadas), el número de capas de la red neuronal se tomará de los resultados obtenidos en trabajos anteriores [16], en donde se ha encontrado que dada una función no lineal continua existe una red neuronal con tres capas, capa de entrada, capa de neuronas ocultas y capa de salida, que es capaz de aproximar tal función en un conjunto compacto [16] y con un error de aproximación arbitrariamente pequeño.

Por tanto para el presente trabajo se propone utilizar una red neuronal estática, con prealimentación, totalmente conectada y de tres capas, tal como se muestra en la Fig. 5.1.

5.2 Entrenamiento de la red neuronal

El método de entrenamiento de la red neuronal utilizado en el presente trabajo es el método de retropropagación (BP), ésto debido a que es el método más eficiente computacionalmente, para el cálculo de los pesos de la red neuronal.

5.2.1 Algoritmo rápido de retropropagación de error

Este método de entrenamiento de redes neuronales fue propuesto por Nicolaos B. Karayiannis [2], este método propone un criterio generalizado para el entrenamiento de redes neuronales prealimentadas.

Dependiendo de la estrategia de optimización utilizada, este método lleva a una variedad de algoritmos de rápida retropropagación, tanto para redes de una capa como para redes neuronales multicapa. El algoritmo más simple obtenido de este criterio generalizado es

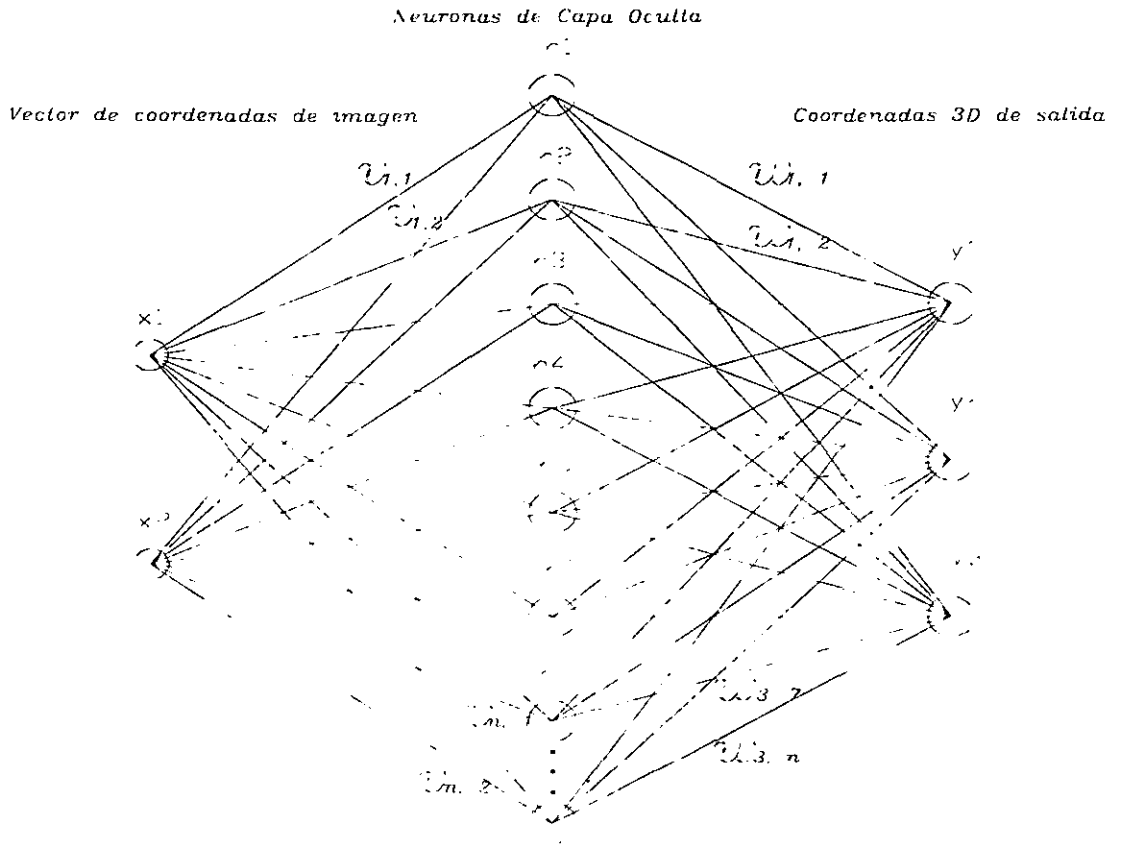

Figura 5.1: Red Neuronal Propuesta

el algoritmo de regla delta, propuesto para el entrenamiento de redes neuronales de una capa. La aplicación de una estrategia de optimización similar a redes neuronales multicapa en conjunto con el criterio de generalización propuesto, proporciona el algoritmo rápido de retropropagación.

Durante los primeros pasos de la revitalizada investigación en redes neuronales, los algoritmos de entrenamiento fueron satisfactorios. Como la investigación en redes neuronales se mueve del estado del arte hacia aplicaciones en tiempo real, el tiempo de entrenamiento asociado y los requerimientos de cálculo, se hacen un factor incrementalmente importante, afectando la comparación de redes neuronales con otras alternativas competitivas. De esta forma la existencia de algoritmos rápidos y eficientes de entrenamiento es crucial para este campo de investigación en el futuro. Un algoritmo de entrenamiento es juzgado sobre la base de ciertos requerimientos contrapuestos, tales como simplicidad, flexibilidad y eficiencia. La simplicidad de un algoritmo de entrenamiento relaciona el esfuerzo requerido para la

reproducción e implementación por un programador promedio y no por un programador experimentado. La flexibilidad de un algoritmo de entrenamiento se refiere al esfuerzo analítico y de cálculo requerido para la aplicación del algoritmo al entrenamiento de redes neuronales en varias configuraciones.

La eficiencia del algoritmo de aprendizaje es evaluada con respecto a los requerimientos de cálculo y tiempo para el entrenamiento de una red neuronal así como también el desempeño de la red resultante.

La Regla Delta, introducida por Widrow et.al. [4] hace algunas décadas fue la base para el desarrollo de algoritmos de aprendizaje para el entrenamiento de redes neuronales de una capa. Además, los algoritmos de aprendizaje basados en la Regla Delta son caracterizados por su lenta convergencia y en algunas situaciones, pueden caer en un mínimo local.

El algoritmo de entrenamiento más popular para el entrenamiento de redes neuronales es el algoritmo de retropropagación de error, originalmente inventado por Werbos [5] y popularizado por Rumelhart. El impacto de este algoritmo sobre la investigación de redes neuronales ha sido indudablemente enorme. Por otro lado es ampliamente conocido que este algoritmo sufre de una convergencia muy lenta.

En el algoritmo de retropropagación presentado, la velocidad de aprendizaje es adaptada para reducir el valor de la dirección del gradiente en una forma cercana a la óptima. El algoritmo presentado forma parte de la familia de Algoritmos de Aprendizaje Eficiente para Redes Neuronales (siglas en inglés "ELEANE"), familia en la cual existen algoritmos para redes neuronales multicapa que combinan las propiedades de segundo orden de los algoritmos de redes neuronales de una capa y la simplicidad del gradiente descendiente [2].

Las redes neuronales con prealimentación han sido convencionalmente entrenadas minimizando un criterio de error cuadrático entre la salida esperada y los valores estimados proporcionados por la red. El algoritmo presentado para el entrenamiento de redes neuronales multicapa realiza el entrenamiento de forma rápida enfocándose sobre el uso de un criterio alternativo después de los ciclos de adaptación iniciales.

El presente algoritmo es derivado en el Apéndice A, minimizando la función objetivo $G_k(\lambda)$ definida por:

$$G_k(\lambda) = \lambda \Sigma_{i=0}^{n_0} \phi_2(y_{i,k} - \hat{y}_{i,k}) + (1-\lambda)\Sigma_{i=1}^{n_0} \phi_1(y_{i,k} - \hat{y}_{i,k}) \tag{5.3}$$

para $k = 1, 2, ..., m$, donde m es el número de pares entrada-salida de la red, $\phi_2(x) = \frac{1}{2}x^2$ y $\phi_1(x) = (\frac{1}{\beta}) \ln[\cosh \beta x]$[2].[2] La ecuación de actualización para los pesos sinápticos $w_{p,k}$;

$$w_{p,k} = w_{p,k-1} + \alpha \varepsilon_{p,k}^0(\lambda) - h_k \tag{5.4}$$

[2]Esta función satisface el criterio de error establecido, β es una constante que define el comportamiento de la función ϕ_1.

Para este caso red 2 capas implica ϕ_1, ϕ_2

Si la salida de la red es analógica

$$\epsilon^0_{p,k}(\lambda) = \lambda(y_{p,k} - \hat{y}_{p,k}) + (1-\lambda)\tanh[\beta(y_{p,k} - \hat{y}_{p,k})] \tag{5.5}$$

Por otro lado si la salida de la red consiste de elementos binarios.

$$\epsilon^0_{p,k}(\lambda) = (1 - \hat{y}^2_{p,k})(y_{p,k} - \lambda\hat{y}_{p,k}) \tag{5.6}$$

También se muestra en el apéndice A que los pesos sinápticos $v_{p,q}$ se pueden actualizar por medio de la ecuación:

$$v_{p,k} = v_{p,k-1} + \alpha\epsilon^h_{p,k}(\lambda)x_k \tag{5.7}$$

donde:

$$\epsilon^h_{p,k}(\lambda) = (1 - \hat{h}^2_{p,k})\Sigma_{i=1}^{n0}\epsilon^0_{i,k}(\lambda)w_{i,p} \tag{5.8}$$

El error de salida $\epsilon^0_{i,k}(\lambda)$ está dado por (5.5) si la salida de la red es analógica y (5.6) si la salida de la red está formada por elementos binarios. El algoritmo rápido de retropropagación, se resume en el diagrama de flujo, mostrado más adelante.

Debido a su simplicidad, este algoritmo proporciona una base ideal para investigación del papel de λ durante el entrenamiento de la red.

Considere aquí, que la salida de la red es analógica. Durante los ciclos iniciales de adaptación, $\lambda = 1$. En este caso el papel del primer término de (5.5) es dominante. Como $e_{p,k} = y_{p,k} - \hat{y}_{p,k}$ disminuye durante el entrenamiento, λ también disminuye[3] y el papel del segundo término de (5.5) se hace dominante. En este caso el comportamiento de la no-linealidad $\tanh(\beta e_{p,k})$ depende de ambos β y $e_{p,k} = y_{p,k} - \hat{y}_{p,k}$. Si $\beta e_{p,k}$ es suficientemente pequeña $\tanh(\beta e_{p,k}) \approx \beta e_{p,k}$ y por tanto $\epsilon^0_{i,k}(\lambda) \approx [\lambda + (1-\lambda)\beta]e_{p,k}$.

En el caso partícular donde $\beta = 1$, $\epsilon^0_{p,k} = e_{p,k} = y_{p,k} - \hat{y}_{p,k}$. Si $\beta e_{p,k}$ es suficientemente grande $\tanh(\beta e_{p,k})$ se acerca a +1 ó -1, dependiendo del signo de $e_{p,k} = y_{p,k} - \hat{y}_{p,k}$.

La discusión anterior indica que la convergencia del algoritmo resultante puede ser monitoreada por el valor de β, el cual determina el comportamiento de la no-linealidad $\tanh(\beta e_{p,k})$. Si $\beta \approx 1$, $\tanh(\beta e_{p,k})$ se hace una función lineal y la convergencia del algoritmo resultante se espera sea similar con el algoritmo de retropropagación de error. Si el valor de β es considerablemente más grande que la unidad, la convergencia del algoritmo se espera que mejore. Cuando β se aproxima a infinito $\tanh(\beta(y_{p,k} - \hat{y}_{p,k}))$ se aproxima asíntoticamente a la función $sgn(y_{p,k} - \hat{y}_{p,k})$. En este caso, la convergencia del algoritmo no se garantiza..

La investigación del papel de λ es más informativa en el caso donde la salida de la red es binaria. Cuando $\lambda = 1$, (5.6) sería igual a $\epsilon^0_{p,k}(1) = (1 - \hat{y}^2_{p,k})(y_{p,k} - \hat{y}_{p,k})$.

Claramente si $\lambda = 1$, este algoritmo coincide con el algoritmo de retropropagación de error. En este caso. la adaptación de los pesos sinápticos $w_{p,q}$ es determinada por el error $e_{p,k} = y_{p,k} - \hat{y}_{p,k}$.

[3] $\lambda = \exp(\frac{\mu}{E^2})$. E se actualiza en el algoritmo de entrenamiento, μ es una constante seleccionada heurísticamente

Se puede ver fácilmente que:

$$\epsilon^0_{p,k}(1) = \left\{ \begin{array}{c} (1+\hat{y}_{p,k})(y_{p,k}-\hat{y}_{p,k})^2, \text{ si } y_{p,k}=1 \\ -(1-\hat{y}_{p,k})(y_{p,k}-\hat{y}_{p,k})^2, si\ y_{p,k}=-1 \end{array} \right\} \tag{5.9}$$

Cuando el estimado $\hat{y}_{p,k}$, está cercano a su objetivo $y_{p,k}$, la adaptación de los pesos sinápticos es dominada por el término $e^2_{p,k} = (y_{p,k} - \hat{y}_{p,k})^2$. Este término hace muy lenta la convergencia del algoritmo después de los ciclos iniciales de adaptación. Se puede verificar fácilmente que si $\lambda = 0$:

$$\epsilon^{\upsilon}_{p,k}(1) = \left\{ \begin{array}{c} y_{p,k}(1+\hat{y}_{p,k})(y_{p,k}-\hat{y}_{p,k}), \text{ si } y_{p,k}=1 \\ -y_{p,k}(1-\hat{y}_{p,k})(y_{p,k}-\hat{y}_{p,k})^2, si\ y_{p,k}=-1 \end{array} \right\} \tag{5.10}$$

La ecuación anterior indica que, sin importar el valor de λ, la adaptación de los pesos sinápticos es determinada por el error $e_{p,k} = y_{p,k} - \hat{y}_{p,k}$. Por otro lado, el efecto del error $e_{p,k} = y_{p,k} - \hat{y}_{p,k}$ sobre la adaptación de los pesos sinápticos es afectada por el término de offset, el cual cambia durante el entrenamiento de la red.

El análisis anterior proporciona la base para la investigación de la relación entre el criterio convencional de error cuadrático, el criterio de entropia relativa y el criterio generalizado utilizado en este trabajo.

5.2.2 Diagrama de flujo detallado del algoritmo rápido de retropropagación para redes neuronales multicapa, con una capa de unidades ocultas.

Inicio

Inicialice aleatoriamente las matrices de pesos sinápticos W y V.
Seleccione μ
$\lambda = 1$.

1. $k = 0$ *(número de asociación)*
   $E = 0$

2. $k = k + 1$

   $x = x_k$

   $y = y_k$

   $\hat{h}_j = \rho(\Sigma_{l=1}^{ni} v_{jl} x_l)$

   $\hat{y}_i = \sigma(\Sigma_{j=1}^{nh} w_{ij} \hat{h}_j)$

   $\epsilon^0_i(\lambda) = \lambda(y_i - \hat{y}_i) + (1-\lambda)\tanh[\beta(y_i - \hat{y}_i)]$, *Si la salida de la red es analógica.*

$\epsilon_i^0(\lambda) = (1 - \hat{y}_i^2)(y_i - \lambda\hat{y}_i)$, *Si la salida de la red es binaria.*

$W_i \leftarrow W_i + \alpha\epsilon_i^0(\lambda)\hat{\mathbf{h}}$

$\epsilon_j^n = (1 - \hat{h}_j^2)\Sigma_{i=1}^{n0}\epsilon_i^0(\lambda)w_{ij}$

$V_j \leftarrow V_j + \alpha\epsilon_i^h(\lambda)x$

$\hat{h}_j = \rho(\Sigma_{l=1}^{ni} v_{jl}x_l)$

$\hat{y}_i = \sigma(\Sigma_{j=1}^{nh} w_{ij}\hat{h}_j)$

$E \leftarrow E + \frac{1}{2}\Sigma_{i=0}^{n0}(y_i - \hat{y}_i)^2$

Si k<m; entonces regresa al punto 2.

$\lambda = \exp(-\frac{\mu}{E^2})$

Si $E > E_0$, *entonces regresa al punto 1, de otra forma termina la ejecución del algoritmo.*

5.2.3 Determinación del conjunto de entrenamiento.

Antes de la sesión de entrenamiento debe determinarse el conjunto de casos de entrenamiento S_{tr}. La selección de los casos de entrenamiento es importante para la inferencia inductiva. Los requerimientos generales para los casos de entrenamiento, del conjunto de entrenamiento son:

1) Consistentes.
2) Presentables.
3) Informativos.

La consistencia significa que los casos de entrenamiento deben ser consistentes con el comportamiento del proceso bajo estudio, esto es, $S_{tr} \triangleq (x^p, z^p) \in P$, donde P es el conjunto de pares de entrenamiento con alta probabilidad de asociación; la representabilidad significa que estos casos de entrenamiento deben satisfacer las restricciones sobre x tal que se puede obtener un mínimo global de la función de evaluación con respecto a los parámetros, esto es, $x^p \in G$ para $p = 1, 2, ..., P$, donde G formado por un conjunto acotado por alguna restricción; la informatividad implica que los componentes de entrada de los casos de entrenamiento deben distribuirse uniformemente o tan uniformemente como sea posible en el dominio de interés y los componentes de salida deben cubrir completamente o tan completamente como sea posible sobre el espectro del valor de salida z y la informatividad también implica no redundancia, esto es, sin pérdida de información, el conjunto de entrenamiento debe ser mínimo.

5.2.4 Obtención del conjunto de datos de entrenamiento

Definición del espacio de trabajo

Se creó un ambiente de alto contraste, que comprende el robot y la mesa de trabajo, se cubrieron de color negro las articulaciones del robot, la mesa y el fondo del espacio de trabajo se cubrieron de color blanco, en el efector final se colocó una marca circular de 2.54 (cm) de diámetro y de color blanco sobre un fondo negro, esto con el fin de fácil identificación dentro de la imagen tomada por la cámara.

Procedimiento de toma de fotografías

Para formar el conjunto de pares de entrenamiento se utilizó un programa en Labview[4] junto con un proceso interactivo, se tomaron 400 fotos (tomas con la cámara), las cuales se tomaron en posiciones diferentes del efector final, en dicho efector final se puso una marca circular (2.54 cm), de tal forma que se pudieran obtener las coordenadas del centroide de la marca y relacionarlas con las coordenadas de imagen. Las 400 posiciones del efector final definen un plano en las coordenadas X vs Z.

5.3 Preprocesamiento

El objetivo de control visual del robot nos marca la necesidad del tratamiento de imágenes, para este caso la cámara que estamos utilizando es una cámara CCD en blanco y negro, el espacio de trabajo no tendrá una iluminación especial, además si se considera que la toma de imágenes se hará a una frecuencia máxima de 30 (Hz) por lo tanto es indispensable aplicar un preprocesamiento a la imagen captada. El preprocesamiento aplicado a la imagen será de forma simple la aplicación de un umbral a la imagen tomada y de esta forma se obtiene una imagen binaria, libre de fallas de la imagen original, esto se realizó utilizando Labview Matlab e Imaq vision el programa (VI) se presenta en el Apéndice B.

5.3.1 Obtención de las coordenadas del objeto de trabajo

En el presente trabajo para facilitar la identificación y la operación de retroalimentación visual se colocará en el efector final una círculo blanco en un fondo negro, la operación a realizar por el robot será el seguimiento de una trayectoria definida en el espacio de trabajo en coordenadas espaciales.

5.4 Entrenamiento de la red neuronal

El entrenamiento de la red neuronal propuesta se realizó utilizando el algoritmo rápido de retropropagación [2], tanto para simulación como para las imágenes reales.

[4]Se ofrece una copia electrónica del programa a petición expresa del lector.

El entrenamiento se realizó fuera de línea con 400 pares de entrada-salida, los parámetros utilizados para el entrenamiento de la red neuronal propuesta fueron los siguientes:

$\alpha = 0.06,\ \beta = 2.5,\ \mu = 0.7, N_i = 2,\ N_h = 15,\ N_s = 3\ y\ m = 400$, donde:

α es la velocidad de entrenamiento.

β es una constante de convergencia para la función de transferencia de salida de la red neuronal.

μ es una constante de la función de error.

N_i es en número de entradas a la red neuronal.

N_h es el número de neuronas en la capa oculta de la red neuronal.

N_s es el número de salidas de la red neuronal.

m es el número de datos entrada-salida proporcionados para el entrenamiento de la red neuronal.

Los parámetros mencionados anteriormente se determinaron de forma heurística[5] y utilizando los resultados obtenidos en [2].

[5]Se realizaron pruebas con el algoritmo, para entrenar la red y que realizara generalización de funciones no lineales conocidas.

5.4.1 Resultados de entrenamiento de la red neuronal con datos de simulación.

Generación de datos entrada salida para simulación.

Utlizando el modelo de la cámara [7] se implementó un programa en Matlab que como entrada tiene las coordenadas de las tres caras de un cubo y como salida entrega las coordenadas de imagen correspondientes, este programa se presenta en el Apéndice B.(Corimagx2.m)

En la Fig. 5.2 se presentan las coordenadas espaciales del objeto, así como también las coordenadas de imagen obtenidas utilizando el modelo de la cámara. En conjunto estas gráficas representan los datos de entrenamiento de la red neuronal propuesta.

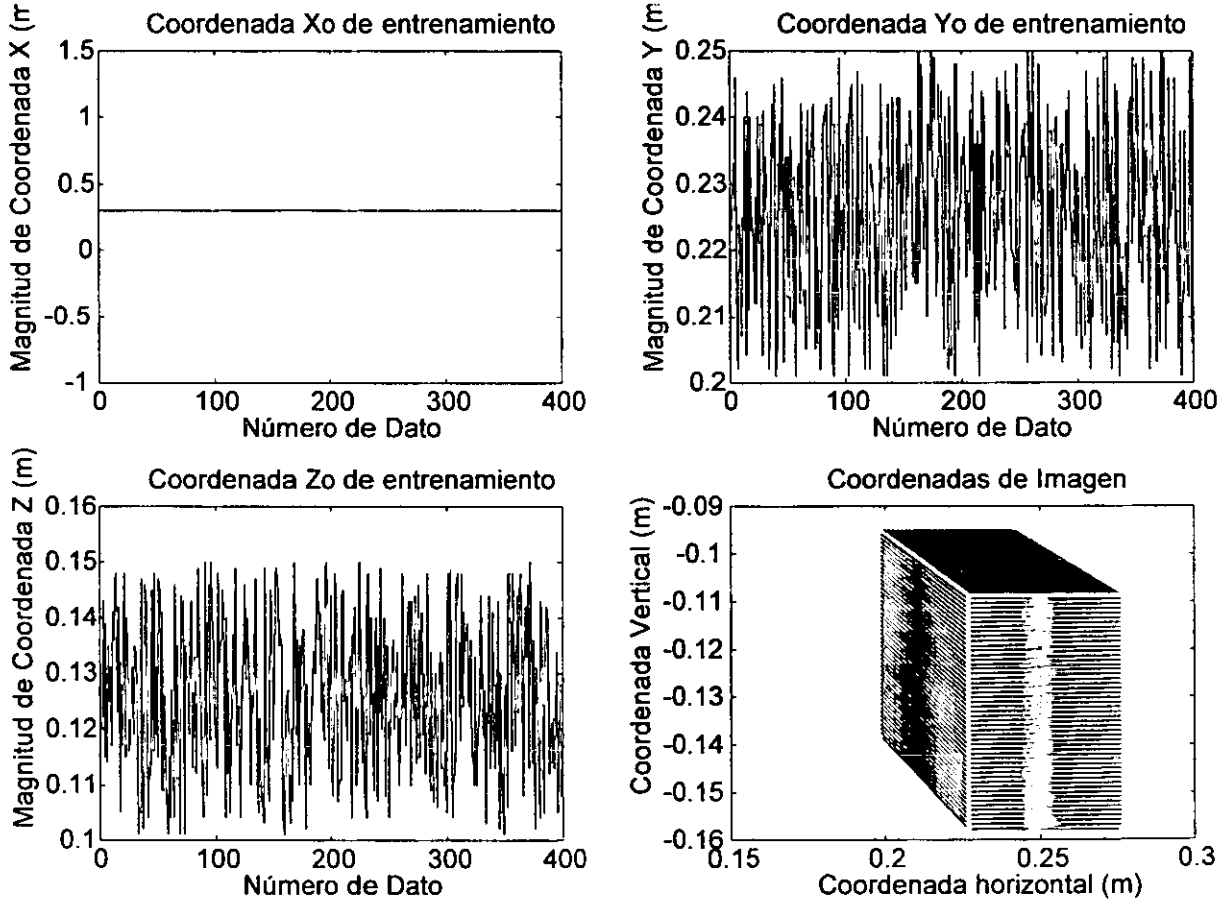

Figura 5.2: Coordenadas de entrada para entrenamiento

Para esta prueba de simulación se utilizó el plano de coordenadas de imagen que es perpendicular a la hoja del papel, que corresponde con el plano Y vs Z, con X constante.

En la Fig. 5.3 se presenta el error resultante del proceso de generalización de la red neuronal propuesta, para los datos de simulación del entrenamiento, en coordenadas espaciales del objeto y las coordenadas calculadas como salidas de la red neuronal para los datos de entrada de entrenamiento, así como también la gráfica del desempeño del algoritmo de entrenamiento, donde se presenta la relación del error contra el número de epocas[6] del proceso de entrenamiento.

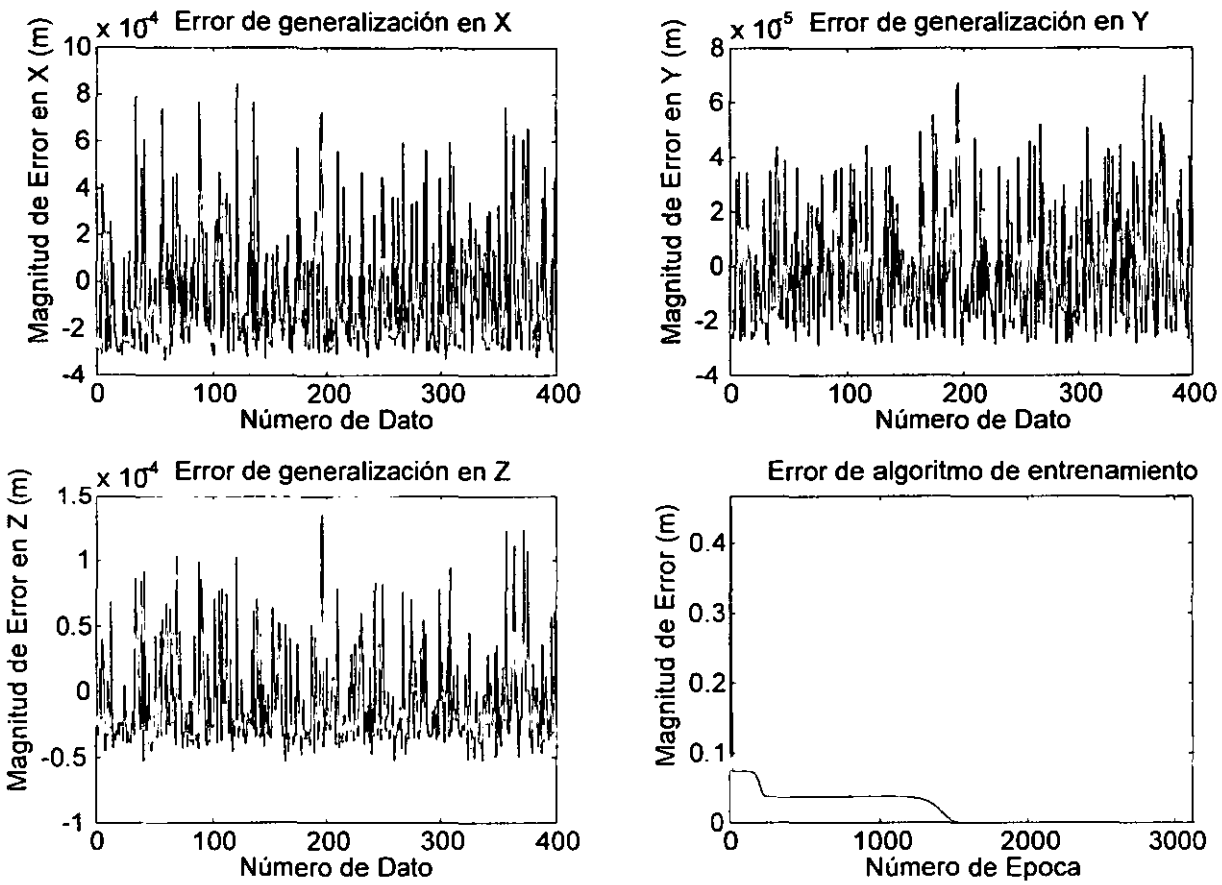

Figura 5.3: Errores de generalización de la red

Observación 5.1 *La generalización se refiere al proceso de cálculo de la salida con las matrices de pesos (resultado del entrenamiento) y con las entradas, que para este caso pertenecen al conjunto de entrenamiento*

[6]Una época se puede entender como un ciclo de entrenamiento para el lote completo de datos.

En la Fig. 5.4 se presentan los resultados de comparación de las coordenadas proporcionadas como salidas para el algoritmo de entrenamiento de la red neuronal propuesta, esto es, Xo, Yo y Zo, y las coordenadas calculadas como salidas de la red neuronal propuesta.Xoe, Yoe y Zoe, se debe aclarar que no hay diferencia apreciable entre ambas, sobre todo para las coordenadas Y y Z[7].

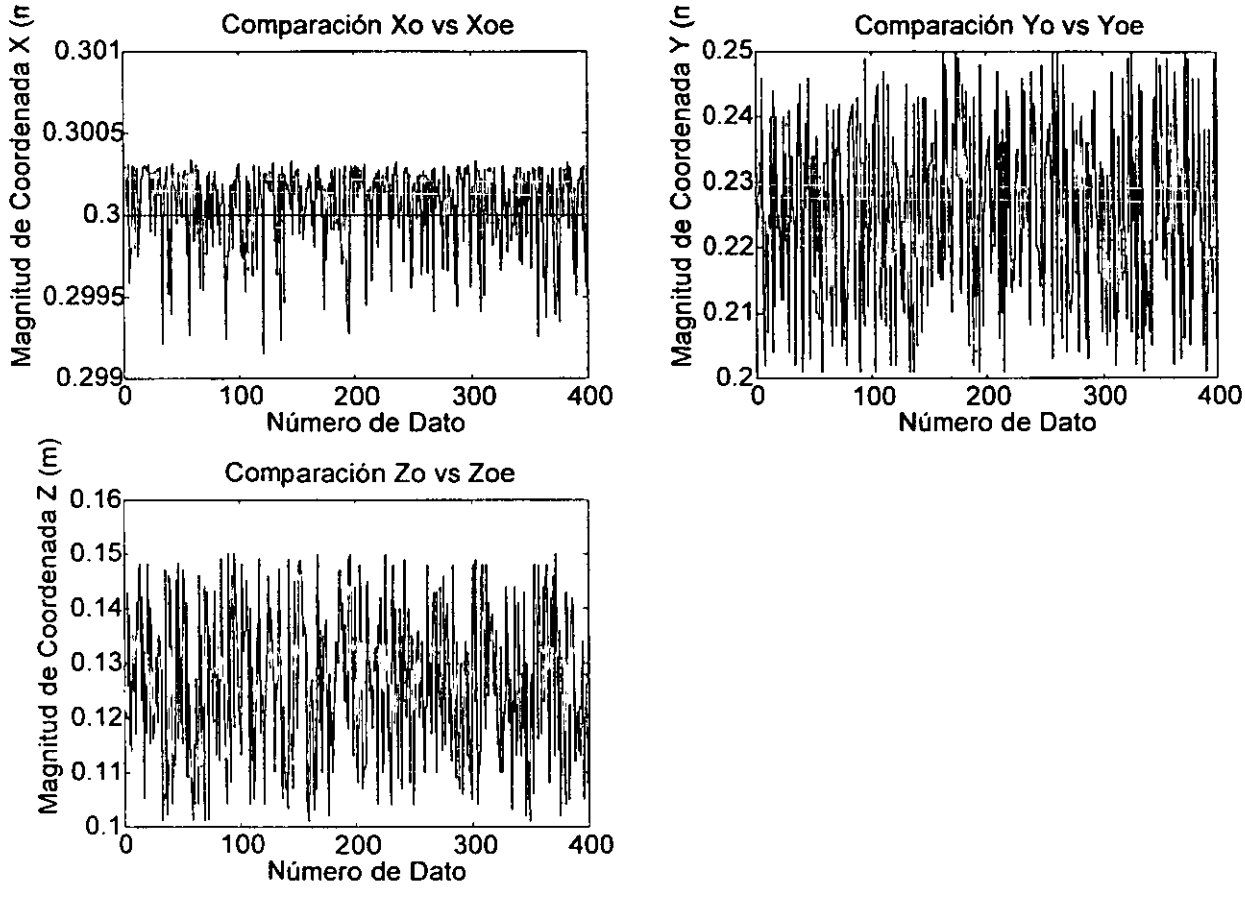

Figura 5.4: Resultados de comparación de coordenadas

Observación 5.2 *Cabe mencionar en este momento la forma en que se deben presentar los datos de entrenamiento, es decir de acuerdo con las figuras anteriores, los datos deben estar dentro del rango* $0.1 \leq dato < 1$, *de tal forma que si es necesario se debe realizar algún escalamiento de los mismos, además los datos se deben presentar de forma aleatoria*

[7] Los datos de salida para el entrenamiento de la red neuronal propuesta, se presentaron de forma aleatoria por conveniencia del proceso.

Observación 5.3 *El cuidado que se debe tener en la presentación de los datos al algoritmo de entrenamiento, se debe por un lado a la función de transferencia utilizada en la capa de salida (tangente hiperbólica) y por otro lado para proporcionar la mayor información posible para un buen proceso de generalización*

De los resultados anteriores se puede decir que la red neuronal propuesta es la alternativa esperada para realizar el mapeo de coordenadas[8], cabe mencionar que el movimiento del manipulador se realizará en un plano del espacio de trabajo.

[8] Coordenadas de imagen a coordenadas espaciales

5.4.2 Resultados de entrenamiento de ANFIS (caja de herramientas de Matlab)

Los siguientes resultados se obtuvieron utilizando los mismos datos de entrenamiento del algoritmo de retropropagación, presentado en la sección anterior, el programa utilizado para obtener estos resultados se presenta en el Apéndice B.

En la Fig. 5.5 se presentan el error de generalización de las dos coordenadas aproximadas (Y, Z)[9]

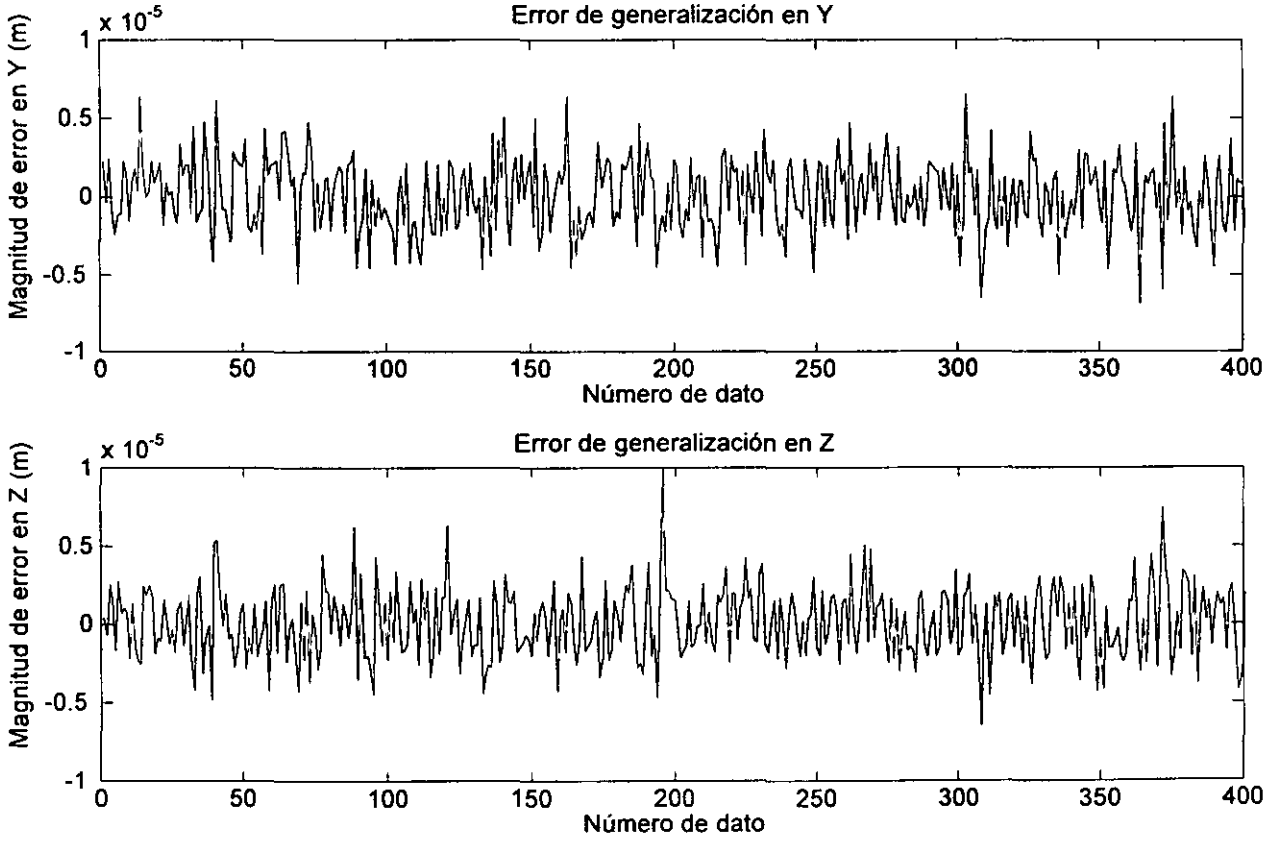

Figura 5.5: Error de generalización con ANFIS

[9]La coordenada X, es constante, ANFIS no apróxima funciones lineales

En la Fig. 5.6 se presenta una comparación de las coordenadas de entrenamiento con las coordenadas estimadas con la caja de herramientas ANFIS, de Matlab, después del entrenamiento.

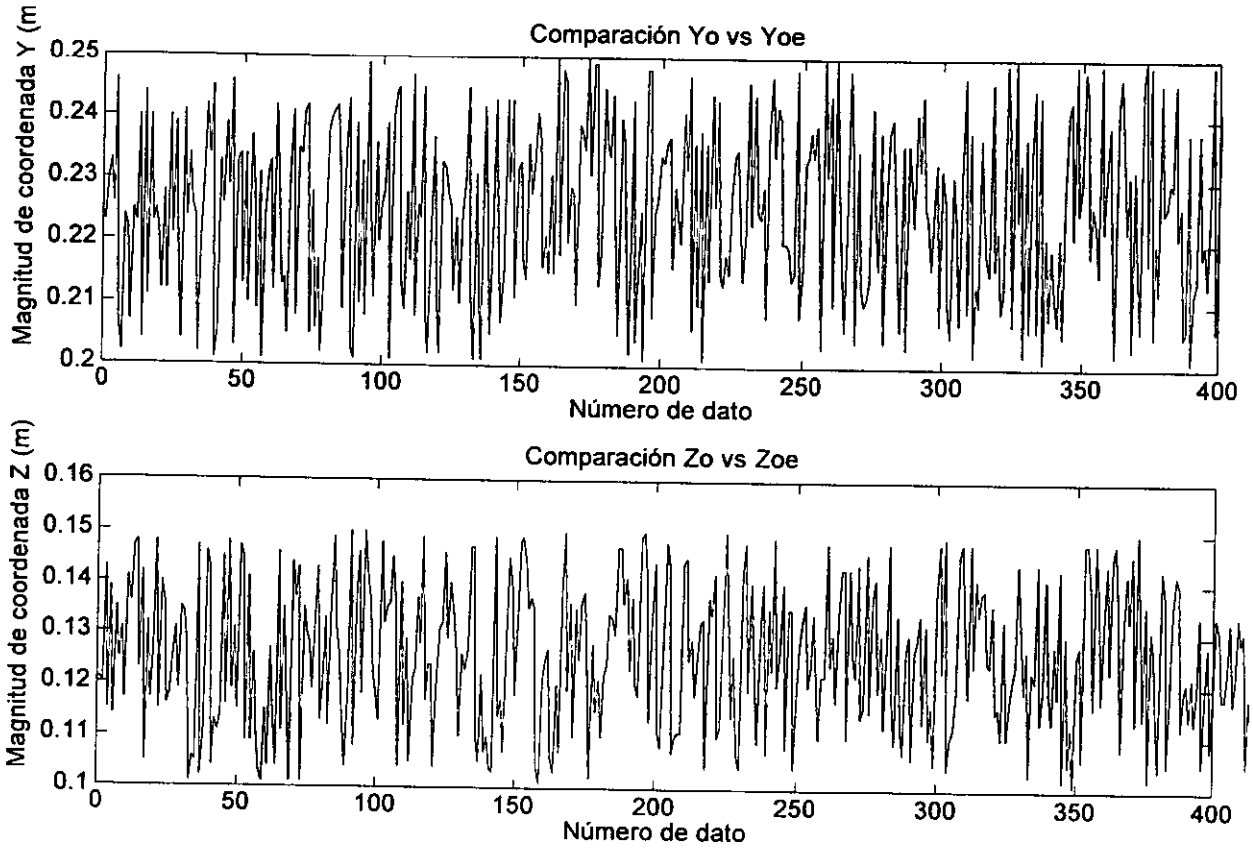

Figura 5.6: Comparación de coordenadas de entrenamiento con coordenadas estimadas con ANFIS

Cabe hacer notar que a pesar de que se graficó con diferente tipo de línea a las coordenadas de entrenamiento y a las coordenadas calculadas con ANFIS, no existe una diferencia que pueda observarse.

5.4.3 Resultados de entrenamiento de NN (caja de herramientas de Matlab)

Los siguientes resultados se obtuvieron utilizando los mismos datos de entrenamiento del algoritmo de retropropagación y la caja de herramientas de ANFIS de Matlab ("ANFIS Toolbox") presentado anteriormente, el programa utilizado para obtener estos resultados se presenta en el Apéndice B.

En la Fig. 5.7 se presentan el error de generalización de las coordenadas aproximadas (X, Y, Z)

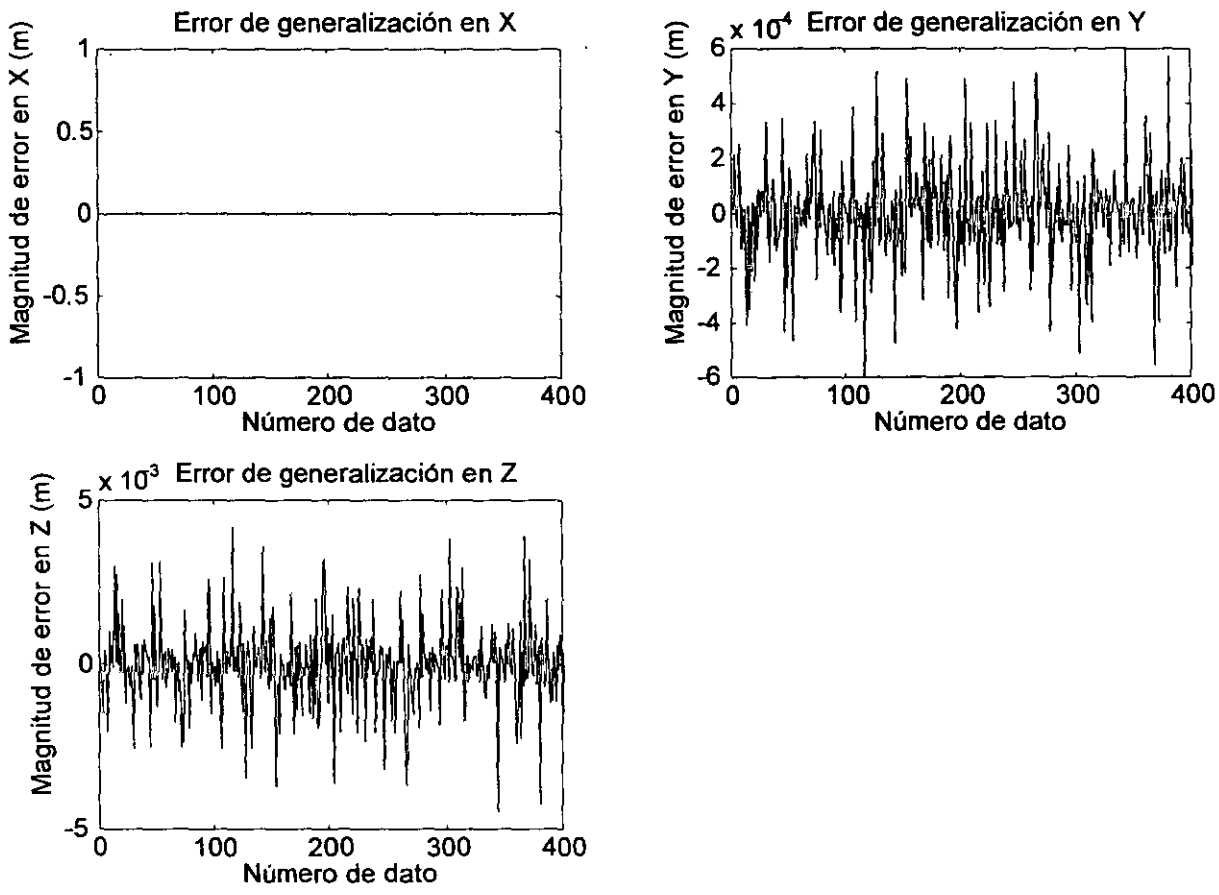

Figura 5.7: Error de generalización, caja de herramientas para redes neuronales de Matlab

En la siguiente figura se presenta una comparación de coordenadas de entrenamiento con las coordenadas estimadas utilizando la caja de herramientas para redes neuronales de Matlab, aunque se realizaron con diferente tipo de línea, la diferencia es mínima, resultado esperado dado que la magnitud del error es pequeña.

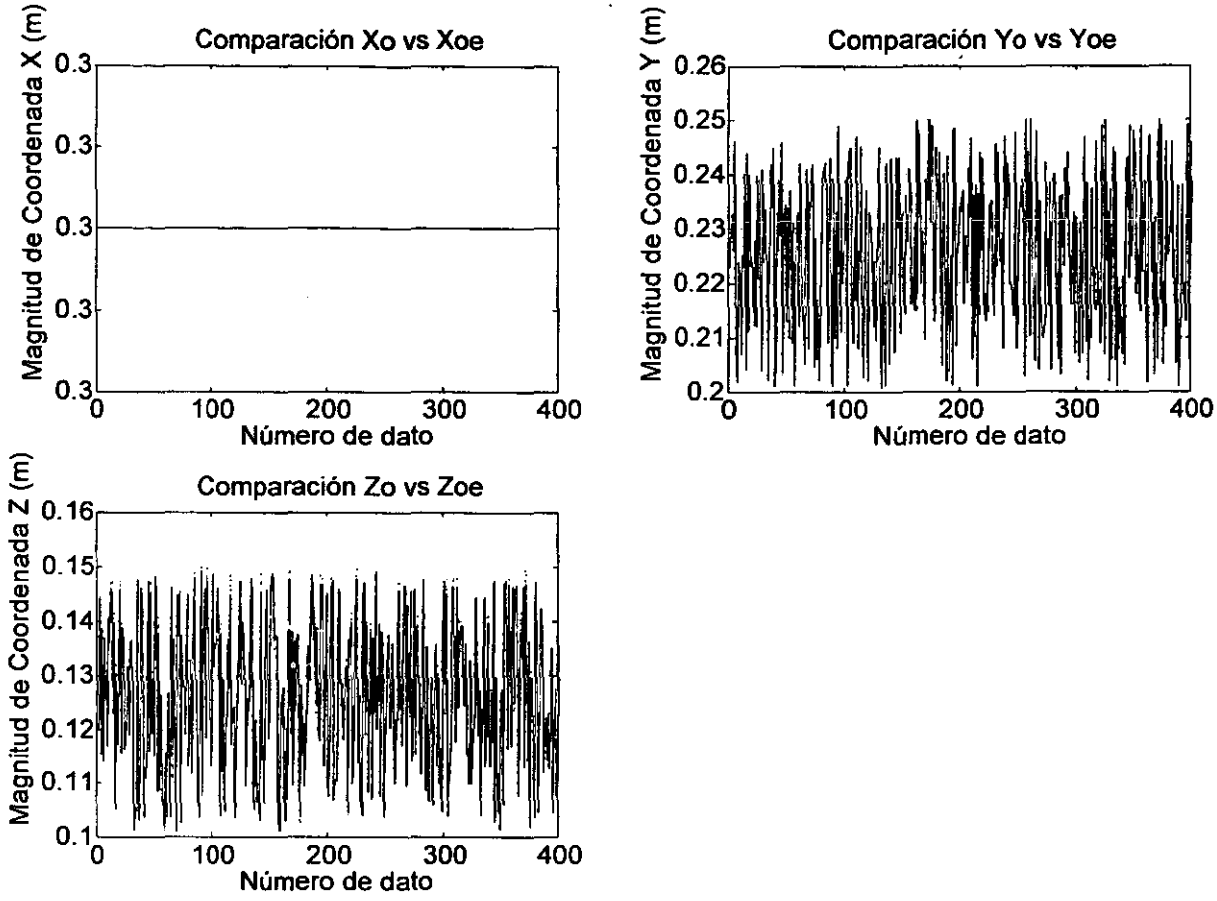

Figura 5.8: Comparación de coordenadas de simulación con coordenadas estimadas utilizando NN.

5.4.4 Comparación de resultados, algoritmo implementado BP y las cajas de herramientas ANFIS y NN de Matlab

Con el fin de verificar el desempeño del algoritmo de entrenamiento y la arquitectura de la red neuronal propuesta, se realizó una comparación[10] de los resultados de generalización de las cajas de herramientas NN (redes neuronales) y ANFIS (sistemas de inferencia adaptables neuro difusos). De estos resultados se puede concluir lo siguiente:

La caja de herramientas ANFIS de Matlab resultó con el menor error de generalización. Esta no aproxima de manera conveniente las funciones lineales y el entrenamiento como la generalización se realiza con una variable de salida a la vez, esto representa una desventaja considerable en la ejecución en línea, consumiría más tiempo de máquina. Tiempo de entrenamiento aproximado 15 (min).

[10]Se utilizaron datos de simulación, el entrenamiento se realizó en una pc pentium II 300 (MHz)

La caja de herramientas NN de Matlab, presentó menor error de generalización que el algoritmo implementado de retropropagación, pero mayor que ANFIS. En esta caja de herramientas no hay restricción al número de variables de entrada y salida para entrenamiento. Tiempo de entrenamiento aproximado 15 (min).

El algoritmo implementado de retropropagación de error, presentó el error más grande en el proceso de generalización, pero para los fines del presente trabajo se puede considerar este como una buena alternativa. Tiempo de entrenamiento aproximado 5 (hrs).

Como conclusión de esta sección se puede mencionar que la mejor alternativa para los datos de simulación resultó ser, la caja de herramientas ANFIS de Matlab. En favor del algoritmo implementado, se puede decir que es el más simple de los tres métodos de aproximación de funciones, se puede implementar en cualquier lenguaje de programación de alto nivel, en ejecución, es simplemente una multiplicación de matrices, el tiempo de generalización[11] es menor y el error de generalización se puede hacer arbitrariamente pequeño. El costo a pagar por las ventajas anteriores del algoritmo implementado, es el tiempo de entrenamiento, mucho más largo que las otras dos alternativas (ANFIS, NN).

A pesar del tiempo de entrenamiento de la red neuronal propuesta, por los argumentos anteriores se utilizará ésta para realizar el mapeo de coordenadas de imagen a coordenadas espaciales, en el presente trabajo.

[11] Tiempo de cálculo de las salidas con las entradas propuestas

5.4.5 Resultados de entrenamiento de la red neuronal con datos reales

En la siguiente figura se presentan las coordenadas espaciales de la marca ubicada en el efector final, así como también las coordenadas de imagen obtenidas utilizando la cámara y el programa de labview sctrl11.vi[12] En conjunto estas representan los datos reales[13] de entrenamiento de la red neuronal propuesta, cabe mencionar que las coordenadas de imagen (gráfica inferior derecha) representan el centroide de la marca del efector final en varias posiciones dentro del plano de trabajo.

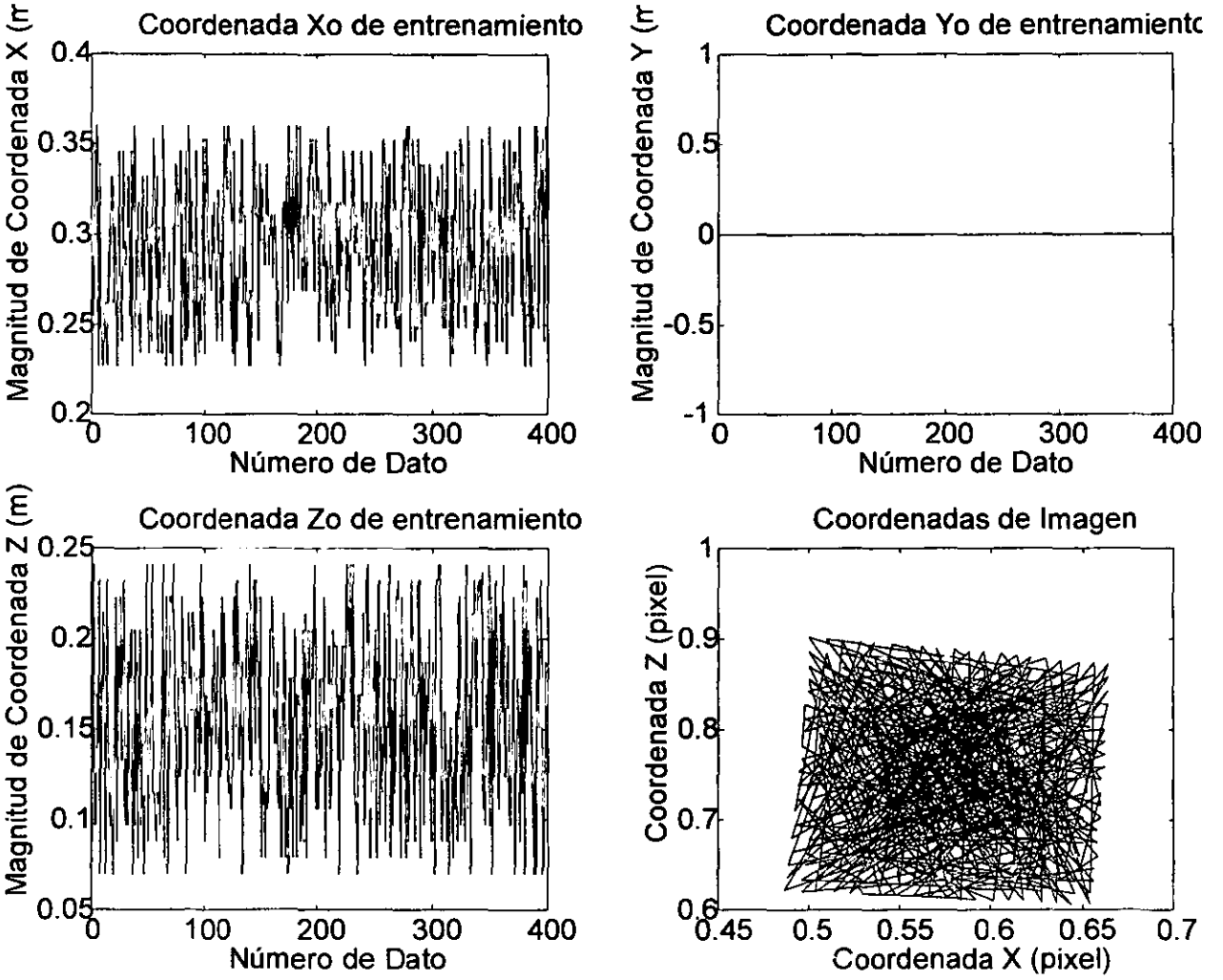

Figura 5.9: Coordenadas reales de entrenamiento

[12]Bajo petición expresa del lector se ofrece una copia electrónica por el autor (sebastianibarra@usa.net).
[13]Los coordenadas de imagen se presentan para el entrenamiento en forma aleatoria.

A continuación se presentan los resultados en forma gráfica del error de generalización de la red neuronal, para los datos reales de entrenamiento.

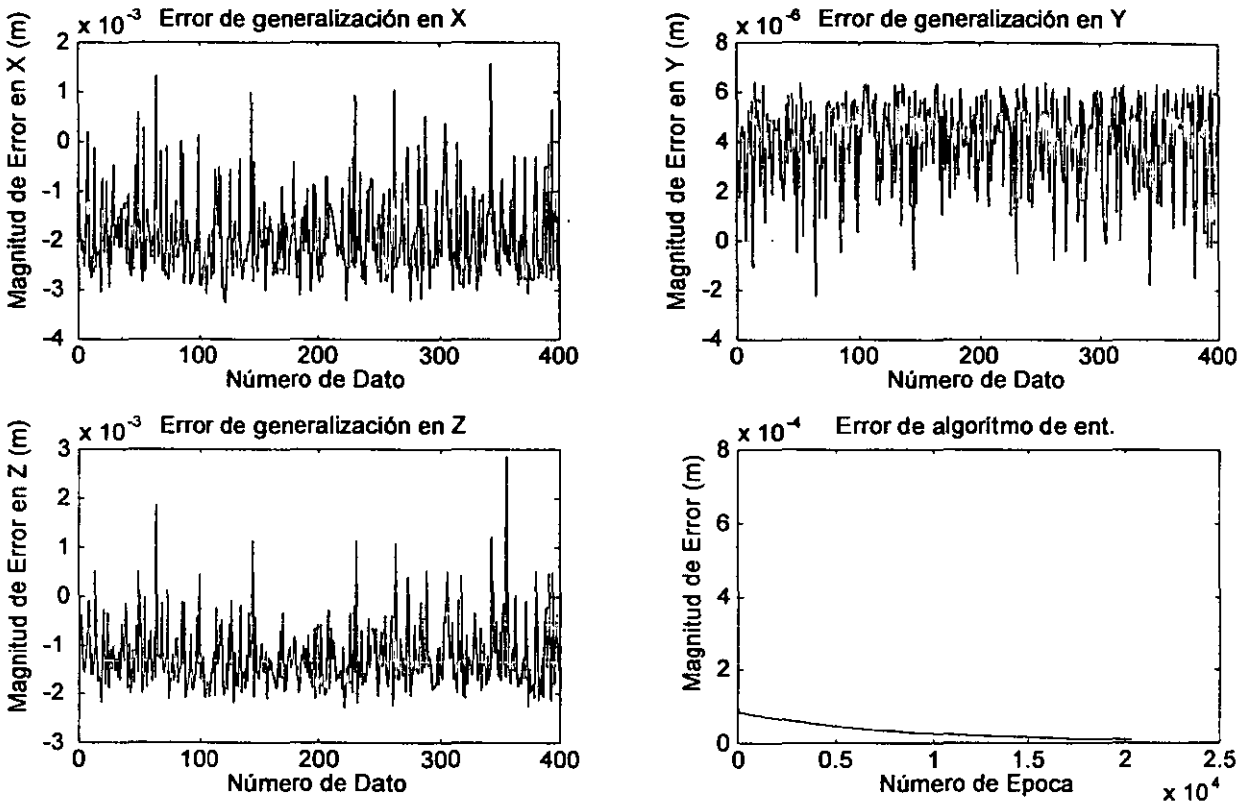

Figura 5.10: Error de generalización para datos reales

De acuerdo con los resultados anteriores (magnitud pequeña de error), se justifica la utilización de esta red para aproximar el mapeo de coordenadas de imagen contra coordenadas espaciales.

Debe comentarse en este punto que las coordenadas tanto las coordenadas espaciales como las de imagen se escalaron para realizar el entrenamiento, por lo que el error presentado anteriormente deberá ser escalado también.

En la siguiente figura se presenta la comparación de las coordenadas espaciales de entrenamiento con las coordenadas calculadas como salidas de la red neuronal, prácticamente no existe diferencia entre unas u otras, a pesar que el error objetivo de entrenamiento fué de una décima de pulgada.

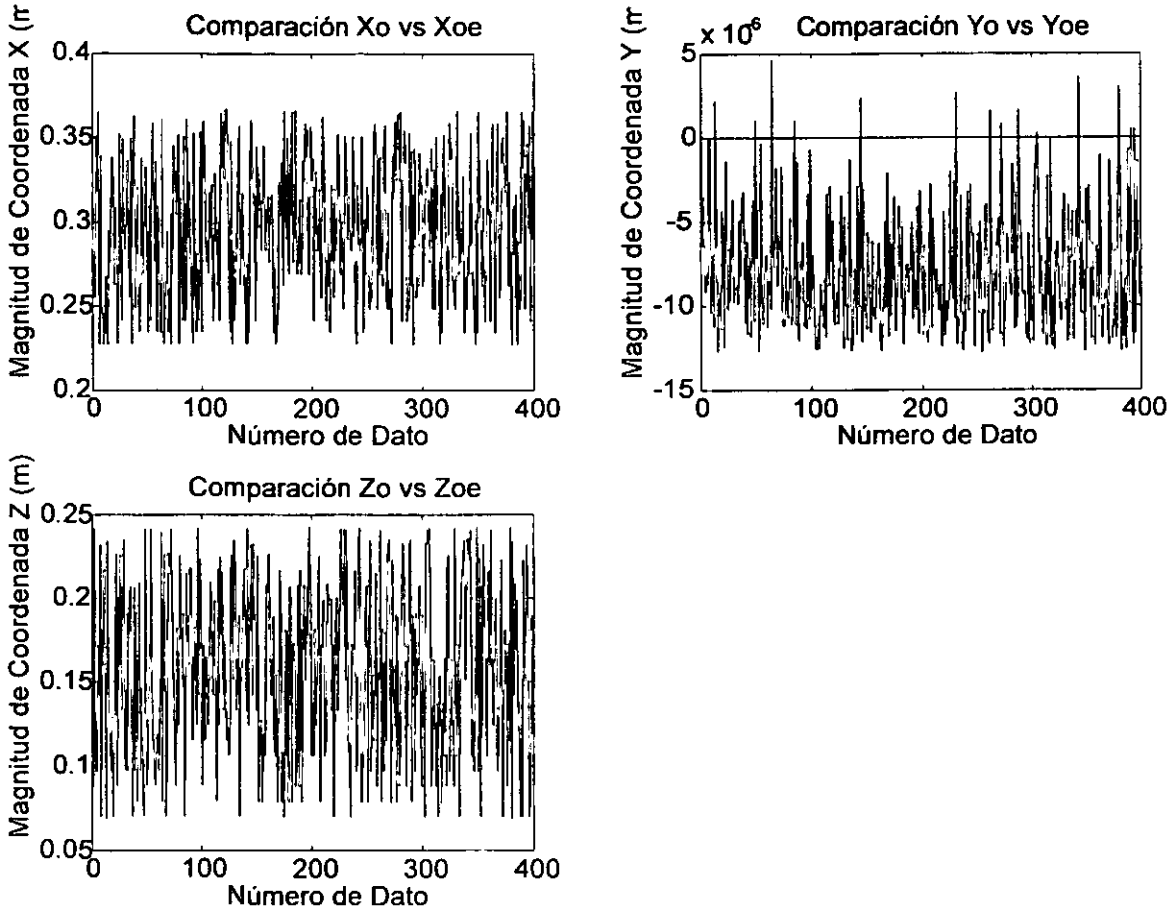

Figura 5.11: Comparación de coordenadas reales v.s calculadas

Los resultados anteriores nos permiten predecir un buen comportamiento en cuanto al mapeo de coordenadas necesario para el control del robot,.también se puede decir que se verifican los resultados de simulación con los datos experimentales y se concluye por lo tanto que podemos considerar la red neuronal propuesta como buen método de aproximación funcional para el presente trabajo.

Capítulo 6

Control de manipuladores robóticos basado en retroalimentación visual

Para lograr el objetivo de control de manipuladores robóticos con retroalimentación visual es necesario realizar las siguientes operaciones:

1. Toma de imagen.
2. Identificación del centroide de la marca en el efector final.
3. Cálculo de las coordenadas del efector final.
4. Cálculo de la cinemática inversa del robot.
5. Cálculo de las trayectorias deseadas.
6. Aplicación de la ley de control.
7. Cálculo y aplicación de los pares de entrada al robot.

Cabe mencionar en este momento que las 7 operaciones mencionadas anteriormente forman parte de un programa general realizado en Labview, del que se presenta un diagrama de bloques Fig. 6.1[1] El tiempo que tarda en ejecutarse el programa de Labview, se define como el *tiempo de muestreo*, para el presente trabajo se realizaron mediciones de este tiempo, encontrandose en el rango de 230-320 ms.El tiempo de muestreo es función de la velocidad de la computadora, el tamaño en pixels de la imagen adquirida. la posición del efector final en el plano de trabajo y el valor del umbral aplicado a la imagen. Cabe mencionar que la mayor parte de este tiempo se consume en la rutina de búsqueda del centroide.

[1]Bajo petición expresa del lector, el autor ofrece una copia electrónica del programa (sebastianibarra@usa.net).

6.1 Toma de imágenes

Toma de imagen. La operación de adquisición de imágenes se realizo de forma automática y cíclica. No se utilizó ninguna técnica especial de iluminación. En el panel frontal del programa se presenta una barra de control que permite fijar el valor de umbral utilizado para el preprocesamiento de la imagen, esto permite soportar un ambiente sin iluminación especial, de tal forma que el desempeño de la rutina de identificación de centroide de la marca puede mejorarse cambiando el nivel de umbral con la barra antes mencionada.

6.2 Identificación de centroide

Identificación del centroide de la marca en el efector final. Con la ayuda de las librerías de Imaq vision (Imaq vision versión 4.1.1) bloque "find.vi"[2] , se realiza una búsqueda en la imagen de la marca en el efector final, para realizar la identificación se generó una imagen patrón, tomada realizando un acercamiento al efector final, de tal forma que prácticamente se tiene un círculo blanco en fondo negro. Con esta marca patrón se realiza una exploración en la imagen recién adquirida, esta exploración se realiza en toda la imagen, iniciando en la esquina superior izquierda de la imagen y terminando en la esquina inferior derecha, una vez terminada la exploración se genera dentro del programa un reporte del número de coincidencias y los centroides de cada una de las marcas encontradas.

6.3 Cálculo de coordenadas

Cálculo de las coordenadas del efector final. Una vez determinadas las coordenadas de imagen del centroide de la marca del efector final y utilizando las matrices de pesos determinadas durante el proceso de entrenamiento de la red neuronal propuesta, se realiza una simple multiplicación de este vector de coordenadas de imagen por una matriz de pesos, lo cual resulta en un vector que a su vez será premultiplicado por la otra matriz de pesos, para finalmente obtener las coordenadas espaciales de la marca del efector final. Es necesario hacer notar la forma simple y rápida de obtener las coordenadas espaciales del efector final, esto por supuesto es posible por la utilización de las redes neuronales como método de aproximación de funciones. Esto representa una ventaja, pues una de las principales limitaciones en la aplicación de retroalimentación visual es el tiempo de muestreo tan grande.

6.4 Cálculo de la cinemática inversa

Cálculo de la cinemática inversa del robot. Con el valor de las coordenadas espaciales del efector final dentro del mismo programa de Labview (sctrl11.vi) se realiza el cálculo de los

[2]Dentro del programa general de Labview.

angulos de las articulaciones 2 y 3. Utilizando para esto las relaciones que se obtuvieron en el Capítulo 4 (4.4-4.10) Cabe mencionar que se fijaron las articulaciones 1, 4 y 5, por limitaciones de equipo en el laboratorio de robótica el presente trabajo sólo se realizan movimientos del efector final en un plano del espacio de trabajo, es decir se fijó la coordenada espacial "Y" y el movimiento se realizó en el plano X vs Z.

6.5 Cálculo de las trayectorias deseadas

Cálculo de las trayectorias deseadas. Utilizando el tiempo de muestreo se generan, en línea, las trayectorias deseadas para la el efector final[3].

$$X = Am\sin(2\pi ftm) + X_o \tag{6.1}$$

$$Z = Am\sin(2\pi ftm) + Z_o \tag{6.2}$$

donde Am es la amplitud de la senoidal, f es la frecuencia de la señal, tm es el tiempo de muestreo y X_o, Z_o son corrimientos de posición en ambas coordenadas. Con estas coordenadas espaciales del efector final y las relaciones de la cinemática inversa (4.10 , 4.7) se generan los ángulos de las articulaciones 2 y 3, de tal forma que se tienen las trayectorias deseadas para las articulaciones del robot.

6.6 Aplicación de la ley de control

La estrategia de control utilizada en el presente trabajo es la técnica de control descentralizado [19]

La ley de control es la siguiente:

$$\tau = -Ks + w \tag{6.3}$$

K_1, K_2 > 0 forman la matriz K de ganancia constante, λ_1, $\lambda_2 > 0$ forman la matriz λ de ganancia contante y δ, $\epsilon_i > 0$, $i = 1, 2$; son parámetros de diseño. Los parámetros del controlador utilizados son: $K_1 = 0.59$, $K_2 = 93$, $\lambda_1 = 0.024$, $\lambda_2 = 0.034$, $\epsilon_1 = \epsilon_2 = 0.1$ y $\delta = 2$.

[3]Estas ecuaciones definen un desplazamiento en línea recta del efector final, en el plano de trabajo.

6.7 Aplicación de los pares de control al robot

Cálculo y aplicación de los pares de entrada al robot. Una vez que se calcula el par de entrada al robot se realiza la conversión par a voltaje. Esto se realiza utilizando un módulo de labview, del programa sctrl11.vi. Cabe mencionar que por razones de seguridad para el robot, se limita el desplazamiento de las articulaciones, incluyendo en el programa sctrl11.vi, un bloque de saturación para los ángulos de las articulaciones, evitando que las articulaciones salgan de su límite y el robot sufra algún daño.

6.8 Realización de experimentos finales

En la siguiente figura se presenta un diagrama de bloques de las operaciones realizadas para lograr la retroalimentación visual y el control del manipulador robótico, es decir del programa de Labview.

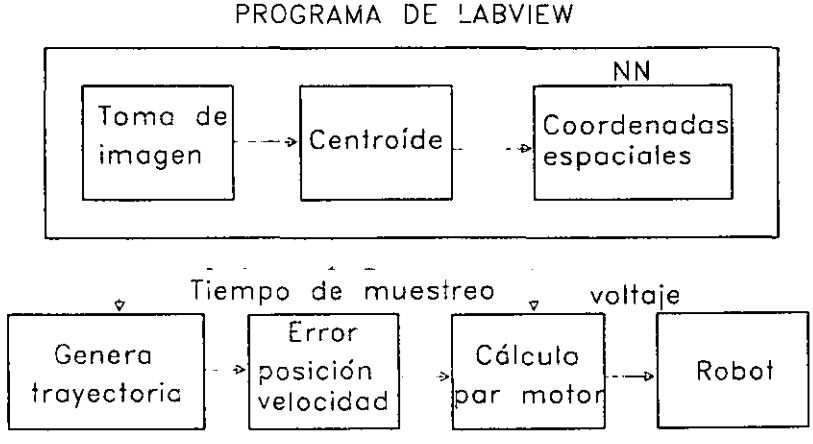

Figura 6.1: Operaciones para retro - visual

6.8.1 Condiciones del experimento

El movimiento del robot se restringe al desplazamiento en un plano del espacio del efector final, es decir, en el plano X, Z, Y=0.

El plano de movimiento se restringe, a una área de 15(cm)x15(cm).

La velocidad de movimiento queda restringida por el tiempo de muestreo, es decir, por el tiempo que tarde en ejecutarse el programa de labview[4].

[4]sctrl11.vi, a petición expresa del lector se ofrece una copia electrónica del programa.

La trayectoria deseada del efector final es una línea recta en un plano del espacio de trabajo del efector final.

El período para la señal de seguimiento se fija en 50(seg).

Equipo utilizado:

Robot de 5 grados de libertad CRS PLUS A255, con características principales:
5 grados de libertad.
Máxima carga en efector final 2 (Kg).
Alcance de 560 (mm).
Repetibilidad de 0.05 (mm).
Controlador C500
Cámara de video CCD Sony XC-75 cuyas características son:
Resolución horizontal 560 TV líneas.
Sensibilidad 400lx, F4
Relación señal a ruido 54 (dB).
Función de disparo externo (S-DONPISHA).
Tamaño del lente 1/2" de díametro.
Sincronía externa HD/HV.
Tarjeta de adquisición de imagenes IMAQ PCI/PXI 1408, con características:
Tarjeta de adquisición de imagenes monocromáticas.
Tipo de bus PCI.
Formatos: RS-170, CCIR, NTSC y PAL.
Transferencia directa a memoria.
Entradas de video 4.
Impedancia de entrada 75 (Ω).
Computadora pentium III (clon), con características:
Velocidad 800 Mhz.
Memoria 128 MB.
Disco duro 2GB.
Sistema operativo windows 98.
Programa Labview, versión 5.1.1 (propósito general de laboratorio).
Programa Matlab versión 5.3 (propósito general de laboratorio).
Programa de utilerias de visión IMAQ Vision, versión 4.1.1.
Programa de Flexmotion, versión 2.1.

En la siguiente fotografía se presenta la cámara y el robot de 5 GDL (robot de la derecha), utilizado en este trabajo[5].

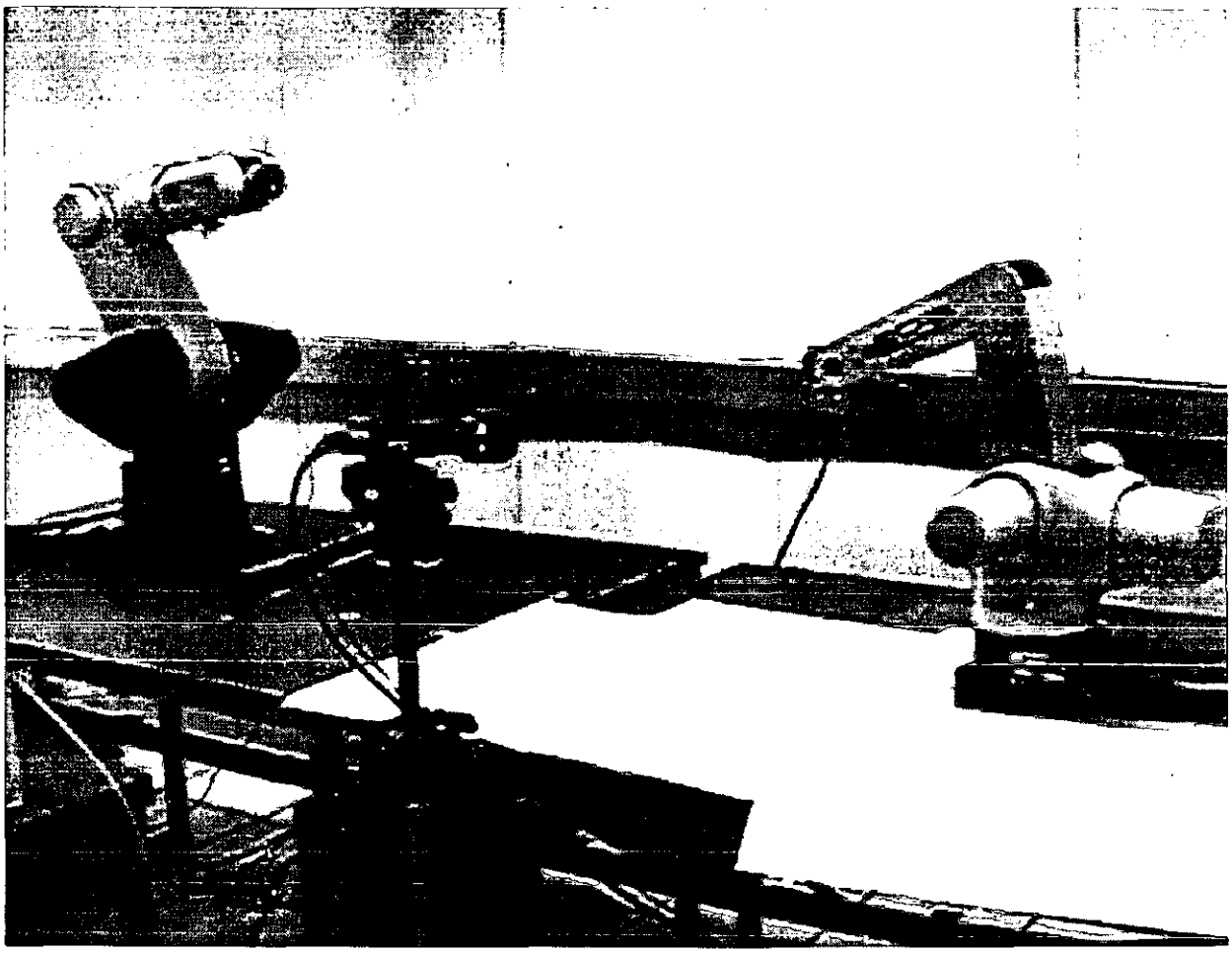

Figura 6.2: Equipo de trabajo, cámara y robot

[5]Esto es parte del equipo con el que cuenta el laboratorio de robótica de la DEPFI - UNAM.

6.8.2 Realización de experimentos

Se realizó una serie de experimetos para llegar al valor de las matrices de ganancias y los valores de los parámetros de diseño del controlador. Los parámetros del controlador quedaron como sigue: $K_1 = 0.59$, $K_2 = 93$, $\lambda_1 = 0.024$, $\lambda_2 = 0.034$, $\epsilon_1 = \epsilon_2 = 0.1$ y $\delta = 2$.

En la siguiente figura se presenta el resultado del experimento, donde se observa el comportamiento de las coordenadas deseadas contra las coordeenadas reales.

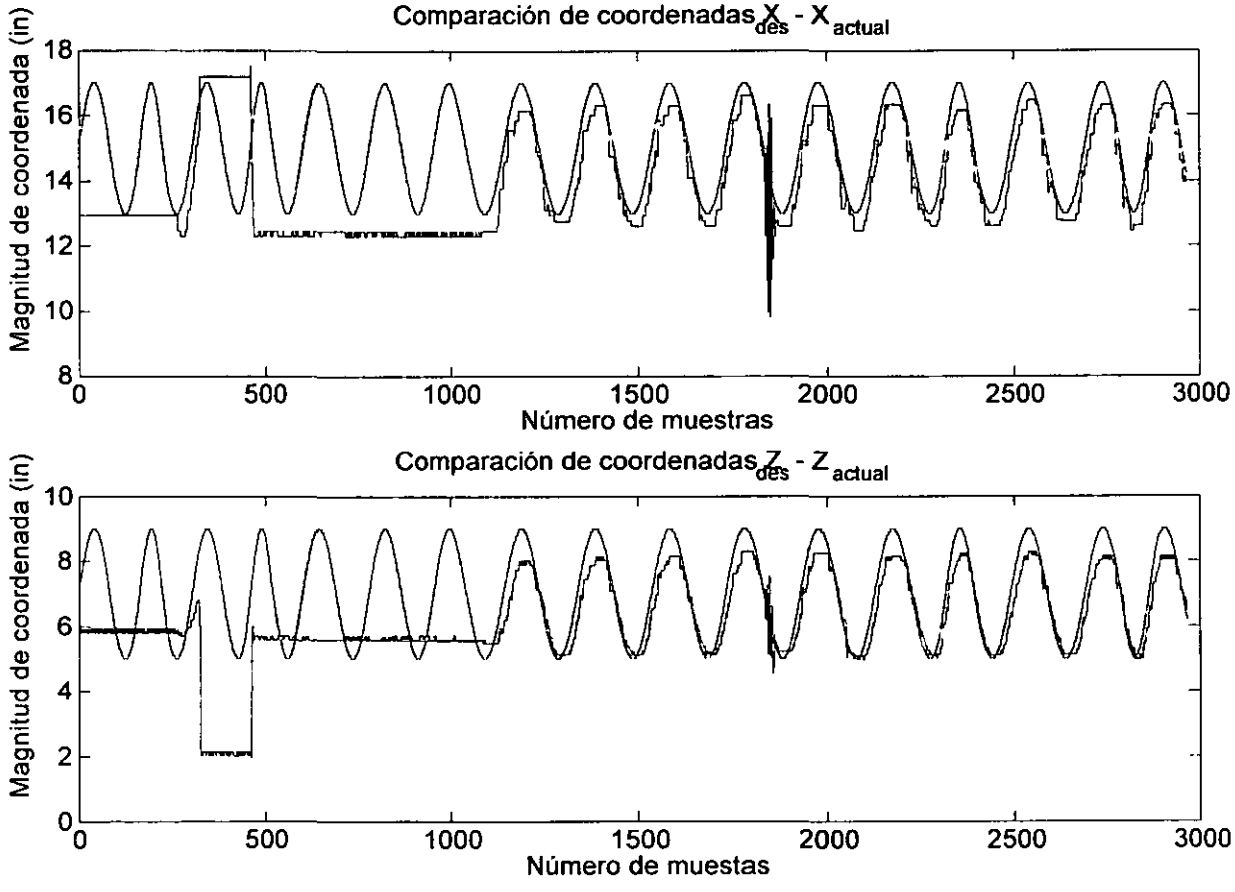

Figura 6.3: Coordenadas espaciales deseadas contra coordenadas reales.

Cabe mencionar que el controlador se encendió entre 1100 y 1200 de período de muestreo, por lo que la parte inicial de la gráfica sólo se incluye como una condición inicial del manipulador.

En la siguiente figura se presenta la gráfica de comparación de los ángulos de articulación, es decir, ángulos de junta deseados contra ángulos de junta reales.

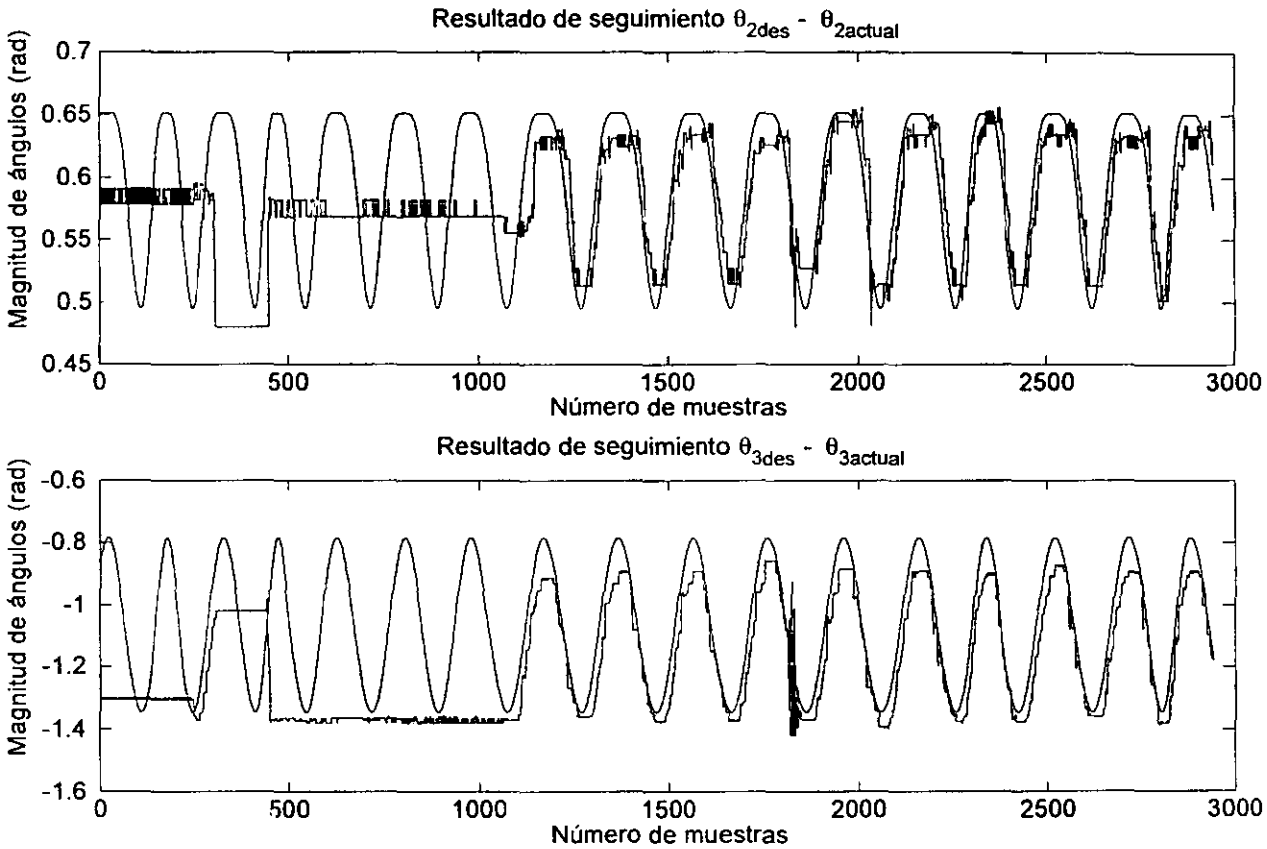

Figura 6.4: Ángulos de junta deseados contra ángulos de junta reales

Cabe mencionar nuevamente que el controlador se encendió despues del período 1000, por lo que la parte inicial de la gráfica sólo representa la condición inicial del manipulador.

En la siguiente figura se presentan los voltajes aplicados a los motores de las juntas 2 y 3, para provocar el seguimiento de la trayectoria propuesta, estos se obtuvieron con la ley de control antes discutida.

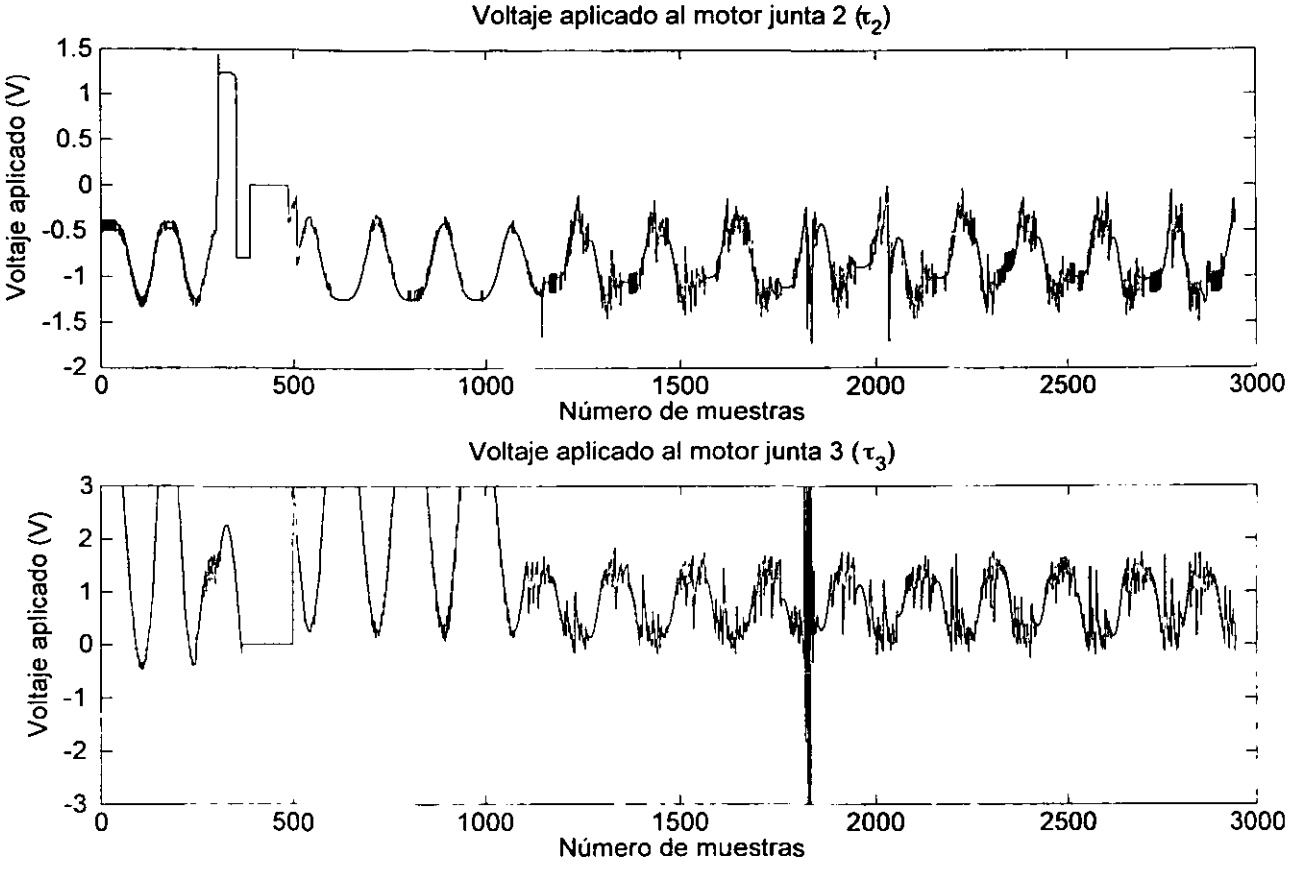

Figura 6.5: Voltajes aplicados a los motores de juntas 2 y 3

Cabe hacer la aclaración que el controlador toma el control del robot depués del período de muestreo 1000, en la parte inicial de la gráfica, el robot no ha realizado movimiento alguno y sólo es una condición inicial.

Observando la gráfica del voltaje aplicado al motor de la junta 3, se observa la respuesta a una perturbación[6] entre el período de muestreo 2000 y 2500. Se considera que esta perturbación se presenta debido a la trayectoria deseada (línea recta) la cual fue generada por dos senoidales cuyá velocidad será máxima en el cruce por cero (esto coincide con los extremos de la línea recta) considerando además que el tiempo de muestreo es grande, por lo tanto esta perturbación es generada por la misma trayectoria propuesta, cabe mencionar que no se considerarón las propiedades de la trayectoria propuesta para este experimento. Se

[6]Esta pertubación se presenta en todas las gráficas finales.

puede observar en la gráfica que después de algunos períodos de muestreo el control obliga al sistema a continuar con el seguimiento de la trayectoria, verificando con esto la robustez del control propuesto.

Capítulo 7

Conclusión y trabajo futuro

7.1 Conclusión.

El objetivo del presente trabajo "Control Visual de un Robot Basado en Retroalimentación Visual" se cumplió de forma satisfactoria, de acuerdo con los resultados experimentales presentados en forma gráfica.

La red neuronal propuesta para realizar el mapeo de coordenadas de imagen a coordenadas espaciales, funcionó de acuerdo a lo esperado y se comprueba este método de aproximación de funciones no lineales en un conjunto compacto. Así mismo se verifíco la operación del algoritmo de entrenamiento implementado [2] Se debe considerar a las redes neuronales como una buena alternativa para su aplicación, en general a sistemas de control, principalmente dónde.sea muy difícil la obtención del modelo del sistema. Se debe mencionar en este momento que es necesario tener un conocimiento preciso del comportamiento del sistema donde se aplicarán las redes neuronales, pues, para este método de aproximación de funciones, es necesario que exista una relación funcional de naturaleza no lineal entre los datos de entrada-salida del sistema. El espacio de trabajo del manipulador, se restringió a un plano perpendicular a la cámara, en caso de haber contado con otra cámara la restricción sobre el espacio de trabajo sería mínima, es decir se hubiera realizado visión estéreo, por lo que el efector final se podría haber desplazado en un volúmen de trabajo en lugar de un plano de trabajo, como fue este caso.

Verificando el desempeño del controlador, se puede decir que el tiempo de muestreo tan grande provocó fallas de seguimiento, pues es claro que a menor tiempo de muestreo el seguimiento es mejor, cabe mencionar también que el período de muestreo no fué constante y a pesar de esto el controlador funcionó adecuadamente. La perturbación presente en las gráficas de resultados se debe a la mala elección de la trayectoria a seguir, pues es posible eliminar dicha perturbación eligiendo otra trayectoria.

Otras contribuciones al error de seguimiento son las siguientes: La red neuronal se entrenó para un error objetivo de una décima de pulgada, se realizaron movimientos considerables de la cámara, es decir la posición de entrenamiento no fue exactamente la misma que la posición del experimento final y no se utilizó alguna técnica de iluminación especial en el

ambiente de trabajo experimental, lo que necesariamente ocasiona errores en el proceso de identificación de las coordenadas del centroide según se comprobó en los experimentos. De tal forma que si se toma en cuenta lo anterior necesariamente redundará en una mejoría en el desempeño del controlador.

Gran parte del período de muestreo, más del 95%, se debe al procesamiento de la imagen y al proceso de indentificación del centroide. La utilización de la red neuronal en línea implica prácticamente 2 multiplicaciones de matrices, por lo que la falla en el desempeño del sistema total de control, en este caso no se debe atribuir al uso de la red neuronal ni a la ley de control utilizada.

7.2 Trabajo futuro

Se propone como trabajo futuro la realización de control visual de robots utilizando técnicas de visión estéreo y redes neuronales estáticas o dinámicas, considerando que se cuenta con el equipo básico de laboratorio y que las redes neuronales son una buena alternativa en sistemas de control.

La experiencia ganada con el presente trabajo se debe aprovechar también en realizar control coordinado de robots utilizando retroalimentación visual.

La red neuronal propuesta en el presente trabajo se puede entrenar con un error objetivo de milésimas de pulgada y ampliar el número de datos, para cubrir mejor el espacio de trabajo, de ésta forma se reduciría el error de seguimiento de trayectoria y por tanto se observaría un mejor desempeño del sistema de control.

Se debe buscar la posibilidad de mejorar las rutinas de identificación de centroides, de tal forma que utilicen menos tiempo de procesamiento y de esta forma reducir el tiempo de muestreo, se debe mencionar también que existen mejores librerías de programas de Labview para aplicación en sistemas de visión, por lo que serían un buen punto de partida.

Apéndice A

A.1 Derivación del algoritmo rápido de retropropagación de error.

El algoritmo rápido de retropropagación es derivado en este apéndice minimizando la función objetivo $G_k(\lambda)$

$$G_k(\lambda) = \lambda\Sigma_{i=1}^{n0}\varphi_2(y_{i,k} - \hat{y}_{i,k}) + (1-\lambda)\Sigma_{i=1}^{n0}\varphi_1(y_{i,k} - \hat{y}_{i,k})$$

La ecuación de actualización de los pesos sinápticos w_{pq} es la siguiente.

$$w_{p,k} - w_{p,k-1} = -\alpha\frac{\partial G_k(\lambda)}{\partial w_p} = -\alpha\Sigma_{i=1}^{n0}[\lambda\frac{d\varphi_2(e_{i,k})}{de_{i,k}} + (1-\lambda)\frac{d\varphi_1(e_{i,k})}{de_{i,k}}]\frac{\partial e_{i,k}}{\partial w_p} \tag{A.1}$$

De la definición de $\hat{y}_{i,k} = \sigma(\Sigma_{j=1}^{nh} w_{i,j}\hat{h}_{j,k})$.

$$\frac{\partial e_{i,k}}{\partial w_p} = -\frac{\partial \hat{y}_{i,k}}{\partial w_p} = -\frac{d\sigma(\bar{y}_{i,k})}{d\hat{y}_{i,k}}\frac{\partial}{\partial w_p}(\Sigma_{j=1}^{nh} w_{i,j}\hat{h}_{j,k}) = -\acute{\sigma}(\bar{y}_{i,k})\hat{h}\delta_{ip} \tag{A.2}$$

La combinación de (A.1) y (A.2) nos da.

$$w_{p,k} = w_{p,k-1} + \alpha\epsilon_{p,k}^0(\lambda)\hat{\mathbf{h}}_k \tag{A.3}$$

Donde

$$\epsilon_{p,k}^0(\lambda) = \acute{\sigma}(\bar{y}_{p,k})[\frac{d\varphi_2(e_{p,k})}{de_{p,k}} + (1-\lambda)\frac{d\varphi_1(e_{p,k})}{de_{p,k}}] \tag{A.4}$$

La ecuación para los pesos sinápticos v_{pq}, se pueden obtener de forma similar. De la definición de $G_k(\lambda)$

$$v_{p,k} - v_{p,k-1} = -\alpha\frac{\partial G_k(\lambda)}{\partial v_p} = -\alpha\Sigma_{i=1}^{n0}[\frac{d\varphi_2(e_{i,k})}{de_{i,k}} + (1-\lambda)\frac{d\varphi_1(e_{i,k})}{de_{i,k}}]\frac{\partial e_{i,k}}{\partial e_{i,k}} \tag{A.5}$$

Tomando en cuenta la definición de $\hat{y}_{i,k} = \sigma(\Sigma_{j=1}^{nh} w_{i,j}\hat{h}_{j,k})$.

$$\frac{\partial e_{i,k}}{\partial v_p} = -\frac{\partial \hat{y}_{i,k}}{\partial v_p} = -\frac{d\sigma(\bar{y}_{i,k})}{d\bar{y}_{i,k}}\frac{\partial}{\partial v_p}(\Sigma_{j=1}^{nk} w_{i,j}\hat{h}_{j,k}) = -\acute{\sigma}(\bar{y}_{i,k})\Sigma_{j=1}^{nh} w_{i,j}\frac{\partial \hat{h}_{j,k}}{\partial v_p} \tag{A.6}$$

Finalmente, la definición de $\hat{h}_{j,k} = \rho(\bar{h}_{j,k}) = \rho(x_k^* v_j) = \rho(\Sigma_{l=0}^{ni} v_{j,l} x_{l,k})$, nos da

$$\frac{\partial \hat{h}_{j,k}}{\partial v_p} = \frac{d\rho(\bar{h}_{j,k})}{\partial \bar{h}_{j,k}} \frac{\partial}{\partial v_p}(\Sigma_{l=1}^{ni} v_{jl} x_{l.k}) = \acute{\rho}(\bar{h}_{j,k}) x_k \delta_{jp} \tag{A.7}$$

La combinación de (A.5), (A.6) y (A.7) nos dan la siguiente ecuación de actualización de los pesos sinápticos v_{pq}

$$v_{p,k} = v_{p,k-1} + \alpha \epsilon_{p,k}^h(\lambda) x_k \tag{A.8}$$

donde:

$$\epsilon_{p,k}^h(\lambda) = \acute{\rho}(\bar{h}_{p,k}) \Sigma_{i=1}^{n0} \acute{\sigma}(\bar{y}_{i,k}) [\lambda \frac{d\varphi_2(e_i,k)}{de_{i,k}} + (1-\lambda) \frac{d\varphi_1(e_i,k)}{de_{i,k}}] w_{ip} \tag{A.9}$$

Si la salida de la red es analógica[1], $\sigma(x) = x$ y $\sigma'(x) = 1$, mientras que $\rho(x) = \tanh(x)$ y $\rho'(x) = 1 - \rho^2(x)$.

La medida de discrepancia $\phi_1(\cdot)$ es dada por $\phi_1(x) = (\frac{1}{\beta}) \ln(\cosh(\beta x))$, se puede verificar fácilmente $\phi_1'(x) = \tanh(\beta x)$. de la misma forma $\phi_2(x) = (\frac{1}{2}) x^2$ y $\phi_2'(x) = x$.

En este caso, los pesos sinápticos pueden ser actualizados por (A.3), donde:

$$\epsilon_{p,k}^0(\lambda) = \lambda(y_{p,k} - \hat{y}_{p,k}) + (1-\lambda) \tanh[\beta(y_{p,k} - \hat{y}_{p,k})] \tag{A.10}$$

Así mismo los pesos sinápticos $v_{p,q}$, pueden actualizarse por utilizando (A.8), donde:

$$\epsilon_{p,k}^h = (1 - \hat{h}_{p,k}^2) \Sigma_{i=1}^{n0} \epsilon_{i,k}^0(\lambda) w_{i,p} \tag{A.11}$$

y $\epsilon_{p,k}^0(\lambda)$, es dada por (A.10).

Si la salida de la red es binaria, $\sigma(x) = \tanh(x)$ y $\acute{\sigma}(x) = 1 - \sigma^2(x)$. De la misma forma, $\rho(x) = \tanh(x)$ y $\rho'(x) = 1 - \rho^2(x)$, la medida de discrepancia $\phi_1(\cdot)$ la cual corresponde a la p-ésima unidad es dada por $\phi_1(e_{p,k}) = y_{p,k} e_{p,k}$. Claramente, $\frac{d\phi_1(e_{p,k})}{de_{p,k}} = y_{p,k}$, de la misma forma $\phi_2(x) = (\frac{1}{2}) e_{p,k}^2$ y $\phi_2'(x) = e_{p,k}$. En este caso, los pesos sinápticos $w_{p,k}$ pueden actualizarse utilizando (A.3), donde:

$$\epsilon_{p,k}^0(\lambda) = (1 - \hat{y}_{p,k}^2)(y_{p,k} - \lambda \hat{y}_{p,k}) \tag{A.12}$$

Finalmente, la ecuación de actualización para los pesos sinápticos $v_{p,q}$ es dada por (A.8). donde:

$$\epsilon_{p,k}^h(\lambda) = (1 - \hat{h}_{p.k}^2) \Sigma_{i=1}^{n0} \in_{i,k}^0 (\lambda) w_{i,p} \tag{A.13}$$

Donde $\epsilon_{i,k}^0(\lambda)$, es dado por (A.12).

[1]Si la salida es de naturaleza analógica. puede tomar cualquier valor, mientras que la red con salida binaria puede tomar sólo 0 y 1.

La generalización del análisis anterior proporciona el algoritmo rápido de retropropagación, para el entrenamiento de redes neuronales, con más de una capa de unidades ocultas.

Los estimados proporcionados por esta red estan dados por:

$$\hat{y}_{i,k}(\lambda) = \sigma(\Sigma_{j=0}^{ni} w_{ij}\hat{h}_{j,k}^{(1)}) \qquad \forall\ i = 1, 2, ..., n_0 \tag{A.14}$$

donde la salida de las unidades ocultas sería:

$\hat{h}_{j,k}^{(r)} = \rho(\Sigma_{l=0}^{n_{r+1}} v_{jl}^{(r)}\hat{h}_{l,k}^{(r+1)})$, $\quad r = 1, 2, \cdots, L-1$, L representa el número de unidades ocultas.

La ecuación de actualización de los pesos sinápticos w_{pq}, puede obtenerse de (A.13) para $\hat{h}_k = \hat{h}_k^{(1)}$, como sigue:

$$w_{p,k} = w_{p,k-1} + \alpha\ \epsilon_{p,k}^{0}\ (\lambda)\hat{h}_k^{(1)} \tag{A.15}$$

donde $\epsilon_{p,k}^{0}(\lambda)$ es dado por (A.10), si la salida de la red es analógica y por (A.12), si la salida de la red consiste de elementos binarios. Los pesos sinápticos $v_{pq}^{(r)}$, $r = 1, 2, \cdots, L$ pueden ser actualizados minimizando la función objetivo

$G_k(\lambda) = \lambda\Sigma_{i=0}^{n0}\phi_2(y_{i,k} - \hat{y}_{i,k}) + (1-\lambda)\Sigma_{i=1}^{n0}\phi_1(y_{i,k} - \hat{y}_{i,k})$, a tráves de la ecuación

$$v_{pq}^{(r)} - v_{p,k-1}^{(r)} = -\alpha\frac{\partial G_k(\lambda)}{\partial v_p^{(r)}} = -\alpha\Sigma_{i=1}^{n0}[\lambda\frac{d\phi_2(e_{i,k})}{de_{i,k}} + (1-\lambda)\frac{d\phi_1(e_{i,k})}{de_{i,k}}]\frac{\partial e_{i,k}}{\partial v_p^{(r)}} \tag{A.16}$$

Se puede demostrar que de la ecuación anterior se puede obtener la ecuación de actualización de los pesos sinápticos, como sigue:

$$v_{p,k}^{(1)} = v_{p,k\ -1}^{(1)} + \alpha\epsilon_{p,k}^{(1)}(\lambda)\hat{h}_k^{(2)} \tag{A.17}$$

Donde:

$$\epsilon_{p,k}^{(1)}(\lambda) = (1 - \hat{h}_{p,k}^{(1)2})\Sigma_{i=1}^{n0}\epsilon_{i,k}^{0}(\lambda)w_{ip} \tag{A.18}$$

y $\epsilon_{i,k}^{0}(\lambda)$ es definida por (A.10) y (A.12).

Si la salida de la red es analógica o binaria, respectivamente. La generalización directa de los resultados anteriores proporciona la siguiente ecuación de actualización de los pesos sinápticos $v_{pq}^{(r)}$, $r = 1, 2, \cdots L-1$

$$v_{p,k}^{(r)} = v_{p,k-1}^{(r)} + \alpha\epsilon_{p,k}^{(r)}(\lambda)\hat{h}_k^{(r+1)} \tag{A.19}$$

Donde $\epsilon_{p,k}^{(r)}(\lambda)$ es dada por (A17) y

$$\epsilon_{p,k}^{(r)}(\lambda) = (1 - \hat{h}_{p,k}^{(r)2})\Sigma_{l=1}^{n_{r-1}}\epsilon_{l,k}^{(r-1)}(\lambda)v_{lp}^{(r-1)} \qquad \forall\ r = 2, 3, \cdots, L \tag{A.20}$$

En particular, los pesos sinápticos $v_{pq}^{(L)}$ son actualizados a través de la ecuación

$$v_{p,k}^{(L)} = v_{p,k-1}^{(L)} + \alpha\epsilon_{p,k}^{(L)}(\lambda)x_k \tag{A.21}$$

Apéndice B

B.1 Teorema de Stone - Weierstrass

Teorema B.1 *Sea S un conjunto compacto N dimensional y sea* $\Omega \supset C[S]$ *sea un conjunto de funciones reales valuadas en S que satisfacen las siguientes condiciones:*

(a) Función identidad: la función constante $f(x) = 1$ en Ω.

(b) Separabilidad: Para dos puntos cualquiera $x_1 \neq x_2$ en S, existe una $f \in \Omega$ tal que $f(x_1) = f(x_2)$.

(c) Cerradura algebraica: Para cualquier f, $g \in \Omega$ y $\alpha, \beta \in R$, las funciones fg y $\alpha f + \beta g$ pertenecen a Ω

Entonces Ω es denso en $C[S]$.

El Teorema de Stone-Weierstrass tiene aplicación potencial a la aproximación de funciones.

B.2 Implementación del algoritmo rápido de retropropagación en Matlab.

Enseguida se presenta el listado del programa de implementación del algoritmo en Matlab con los comentarios convenientes para el caso.

```
%*************************************************************************
%RUTINA DE ALGORITMO DE APRENDIZAJE PARA REDES NEURONALES
%MULTICAPA POR RETROPROPAGACION DE ERROR %
%*************************************************************************

% DEFINA NUMERO DE ENTRADAS Ni.
% DEFINA NUMERO DE SALIDAS Ns.
% DEFINA VELOCIDAD DE CONVERGENCIA Mu.
% DEFINA NUMERO DE UNIDADES OCULTAS Nh.
% DEFINA NUMERO DE ASOCIACIONES ENTRADA-SALIDA m.
% INTRODUZCA VECTORES DE ENTRADAS Y SALIDAS
% DE ASOCIACION X e Y.
% DEFINICION DE MATRICES DE PESOS INICIALES
```

```
L =ones(Ns,1);                  % Valor de inicio para término lambda.
W = 0.1*rand(Ns,Nh) - 0.25;     % Matriz de parámetros capa de salida.
V = 0.1*rand(Nh,Ni) - 0.55;     % Matriz de parámetros capa oculta
Ie = ones(Nh,1);                % Matriz identidad para calculos
Ies = ones(Ns,1);
Eob =0.08*ones(Ns,1);           % Función objetivo de error
Er = ones(Ns,1);

%     INICIA ALGORITMO RETROPROPAGACION     %

while Er>Eob

Er =0*ones(Ns,1);               % Inicializa error a cero.

for k =1:m
x = Xe(:,k);                    % Asigna columna de elementos de entrada.
y = Cad(:,k);                   % Asigna elementos de columna de salida.
he = tanh(V*x);                 % Calcula vector de salidas neuronas ocultas.
ye = W*he;                      % Calcula vector de salidas.
Dy = y - ye;                    % Calcula error de salida.
Eo = diag(Dy)*L + diag(tanh(Be*Dy))*(Ies -L); % Calcula error de la capa de salida
wi = W + al*Eo*he';             % Actualiza pesos del primer renglón.
W = wi;                         % Actualiza matriz de pesos capa de salida.
WTEo = W'*Eo;                   % Calcula término de error.
Te1 = Ie - diag(he)*he;         % Término uno para error de capa oculta.
Eh = diag(Te1)*WTEo;            % Calcula vector de error de la capa oculta.
Ve = V + al*Eh*x';              % Calcula matriz V estimada.
V = Ve;                         %
he = tanh(V*x);                 % Calcula vector de salidas neurona oculta.
ye = W*he;                      % Calcula vector de salidas.
Dy = y - ye;                    % Calcula error de salida.
Er = Er + 0.5*diag(Dy)*Dy;      % Calcula vector de error.
end
for i=1:Ns
ei=Er(i);
L(i,1) = exp(-Mu/(ei*ei))
end
end % Fin del algorítmo.

%     PROCEDIMIENTO AUXILIAR DE VERIFICACIÓN     %

for i=1:m
as = Xe(:,i);
hes = tanh(V*as);
```

```
yes = W*hes;
Yef(:,i) = yes;
end
ti = 0:1:N;
tis = 1:1:m;
plot(ti,Cad(1,:));hold; % Gráficas de resultado de verificación.
plot(tis,Yef(1,:),'r');
figure
plot(ti,Cad(2,:));hold;
plot(tis,Yef(2,:),'r');
figure
plot(ti,Cad(3,:));hold;
plot(tis,Yef(3,:),'r');
```

B.3 Implementación del programa para la utilización de ANFIS de Matlab.

Enseguida se presenta el listado del programa de implementación de ANFIS, con los comentarios convenientes para el caso.

```
%*****************************************************%
% PROGRAMA DE ENTRENAMIENTO Y GENERALIZACIÓN %
% UTILIZANDO CAJA DE HERRAMIENTAS DE ANFIS *****%
% DE MATLAB OCT-2000 ****%
%*****************************************************%
load evec
xi = evec*[1;0;0;0;0];          % Calculo de coordenada x de imagén.
yi = evec*[0;1;0;0;0];          % Calculo de coordenada y de imagén.
Xe = cat(2,xi,yi);               % Formación de vector de entradas.
Xe = Xe;
sx = evec*[0;0;1;0;0];          % Formación de las componentes de salida.
sy = evec*[0;0;0;1;0]:
sz = evec*[0;0;0;0;1];
Coord = cat(2,sx,sy,sz);        % Formación de vector 3D de salida.
Coord = Coord'

% Realiza entrenamiento de red neuro - difusa para salida X %

trnData = [Xe sx];              % Define datos de entrenamiento.
numMFs = 11;                    % Número de funciones.
mfType = 'gbellmf':             % Tipo de función.
epoch_n = 30;                   % Número de epocas.
```

```
in_fismat = genfis1(trnData,numMFs,mfType);
out_fismat = anfis(trnData,in_fismat,30);

% Calcula error de salida en X %

Ers1 = sx - evalfis(Xe,out_fismat);
plot(Ers1);
title('Error en X');
figure

% Realiza entrenamiento de red neuro - difusa para salida Y %

trnData = [Xe sy];
numMFs = 11;
mfType = 'gbellmf';
epoch_n = 30;
in_fismat = genfis1(trnData,numMFs,mfType);
out_fismat = anfis(trnData,in_fismat,30);

% Calcula error de salida en Y %

Ers2 = sy - evalfis(Xe,out_fismat);
plot(Ers2);
title('Error en Y')
figure

% Realiza entrenamiento de red neuro - difusa para salida Z %

trnData = [Xe sz];
numMFs = 6;
mfType = 'gbellmf';
epoch_n = 30;
in_fismat = genfis1(trnData,numMFs,mfType);
out_fismat = anfis(trnData,in_fismat,30);
% Calcula error de salida en Z %
Ers3 = sy - evalfis(Xe,out_fismat);
plot(Ers3);
title('Error en Z ANFIS Datos de simulación')
xlabel('Número de Dato en arreglo de entrada')
ylabel('Magnitud de error (m)')
grid
figure

% Gráficas para análisis comparativo %
```

```
plot(Ers3);
title('Error en Z ANFIS Datos de simulación')
xlabel('Número de Dato en arreglo de entrada')
ylabel('Magnitud de error (m)')
grid
figure
plot(Ers1);
title('Error en X ANFIS Datos de simulación')
xlabel('Número de Dato en arreglo de entrada')
ylabel('Magnitud de error (m)')
grid
figure
plot(Ers2);
title('Error en Y ANFIS Datos de simulación')
xlabel('Número de Dato en arreglo de entrada')
ylabel('Magnitud de error (m)')
grid
figure
plot(sx);
title('Coordenada X(salida) Datos de simulación')
xlabel('Número de Coordenada en arreglo de entrada')
ylabel('Magnitud de Coordenada (m)')
grid
figure
plot(sy);
title('Coordenada Y(salida) Datos de simulación')
xlabel('Número de Coordenada en arreglo de entrada')
ylabel('Magnitud de Coordenada (m)')
grid
figure
plot(sz);
title('Coordenada Z(salida) Datos de simulación')
xlabel('Número de Coordenada en arreglo de entrada')
ylabel('Magnitud de Coordenada (m)')
grid
```

B.4 Implementación del programa para la utilización del Toolbox de redes neuronales de Matlab.

```
%*********************************************************%
   %     PROGRAMA DE ENTRENAMIENTO Y GENERALIZACIÓN  %
   %     UTILIZANDO CAJA DE HERRAMIENTAS     %
   %     DE REDES NEURONALES %
   %     DE MATLAB OCT-2000                                %
   %*********************************************************%
   evec = R;                    % Matriz de datos ya generados.
   xi = evec*[1;0;0;0;0];       % Calculo de coordenada x de imagen.
   yi = evec*[0;1;0;0;0];       % Calculo de coordenada y de imagen.
   Xe = cat(2,xi,yi);           % Formación de vector de entradas.
   Xe = Xe';
   sx = evec*[0;0;1;0;0];       % Formación de las componentes de salida.
   sy = evec*[0;0;0;1;0];
   sz = evec*[0;0;0;0;1];
   Coord = cat(2,sx,sy,sz);     % Formación de vector 3D de salida.
   Coord = Coord'
   Xew = Xe';
   mini = min(Xew);
   maxi = max(Xew);
   pr = cat(1,mini,maxi);
   pr = pr';
   [nXe, minnXe, maxnXe, nCoord, minc, maxc] = premnmx(Xe, Coord);

   % Define la red neuronal %

   nred = newff(pr,[50,3], {'tansig','purelin'},'trainlm', 'learngdm', 'mse');
   t=1:1000;
   yea = sim(nred,Xe); % Simula red neuronal con entradas.

   % Definición de parámetros de diseño de red neuronal %

   nred.trainParam.show = 5;
   nred.trainParam.lr = 0.050;
   nred.trainParam.epochs = 150;
   nred.trainParam.goal = 0.010;
   nred = train(nred, nXe, nCoord);
   nredn = sim(nred, nXe); %
   red = postmnmx(nredn, minc, maxc); % Postprocesamiento de red

   % Gráficas de resultados %
```

```
figure
plot(t,Coord(3,:),'ro')
hold
plot(t,red(3,:),'g*')
figure
plot(t,Coord(2,:),'ro')
hold
plot(t,red(2.:),'g*')
figure
plot(t,Coord(1,:),'ro')
hold
plot(t,red(1.:),'g*')
figure
err = Coord - red;
plot(t,err);
```

B.5 Programa de generación de datos de entrenamiento para simulación.

```
%*********************************************************%
% PROGRAMA DE CALCULO DE COORDENADAS DE IMAGEN*%
% PARA GENERACION DE PARES ENTRADA SALIDA
% PARA ENTRENAMIENTO ****%
%*********************************************************%
Lo = 5/100; % L = Longitud del objeto(cubo) de prueba.
Dy = 1/1000; % Dy = Incrememnto sobre el eje Y.
Dx = 1/1000; % Dx = Incrememnto sobre el eje X.
Dz = 1/1000; % Dz = Incrememnto sobre el eje Z.
deX = 0.3; % dx = Dist.del origen medida sobre X.
deZ = 0.1; % dy = Dist. del origen medida sobre Y.
deY = 0.2; % dz = Dist. del origen medida sobre Z.
Xo = 0; % Coordenada X de la cámara.
Yo = 0;
Zo = 1;
r1 = 0.03; % Distancias respecto del centro del soporte de la cámara.
r2 = 0.02; r3 = 0.02;
THE = (135/90)*pi;
AL = (135/90)*pi;La = 0.035; % Longitud focal efectiva.
B1 = [0,0]; % Valor de inicio vector unidimensional.
% Datos para red neuronal. %
Ni=2; % Número de entradas.
```

```
Nh=50; % Número de neuronas ocultas.
Ns=3; % Número de salidas.
Mu=0.7; % Factor mu.
al=0.07; % Velocidad de aprendizaje.
Be=2.5; % Factor beta.
m=550; % Número de pares entrada salida.
%*********************************************%
% Generación de puntos de cara Izquierda del objeto          %
%*********************************************%
n = Lo/Dy; % Número de incrementos en Y.
n1 = Lo/Dz; % Número de incrementos en Z.
N = n*n1; % Dimensión de los vector de coordenadas.
X = deX*ones(N,1);
X = cat(1,0,X); % Concatenación de vector.
for j=1:n1 %
Z(j) = deZ + j*Dz; % Calculo de la coordenada Z de mundo.
for i=1:n
Y(i) = deY + i*Dy; % Calculo de la coordenada Y de mundo.
B(i,:) = [Y(i),Z(j)];
end
B = cat(1,B1,B); % Forma el par coordenadas cara del objeto.
B1 = B;
B = [0,0]; % Actualización del vector.
end
CaI = cat(2,X,B1);
B1 = [0,0]; % Valor de inicio vector unidimensional.

%*********************************************%
% Generación de puntos de cara Derecha del objeto        %
%*********************************************%

n = Lo/Dx; % Número de incrementos en Y.
n1 = Lo/Dz; % Número de incrementos en Z.
N = n*n1; % Dimensión de los vector de coordenadas.
Y = deY*ones(N,1);
Y = cat(1,0,Y); % Concatenación de vector.
for j=1:n1 %
Z(j) = deZ + j*Dz; % Calculo de la coordenada Z de mundo.
for i=1:n
X(i) = deX + i*Dx; % Calculo de la coordenada Y de mundo.
B(i,:) = [Z(j),X(i)];
end
B = cat(1,B1,B); % Forma el par coordenadas cara del objeto.
```

```
B1 = B;
B = [0,0]; % Actualización del vector.
end
CaD = cat(2,B1,Y);
Cad = CaD';
Cad = [Cad(2,:);Cad(3,:);Cad(1,:)];
B1 = [0,0]; % Valor de inicio vector unidimensional.
%*****************************************%
% Generación de puntos de cara Superior del objeto      %
%*****************************************%
n = Lo/Dy; % Número de incrementos en Y.
n1 = Lo/Dx; % Número de incrementos en Z.
N = n*n1; % Dimensión de los vector de coordenadas.
Z = Lo*ones(N,1);
Z = cat(1,0,Z); % Concatenación de vector.
for j=1:n1 %
X(j) = deX + j*Dx; % Calculo de la coordenada Z de mundo.
for i=1:n
Y(i) = deY + i*Dy; % Calculo de la coordenada Y de mundo.
B(i,:) = [X(j),Y(i)];
end
B = cat(1,B1,B); % Forma coordenadas de la cara del objeto.
B1 = B;
B = [0,0]; % Actualización del vector.
end
CaS = cat(2,B1,Z);

%*****************************************%
% Calculo de coordenadas de imagen Cara Izquierda      %
%*****************************************%

n = Lo/Dy; % Número de incrementos en Y.
n1 = Lo/Dz; % Número de incrementos en Z.
N = n*n1; % Dimensión de los vector de coordenadas.
for i=2:N % Sin considerar cero.
NuX = La*[(CaI(i,1) - Xo)*cos(THE) + (CaI(i,2)-Yo)*sin(THE) - r1];
DuXY = - [(CaI(i,1) - Xo)*sin(THE)*sin(AL) + (CaI(i,2) - Yo)*cos(THE)*sin(AL) -
(CaI(i,3) - Zo)*cos(AL) + r3 + La];
NuY = - La*[(CaI(i,1) - Xo)*sin(THE)*cos(AL) + (CaI(i,2) - Yo)*cos(THE)*cos(AL) +
(CaI(i,3) - Zo)*sin(AL) - r2];
x(i-1) = (NuX/DuXY); % Calculo coordenada de la imagen.
y(i-1) = (NuY/DuXY); % Calculo coordenada de la imagen.
end
```

```
yi = y; % Coordenadas imagen cara izquierda objeto.
xi = x;

%*****************************************%
% Calculo de coordenadas de imagen Cara Superior      %
%*****************************************%
for i=2:N % Sin considerar cero.
NuX = La*[(CaS(i,1) - Xo)*cos(THE) + (CaS(i,2)-Yo)*sin(THE) - r1];
DuXY = - [(CaS(i,1) - Xo)*sin(THE)*sin(AL) + (CaS(i,2) - Yo)*cos(THE)*sin(AL) -
(CaS(i,3) - Zo)*cos(AL) + r3 + La];
NuY = - La*[(CaS(i,1) - Xo)*sin(THE)*cos(AL) +
(CaS(i,2) - Yo)*cos(THE)*cos(AL) +(CaS(i,3) - Zo)*sin(AL) - r2];
x(i-1) = (NuX/DuXY); % Calculo coordenada de la imagen.
y(i-1) = (NuY/DuXY); % Calculo coordenada de la imagen.
end
ys = y; % Coordenadas imagen cara izquierda objeto.
xs = x;

%*****************************************%
% Calculo de coordenadas de imagen Cara Derecha       %
%*****************************************%
for i=2:N % Sin considerar cero.
NuX = La*[(CaD(i,2) - Xo)*cos(THE) + (CaD(i,3)-Yo)*sin(THE) - r1];
DuXY = - [(CaD(i,2) - Xo)*sin(THE)*sin(AL) + (CaD(i,3) - Yo)*cos(THE)*sin(AL) -
(CaD(i,1) - Zo)*cos(AL) + r3 + La];
NuY = - La*[(CaD(i,2) - Xo)*sin(THE)*cos(AL) +
(CaD(i,3) - Yo)*cos(THE)*cos(AL) +(CaD(i,1) - Zo)*sin(AL) - r2];
x(i-1) = (NuX/DuXY); % Calculo coordenada de la imagen.
y(i-1) = (NuY/DuXY); % Calculo coordenada de la imagen.
end

%*****************************************%
% Calculo de vector de coordenadas de imagen
% de entrada para red  %
%*****************************************%
yd = y; % Coordenadas imagen cara izquierda objeto.
xd = x;
Xe = cat(1,xd,yd); % Forma matriz de entrada.
plot(xd,yd);hold;
plot(xi,yi);
plot(xs,ys);
```

Bibliografía

[1] K.S. Narendra "Neural Networks for real time Control" Proceedings of the 36th Conference on Desicion & Contrl, San Diego, California USA. December 1997

[2] Nicolaos B. Karayianis, IEEE and Anastasios N.V. "Fast Learning Algorithms for Neural Networks" IEEE Transactions and Systems-II: Analog and Digital Signal Processing, Vol, 39, No 7, July 1992.

[3] A.R. Barron Universal Aproximation Bounds for Superpositions of a Sigmoidal Function, IEEE Transactions on Information Theory.

[4] B. Widrow and M. A. Lehr, " 30 Years of Adaptive Neural Networks: Perceptron, madaline and backpropagation", Proc. IEEE , vol. 78, pp. 1415 - 1442, Sept. 1990.

[5] P. Werbos, " Beyond Regression: New tools for prediction and analysis in behavioral sciences" PhD. dissertation, Harvard Univ. Cambridge, MA, Aug. 1974.

[6] Jyh-Shing Roger Jang, "ANFIS: Adaptive-Network-Based Fuzzy Inference System.", IEEE transactions on system, man and cybernetics, vol. 23 no 3, May/Jun 1993.

[7] K. S. Fu, R.C. Gonzales, C. S. G. Lee, Robótica Control, Visión e inteligencia, Mc Graw-Hill/ Interamericana, Sep 1994.

[8] "Darpa Neural Network Study", AFCEA International Press, 1988.

[9] Rosenblat, F., "Principles of Neurodynamics: Perceptrons and Theory of Brain Mechanisms", Washington, D.C.:Spartan Books, 1961

[10] Widrow B., S.D. Sterns, "Adaptive Signal Processing", New York: Prentice-Hall, 1985.

[11] Armstrong, W.M. "Recursive Solution to the Equations of Motion of an N-link Manipulator", Proc. 5th World Congr., Theory of Machines, Mechanisms, vol. 2, pages 1343-1346.

[12] M. W. Spong, M. Vidyasagar. "Robot Dynamics and Control", John Wiley & Sons, Inc. !989.

[13] Xiao-Wei TU and Bernard DUBUISSON. "CCD Camera Model and its Physical Characteristics Consideration in Calibration Task for Robotics". IEEE International Workshop on Robot and Human Communication. 1992.

[14] Castelman, K.R., "Digital Image Understanding", Prentice-Hall, New Jersey.

[15] Roger Y. Tsai, "A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses", IEEE J. Robotics and Automation, vol. RA-3, no. 4, pp. 323-344, 19787.

[16] Kumpati S.and Narendra, "Adaptive Control Using Neural Networks" Neural Networks for Control, Massachussetts Institute of Technology, 1990.

[17] Carlos C. de Wit, B. Siciliano, G. Baskin, " Theory of Control Robot", Berlin: Springer-Verlag, 1997.

[18] Minsky, M. L. and Papert, S.A. "perceptrons: An introduction to computational geometry". Cambridge, MA: MIT.

[19] Yu Tang and Gerardo Guerrero, "Descentralized Robust Control of Robot Manipulators", Proccedings of the American Control Conference, Philadelphia, Pennsylvania. June 1998.

[20] Madan M. Gupta, "Inteligent Control Systems, Theory and Applications", IEEE press, 1996.

[21] K.S. Narendra and K. Parthasarathy, "Identification and Control of Dynamical Systems using Neural Networks", IEEE Transactions on Neural Networks, March 1990.

[22] P. Barron, "Universal Approximation Bounds for superpositions of Sigmoidal Functions" IEEE Transactions on Informatios Theory ,1993.

[23] Thomas M.M., Helge J. Ritter and Klau J.S. "Three Dimensional Neural Networks for Learning Visuomotor Coordination of a Robot Arm", IEEE Transactions on Neural Network, vol.1, No. 1, March 1990.

[24] D. Kuhn, J.L. Buessler, J.P. Urban, "Neural Approach to Visual Servoing for Robotic Hand Eye Coordination", IEEE 1995.

[25] Bernard Espiau, Francoise Chaumete and Patrick River, "A New Approach to Visual Servoing in Robotics", IEEE Transactions on Robotics and Automation, vol 8, No. 3, June 1992.

[26] V.Rao Vemuri, "Artificial Neural Networks Concepts and Control Applicatios", IEEE Computer society press 1992.

[27] A.M.S. Salzala and A.S. Morris, "Neural Networks for Robotic Control, Theory and Applications", Ellis Horwood 1996.

[28] R. Kunchev and I. Topalova, "Inavriant Recognition System for 2D Objets Based on Tree Stage BPNN Structure", IEEE Electronics Letters, vol.34, No. 10, March 1998.

[29] Rosenblat F., "Principles of Neurodynamics", Washington D.C., Spartan Press 1961.

[30] Li-Xin Wang., "Adaptive Fuzzy Systems and Control", PTR Prentice Hall, 1994.

[31] Hornik, K., M. Stinchcombe, and H. White, "Multilayer Feedforward Networks are Universal Approximators" Neural networks, no. 2 1989, 359-366.