APLICACIÓN DE UNA RED NEURONAL ARTIFICIAL PARA LA IDENTIFICACIÓN Y SEGUIMIENTO DE TRAYECTORIA DE UN ROBOT.

ANGEL MIGUEL RUIZ ZABALETA

UNIVERSIDAD TECNOLOGICA DE BOLIVAR

FACULTAD DE INGENIERIA ELECTRICA Y ELECTRONICA

CARTAGENA DE INDIAS D.T. Y C.

2004

APLICACIÓN DE UNA RED NEURONAL ARTIFICIAL PARA LA IDENTIFICACIÓN Y SEGUIMIENTO DE TRAYECTORIA DE UN ROBOT.

ANGEL MIGUEL RUIZ ZABALETA

Monografía realizada como

requisito del

minor en automatización industrial

Director

ING. ELECTRONICO OSCAR ACEVEDO

UNIVERSIDAD TECNOLOGICA DE BOLIVAR

FACULTAD DE INGENIERIA ELECTRICA Y ELECTRONICA

CARTAGENA DE INDIAS D.T. Y C.

2004

Nota de aceptación

_____

_____

_____

_____

_____

_____

Firma del presidente del jurado

_____

Firma del jurado

Cartagena de indias D.T. y C. 30 de noviembre de 2004

CONTENIDO

LISTA DE FIGURAS

Pág.

LISTA DE TABLAS

GLOSARIO

APRENDIZAJE SUPERVISADO: Se asemeja al método de enseñanza tradicional con un profesor que indica y corrige los errores del alumno hasta que éste aprende la lección. Si la red utiliza un tipo de aprendizaje supervisado debemos proporcionarle parejas de patrones entrada-salida y la red neuronal aprende a asociarlos.

APRENDIZAJE NO SUPERVISADO: Si el entrenamiento es no supervisado, únicamente debemos suministrar a la red los datos de entrada para que extraiga los rasgos característicos esenciales. En terminología estadística equivale a los modelos en los que sólo hay vectores de variables independientes y buscan el agrupamiento de los patrones de entrada: análisis de conglomerados o cluster, escalas multidimensionales, etc.

APRENDIZAJE COMPETITIVO: Una red básica de aprendizaje competitivo tiene una capa de neuronas de entrada y una capa de neuronas de salida. Un patrón de entrada x es un simple punto en el espacio real o binario de vectores *n*-dimensional. Los valores binarios (0 ó 1) de representación local son más usados en los nodos de salida. Esto es, hay tantas neuronas de salida como número de clases y cada nodo de salida representa una categoría de patrones.

VISIÓN ARTIFICIAL: La Visión Artificial (VA) es la construcción de descripciones de objetos físicos, llenas de significado y explícitas, a partir de imágenes. Incluyen el procesamiento de imágenes (no pensamos que sea acertado en la actualidad).

La entrada a un sistema de visión por medio de una máquina, es una imagen o varias imágenes mientras que la salida es una descripción que debe cumplir dos criterios: estar relacionada con la imagen observada y contener toda la información que se necesita para realizar una tarea determinada.

SMART VISION CHIPS: Se entiende por *Smart vision chips* aquellos circuitos que integran los elementos fotodetectores y algún tipo de procesamiento en el mismo chip. La intención es mejorar globalmente la sensorización. Tradicionalmente los *Smart vision sensors* han sido sensores que han incorporado parte o la totalidad del procesamiento en el mismo circuito del sensor.

SEÑAL PWM: La modulación de ancho de pulso es una técnica utilizada para controlar dispositivos, o para proveer un voltaje variable de corriente continua. Algunas aplicaciones en las que se utiliza MAP son controles de motores, de iluminación y de temperatura.

La señal generada tendrá frecuencia fija y tiempos de encendido y apagado variables. En otras palabras, el período de la señal se mantendrá constante, pero la cantidad de tiempo que se mantiene en alto y bajo dentro de un período puede variar.

GRADIENTE CONJUGADO: Es un gran método para matrices simétricas y definidas positivas. Ya que aprovecha muy bien la estructura de la matriz y tiene muy buenas propiedades de estabilidad numérica.

El *método del gradiente conjugado* es un método iterativo que a partir de un iterante inicial va calculando sucesivos iterantes que se van acercando a la solución exacta del sistema lineal.

GENERALIZACIÓN: Hasta ahora hemos visto que una RNA a capas es capaz de "aprender" a representar relaciones arbitrarias entre patrones de entrada y patrones de salida. El proceso de aprendizaje consiste en sintetizar la relación entrada-salida, partiendo de un conjunto de datos de entrenamiento. Si este fuese todo el proceso, la RNA no sería más que un dispositivo que permite almacenar y evocar información conocida, similar a las memorias.

No obstante, existe una intención más profunda que la anterior, y es el emplear la RNA, una vez entrenada, como un dispositivo útil en el procesamiento de información que no ha participado en el proceso de entrenamiento.

La información requerida para lograr un modo adecuado de generalización no debe estar oculta por otros rasgos o propiedades de los datos de entrenamiento. En este sentido, en muchos casos es necesario un proceso previo de extracción de rasgos. La información necesaria para una buena capacidad de generalización debe aparecer más o menos explícita en los datos de entrada. Para ello, en muchos casos es de relevancia la selección de un buen esquema de

representación. En muchos casos es de relevancia la selección de un buen esquema de representación.

INTRODUCCIÓN

Actualmente, es cada vez mayor el número de robots que se van incorporando a la industria, al sector servicios, etc. Incluso, ya es común encontrar, aunque aún están en sus primeras generaciones, robots comerciales de entretenimiento, robots domésticos para la realización de tareas del hogar, etc.

Sin embargo, la aplicación de un robot para realizar una determinada tarea depende, en un elevado porcentaje, del conocimiento a *priori* del espacio de trabajo y de la localización de los objetos a manipular. Esta limitación es debida a que los robots industriales comerciales no integran sistemas sensoriales, que les permitan adaptarse a su entorno.

Dentro de los sistemas sensoriales adaptables a los robots industriales comerciales, los sensores visuales se han convertido en elementos cada vez mas frecuentes en las aplicaciones robóticas recientes. Su principal ventaja es que permiten obtener una descripción del entorno bastante completa de forma no intrusiva. Se podría decir que, en general, facilitan la integración de dispositivos robóticos en entornos menos estructurados que los que se encuentran en aplicaciones clásicas de automatización.

La interacción entre sensores visuales y sistemas robóticos ha sido y es objeto de gran interés en la investigación de las últimas décadas. Tiene especial relevancia en aplicaciones en las que un *manipulador robótico* o robot (en el sentido mas genérico de *sistema eléctrico-mecánico articulado susceptible*

de ser accionado) tiene que interactuar con objetos, bien se trate de objetos estáticos cuya localización no sea conocida de forma precisa o simplemente sea impredecible, o bien de objetos móviles, cuyo perfil de movimiento no sea conocido.

En este trabajo se plantea el problema de identificación y seguimiento de trayectoria un robot basado en redes neuronales artificiales. Se plantearan las bases teóricas para entender el desarrollo de un proyecto con redes neuronales y se creará un plan de trabajo para la creación de un sistema autónomo "inteligente", el tipo de sistema autónomo a diseñar esta enfocado a los robots de exhibición para laberintos con intersecciones o bien para sillas de ruedas autónomas; el robot cuenta con dos motores de corriente directa con reductores de velocidad, además cuenta con un sistema de visión artificial muy sencillo conformado por matrices de leds infrarrojos. Las soluciones propuestas están enfocadas al área de investigación de la Universidad Tecnológica de Bolívar como aporte científico y guía practica a los interesados en estudiar aplicaciones de la inteligencia artificial.

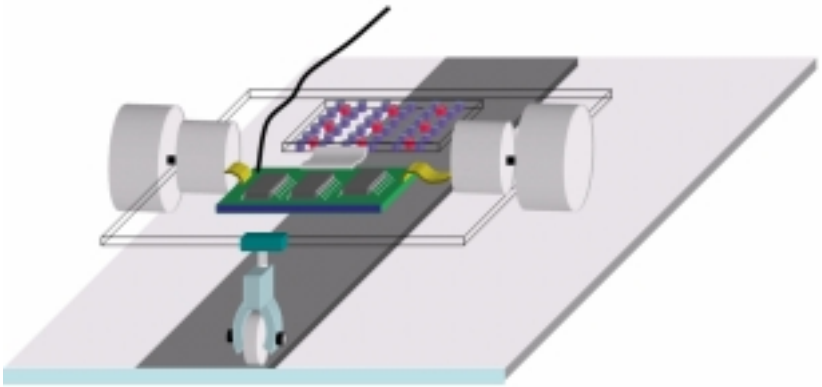


Figura 1. Prototipo robot guiado por trayectos de líneas en superficies.

1. REDES NEURONALES ARTIFICIALES

En general las redes neuronales se pueden clasificar de diversas maneras, según su topología, forma de aprendizaje (supervisado o no supervisado), tipos de funciones de activación, valores de entrada (binarios o continuos); un resumen de esta clasificación se observa en la figura 1

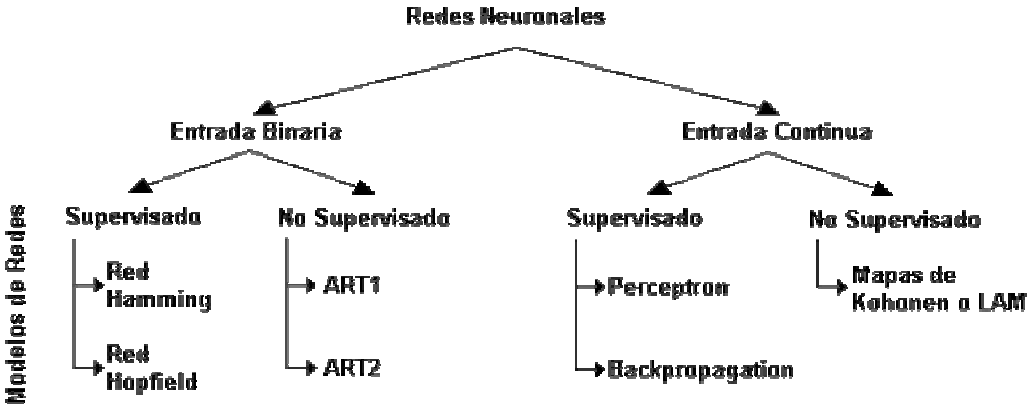


Figura 2. Clasificación de las redes neuronales.

Para la escogencia de una red neuronal debemos empezar por definir si es supervisada o no supervisada, en nuestra aplicación tendremos los valores de entrada de los sensores de visión y las acciones a realizar por las redes que trabajaran para la identificación de trayectoria y el control de la dirección para la navegación; por esto podemos afirmar que tenemos redes supervisadas en nuestra aplicación.

Entre las redes supervisadas más comunes tenemos la Red Hamming, Red Hopfield, Perceptron y la Red Backpropagation.

1.1 RED DE HAMMING

La red de Hamming es uno de los ejemplos más simples de aprendizaje competitivo, a pesar de ello su estructura es un poco compleja ya que emplea el concepto de capas recurrentes en su segunda capa y aunque hoy en día en redes de aprendizaje competitivo se ha simplificado este concepto con el uso de funciones de activación más sencillas [1].

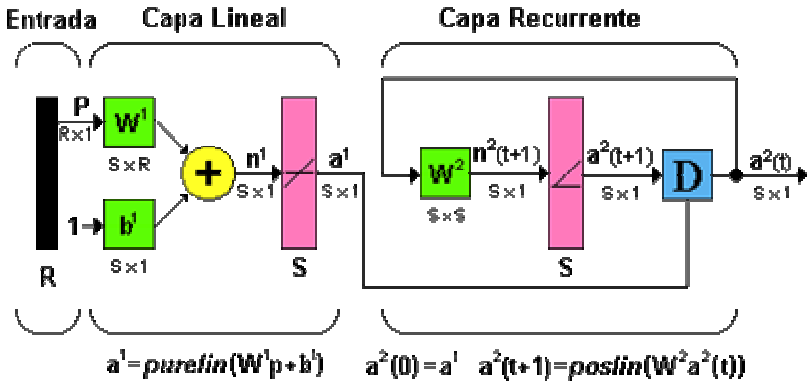


Figura 3 Red de Hamming

Esta red consiste en dos capas; la primera capa, la cual es una red Instar, realiza la correlación entre el vector de entrada y los vectores prototipo, la segunda capa realiza la competición para determinar cual de los vectores prototipo está más cercano al vector de entrada.

[1] Acosta Maria Isabel. Tutorial de redes neuronales. Hipertexto disponible en Internet. ohm.utp.edu.co\neuronales\Capitulo2\Competitivas\Hamming.htm

La red de Hamming representa uno de los primeros avances en este tipo de aprendizaje, convirtiéndola en un modelo obligado de referencia dentro de las redes de aprendizaje competitivo. Las neuronas en la capa de salida de esta red compiten unas con otras para determinar la ganadora, la cual índica el patrón prototipo más representativo en la entrada de la red, la competición es implementada por inhibición lateral (un conjunto de conexiones negativas entre las neuronas en la capa de salida).

1.2 RED HOPFIELD

La red de Hopfield esta compuesta de neuronas dinámicas altamente interconectadas gobernadas por ecuaciones diferenciales no lineales, esta red funciona como una memoria asociativa no lineal que puede procesar patrones presentados de forma incompleta o con ruido, siendo útil como una poderosa herramienta de optimización. La red de Hopfield no tiene una ley de aprendizaje asociada, esto significa que la red no es entrenada ni realiza un proceso de aprendizaje, sin embargo es posible determinar la matriz de pesos por medio de un procedimiento basado en la función de alta ganancia de Lyapunov.

Una red de Hopfield puede diseñarse como una memoria asociativa, en este caso es llamada memoria de contenido direccionable, porque la memoria recupera la información almacenada con base en parte de su contenido, en contraste con las memorias estándar de computo, donde la información se recupera con base en sus direcciones, por ejemplo si se tiene una base de datos de contenido direccionable que contiene nombres y direcciones de los empleados de una

empresa, la información completa se recupera por ejemplo suministrando el nombre (o parte de él) [2].

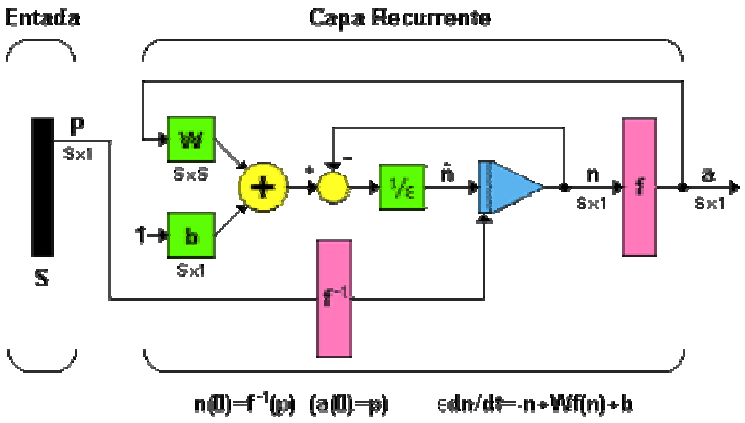


Figura 4. Notación compacta red de Hopfield

Como se observa, la red de Hopfield esta compuesta de neuronas dinámicas altamente interconectadas gobernadas por ecuaciones diferenciales no lineales, esta red funciona como una memoria asociativa no lineal que puede procesar patrones presentados de forma incompleta o con ruido, siendo útil como una poderosa herramienta de optimización

1.3 PERCEPTRÓN

El primer asociador de patrones es el Perceptrón (Rosemblat,1962), una red lineal no recurrente, compuesta por una capa de foto-receptores, una capa de asociadores aleatoriamente conectados y una capa de salida compuesta por una sola unidad, denominada perceptrón.

[2] Acosta Maria Isabel. Tutorial de redes neuronales. Hipertexto disponible en Internet. ohm.utp.edu.co\neuronales\Capitulo2\Recurrentes\EstructuraH.htm

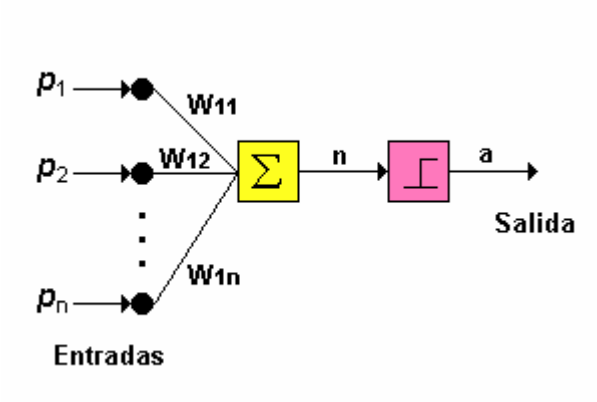


Figura 5. Estructura de la Red Perceptrón.

El Perceptrón es capaz de asociar patrones de entrada formados por variables continuas, con una variable de salida discreta binaria: 0/1, dado que el perceptrón aplica una función de umbral. Si la respuesta de la red fuese la correcta, las conexiones existentes entre los asociadores aleatoriamente conectados y el perceptrón no se modificarían, cosa que sí ocurre en caso contrario. Aunque pueda parecer que este modelo tiene varias capas, técnicamente es una arquitectura de dos capas, ya que sólo hay una capa de conexiones modificables.

El Perceptrón se mostró bastante limitado en cuanto a su capacidad de representar funciones (por ejemplo, la función XOR) linealmente inseparables (Wassermann, 1989). Son necesarias más capas de conexiones modificables para representar funciones complejas y relaciones no lineales.

La única neurona de salida del Perceptrón realiza la suma ponderada de las entradas, resta el umbral y pasa el resultado a una función de transferencia de tipo escalón. La red tipo Perceptrón emplea principalmente dos funciones de transferencia, *hardlim* con salidas 1, 0 o *hardlims* con salidas 1, -1; su uso depende del valor de salida que se espera para la red, es decir si la salida de la red es unipolar o bipolar; sin embargo la función *hardlims* es preferida sobre la *hardlim*, ya que el tener un cero multiplicando algunas de los valores resultantes del producto de las entradas por el vector de pesos, ocasiona que estos no se actualicen y que el aprendizaje sea más lento.

Una técnica utilizada para analizar el comportamiento de redes como el Perceptrón es presentar en un mapa las regiones de decisión creadas en el espacio multidimensional de entradas de la red, en estas regiones se visualiza qué patrones pertenecen a una clase y cuáles a otra, el Perceptrón separa las regiones por un hiperplano cuya ecuación queda determinada por los pesos de las conexiones y el valor umbral de la función de activación de la neurona, en este caso los valores de los pesos pueden fijarse o adaptarse empleando diferentes algoritmos de entrenamiento[3].

1.4 **RED BACKPROPAGATION**

[3] Acosta Maria Isabel. Tutorial de redes neuronales. Hipertexto disponible en Internet ohm.utp.edu.co\neuronales\Capitulo2\Perceptron\AntecedentesP.htm.

Esta red aprovecha la naturaleza paralela de las redes neuronales para reducir el tiempo requerido por un procesador secuencial para determinar la correspondencia entre unos patrones dados.

Además el tiempo de desarrollo de cualquier sistema que se este tratando de analizar se puede reducir como consecuencia de que la red puede aprender el algoritmo correcto sin que alguien tenga que deducir por anticipado el algoritmo en cuestión. La Backpropagation es un tipo de red de aprendizaje supervisado, que emplea un ciclo propagación–adaptación de dos fases. Una vez que se ha aplicado un patrón a la entrada de la red como estímulo, este se propaga desde la primera capa a través de las capas superiores de la red, hasta generar una salida. La señal de salida se compara con la salida deseada y se calcula una señal de error para cada una de las salidas.

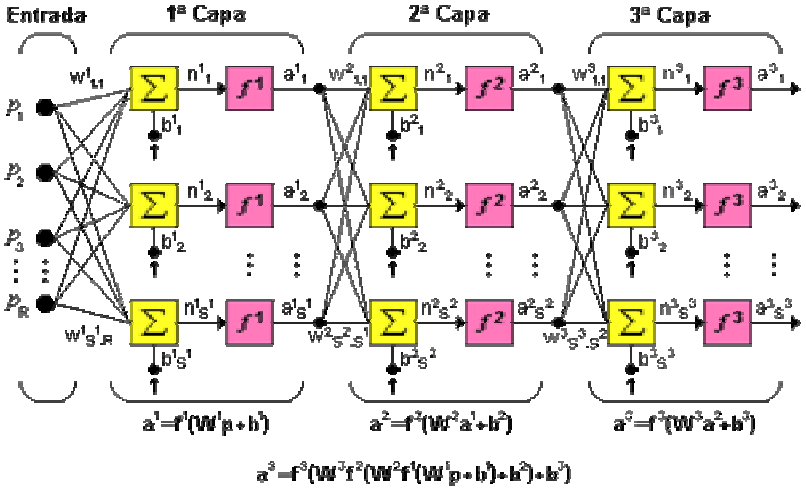


Figura 6. Estructura Red de tres capas.

Las salidas de error se propagan hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de la capa oculta que contribuyen directamente a la salida. Sin embargo las neuronas de la capa oculta solo reciben una fracción de la señal total del error, basándose aproximadamente en la contribución relativa que haya aportado cada neurona a la salida original. Este proceso se repite, capa por capa, hasta que todas las neuronas de la red hayan recibido una señal de error que describa su contribución relativa al error total. Basándose en la señal de error percibida, se actualizan los pesos de conexión de cada neurona, para hacer que la red converja hacia un estado que permita clasificar correctamente todos los patrones de entrenamiento. La importancia de este proceso consiste en que, a medida que se entrena la red, las neuronas de las capas intermedias se organizan a sí mismas de tal modo que las distintas neuronas aprenden a reconocer distintas características del espacio total de entrada. Después del entrenamiento, cuando se les presente un patrón arbitrario de entrada que contenga ruido o que esté incompleto, las neuronas de la capa oculta de la red responderán con una salida activa si la nueva entrada contiene un patrón que se asemeje a aquella característica que las neuronas individuales hayan aprendido a reconocer durante su entrenamiento. Y a la inversa, las unidades de las capas ocultas tienen una tendencia a inhibir su salida si el patrón de entrada no contiene la característica para reconocer, para la cual han sido entrenadas[4].

[4] Acosta Maria Isabel. Tutorial de redes neuronales. Hipertexto disponible en Internet ohm.utp.edu.co\neuronales\Capitulo2\Perceptron\AntecedentesP.htm.

2. VISIÓN ARTIFICIAL

A pesar de la utilización de un sistema de visión artificial sencillo en este proyecto, se requiere involucrar los conceptos relativos a este tema con el propósito de realizar un marco conceptual para futuras investigaciones. En el presente las aplicaciones referentes a la visión artificial se han consolidado y se presumen grandes innovaciones para el uso en robots autónomos, sistemas de navegación y aplicaciones militares.

Figura 7. Silla de ruedas autónoma empleada para experimentos, Universidad de Zaragoza (España)[5].

La Visión Artificial (VA) es la construcción de descripciones de objetos físicos, llenas de significado y explícitas, a partir de imágenes. Incluyen el procesamiento de imágenes (no pensamos que sea acertado en la actualidad).

La entrada a un sistema de visión por medio de una máquina, es una imagen o varias imágenes mientras que la salida es una descripción que debe cumplir dos

[5] R. Martínez, C. Sagüés, J. J. Guerrero. Corrección visual de un robot móvil con homografías. Disponible en Internet, ubicación : webdiis.unizar.es/~rmcantin/ papers/Martinez-Cantin03JA.pdf.

criterios: estar relacionada con la imagen observada y contener toda la información que se necesita para realizar una tarea determinada.[6]

El procesamiento de imágenes es una parte de esta tarea, esta se define como el procesado de una imagen por un computador, con el objeto de producir otra imagen; mientras que la VA trata de la adquisición, el procesado, la clasificación, el reconocimiento, y en su conjunto la toma de decisiones posterior al reconocimiento. La Visión Artificial avanzada describe la deducción automática de las estructuras y propiedades de un mundo tridimensional, posiblemente dinámico, a partir de una o varias imágenes bidimensionales de él. Entre algunas de las áreas de Investigación de la visión artificial encontramos:

- Detección de rasgos en imágenes.
- Representación de contornos.
- Segmentación basada en rasgos.
- Análisis de imágenes de distancias.
- Modelación y representación de la forma.
- Reconstrucción de la forma a partir de una imagen.
- Visión estéreo.
- Análisis del movimiento.

Dentro de las áreas de aplicación encontramos:

- Inspección industrial y control de calidad.
- Vigilancia y seguridad.

[6] Jose Antonio Boluda Grau. Vision log-polar y procesamiento de imágenes tapec.uv.es/~jboluda/curso_doctorado/ curso_doctorado_vision_chips.pdf

- Reconocimiento de caras.
- Reconocimiento de gestos.
- Monitorización de carreteras.
- Vehículos autónomos.
- Sistemas robóticos mano-ojo.
- Espacio y aplicaciones.

Las nuevas tecnologías están llevando a la visión artificial a un nivel de integración muy avanzado, el uso de cámaras de video en este campo ya no será necesario con la implementación de los Sensores visores conocidos como ***Smart vision chips***, los cuales apuntan a ser el pilar de la VA y de seguro serán de uso imprescindible uso en la captación de imágenes.

2.1 SMART VISION CHIPS

Se entiende por *Smart vision chips* aquellos circuitos que integran los elementos fotodetectores y algún tipo de procesamiento en el mismo chip. La intención es mejorar globalmente la sensorización. Tradicionalmente los *Smart vision sensors* han sido sensores que han incorporado parte o la totalidad del procesamiento en el mismo circuito del sensor.

La definición moderna es más extensa. Los *Smart sensors* son circuitos en los que coexisten los sensores y el procesamiento, estando fuertemente ligados todos los niveles de procesamiento. Son sensores específicos de información, no

transductores + elementos de proceso. Todo en un *Smart vision sensor* esta orientado a la aplicación especifica.

Entre las ventajas de los *vision chips* encontramos:

- **Velocidad:** Se evitan los cuellos de botella que muchas veces aparecen entre la etapa de captura y las diversas etapas de procesamiento. Utilización masiva de paralelismo.
- **Tamaño:** Es posible un sólo circuito integrado para implementar un algoritmo de visión complejo.
- **Consumo de potencia:** Se reduce la circuitería innecesaria y la pérdida de energía en la transmisión de información entre etapas.
- **Integración y modularidad:** Se permite liberar a la CPU de la ejecución de tareas muy costosas temporalmente. Orientados a su integración en sistemas modulares jerárquicos.

Desventajas de los *vision chips:*

- Falta de flexibilidad: Ninguno de los *vision chips* son de propósito general (por propia definición). Su eficiencia para resolver una tarea concreta de visión artificial se ve contrarrestada por su falta de flexibilidad.
- Resolución: En la mayoría de de *vision chips* cada píxel incluye un circuito que ocupa una gran proporción del área del píxel. De esta manera, muchos de estos circuitos presentan una baja resolución y/o una baja relación área sensible/área del circuito.

- Dificultad de diseño: El diseño de la zona fotosensible mas la zona de procesamiento se realiza en un área limitada. Todos estos diseños suelen ser *full-custom*, con el riesgo y el coste que conlleva. Además muchas veces se debe llegar a un compromiso prestaciones/área ocupada que puede degradar las prestaciones del circuito.

Entre las diferentes tecnologías utilizadas en los *Smart vision chips* encontramos tres tipos: las tecnologías CCD, CMOS y la BICMOS; a continuación mencionaremos características, ventajas y desventajas de cada una de estas.

2.1.1. Tecnologías ccd. El proceso CCD (Charge Coupled Device) fue originalmente desarrollado para procesado analógico de señales y sensores visuales. Entre las ventajas de esta tecnología encontramos:

- Alta calidad de la imagen.
- Alta resolución.
- Utilidad en gran cantidad de aplicaciones.

Las desventajas son:

- Para adquirir una imagen es necesaria una gran cantidad de fases de reloj y generación de señales de control.
- El acceso es necesariamente secuencial a toda la imagen.
- Complejidad si se desea introducir procesamiento en el mismo circuito (ruido, área por pixel, potencia consumida...)

2.1.2. Tecnologías cmos. Las Ventajas de esta son:

- Tecnología muy madura y avanzada debido a su utilización preferente en el diseño de memorias y CPUs.
- Gran cantidad de recursos (librerías digitales y analógicas).
- Es el proceso más económico del mercado.

  y sus desventajas
- Dificultad en la caracterización de los circuitos analógicos, entre ellos los fotodiodos.
- Efectos no deseados en el escalado de los circuitos analógicos.
- Dificultad en el emparejamiento (*Mismatch* de los circuitos CMOS).

2.1.3. Tecnologías Bicmos, Gaas, Mesfet Y Hemt. El proceso BiCMOS incorpora transistores bipolares en la lógica CMOS, lo cual aumenta la velocidad de CMOS. Se reduce el *Mismatch* de los circuitos CMOS. Por contra son procesos muy costosos y poco integrables (los transistores bipolares ocupan un área mayor que los transistores MOS).

El proceso GaAs, combinado con transistores MESFET (Metal Semiconductor FET) y HEMT (High Electron Mobility Transistors), se caracteriza por la alta velocidad de sus dispositivos analógicos y digitales. Sin embargo es un proceso poco maduro y caro, lo cual dificulta su introducción.

2.2. CLASIFICACIÓN DE LOS VISION CHIPS

los Vision chips se clasifican de acuerdo a la naturaleza de el procesado de los datos adquiridos en :

2.2.1 chips de visión con procesado espacial. Estos circuitos incorporan algún tipo de procesado espacial, desde sencillas operaciones locales de suavizado, hasta operaciones mas complejas de extracción y detección de características. Este tipo de sensores se pueden agrupar en:

- **Retinas de silicio**. Incorporan características de filtrado y de adaptación a la intensidad de luz similares a las retinas de los vertebrados. Realizan operaciones de realce sencillas. Ejemplo: Retina de silicio de Mahowald y Mead.

Los “conos” (elementos sensibles a la luz en lo ojos de los vertebrados) han sido implementados con fototransistores y diodos MOS convertidores logarítmicos de corriente a tensión. En este sistema de visión se realiza un suavizado de la imagen y una adaptación logarítmica a la intensidad de luz.

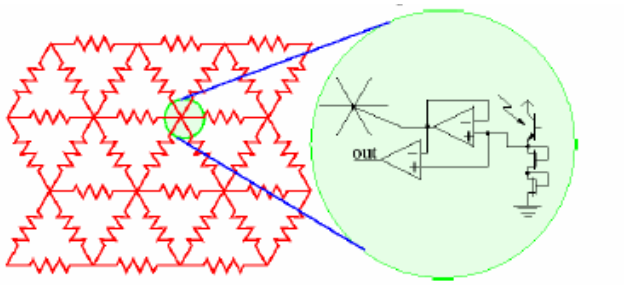


Figura 8. Retina de silicio de Mahowald y Mead.

- **Chips de extracción de características más globales**, como son el cálculo del centroide o la orientación de objetos. Ejemplo: Sensor PASIC de la Linküoping University.

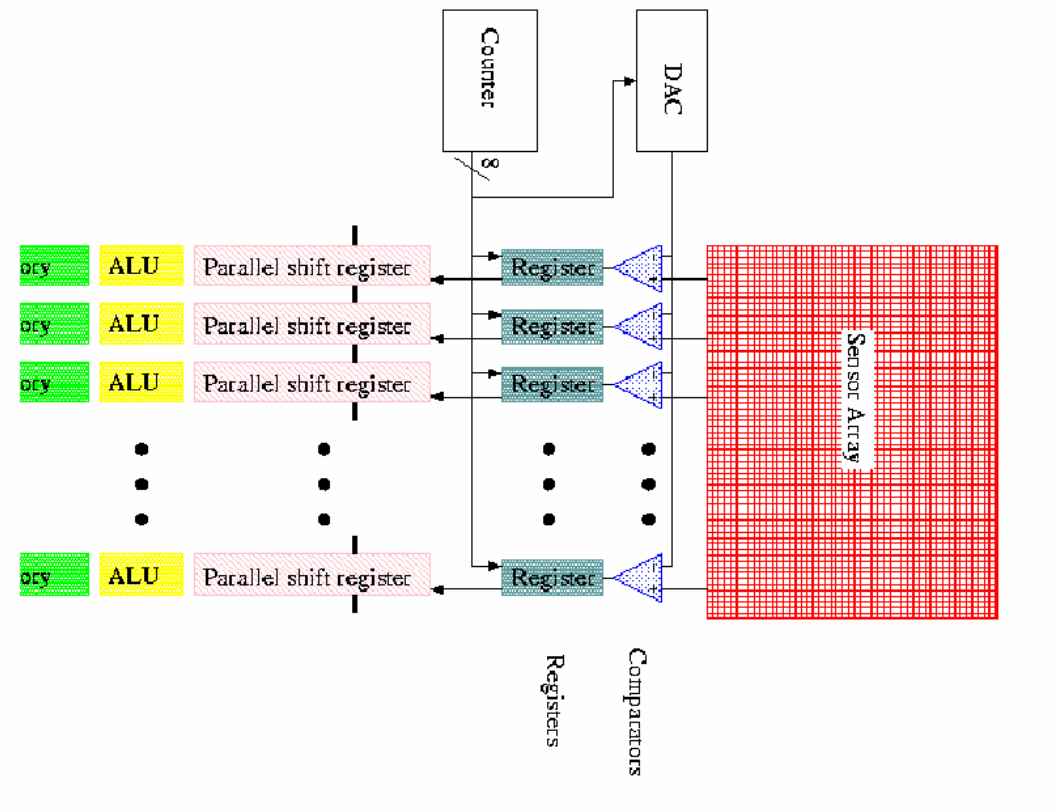
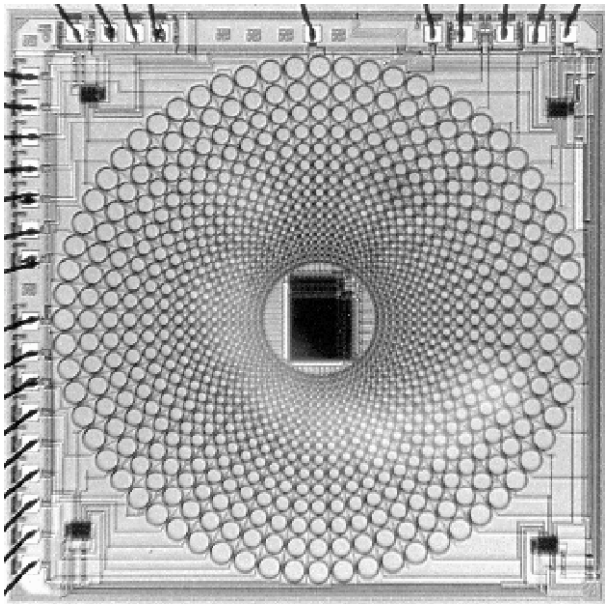


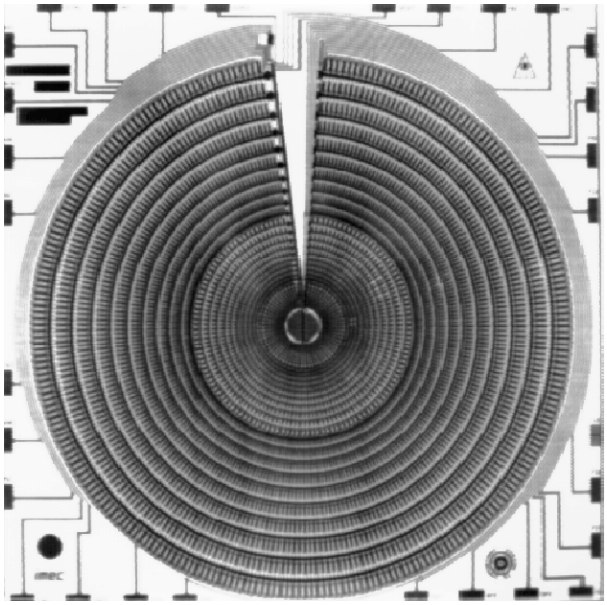
Figura 9. Retina de silicio de Mahowald y Mead.

Dentro de sus características tenemos una resolución de 128x128 fotodetectores, cada fila tiene su propio ADC y su elemento de proceso y se realizan operaciones como dilataciones, convoluciones, etc.

- **Sensores foveales.** En este caso se adopta una distribución espacio-variante de los fotosensores, obteniéndose más resolución en el centro de la imagen que en la periferia.

Figura 10. Sensores CMOS de Wodnicki y CCD de IBIDEM.





Sensor de Wodnicki: Retina con 64 radios x 16 anillos = 1024 pixels.

Sensor de IBIDEM: Retina con 64 radios x 30 anillos =1920 pixels.

Ambos presentan discontinuidad en la fóvea.

- **Chips de visión con procesado espacio- temporal**.

Ejemplo: Sensor de Tanner y Meads. Se computa la velocidad global de toda la imagen a partir de la ecuación de flujo óptico: $\frac{\partial I}{\partial x} v_x + \frac{\partial I}{\partial y} v_y + \frac{\partial I}{\partial t} = 0$

El chip contiene fototransistores y circuitería para realizar diferenciación espacial y temporal.

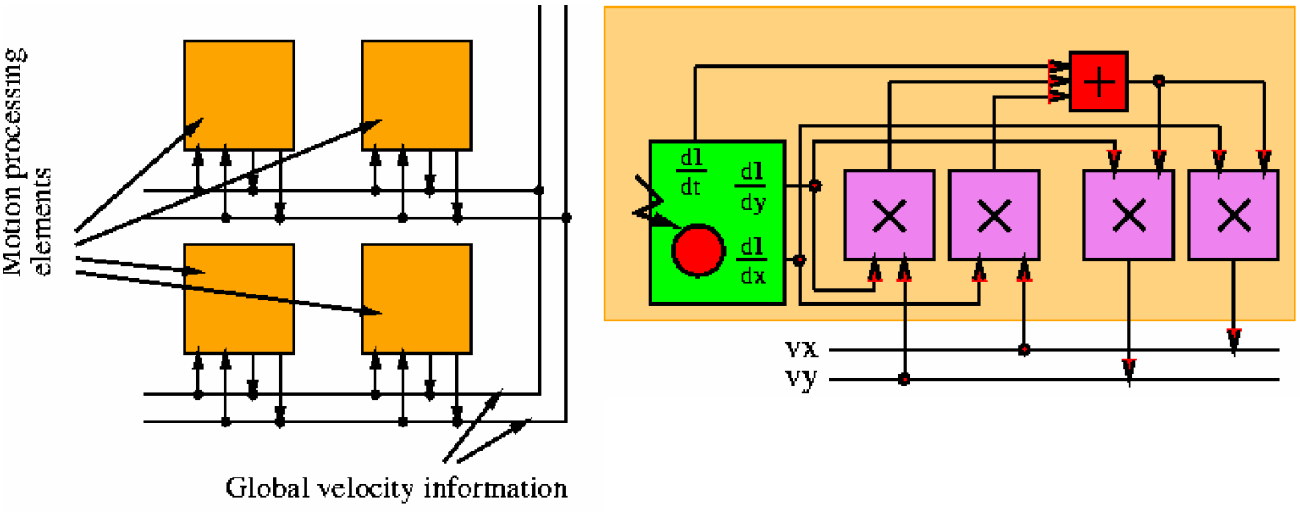


Figura 11 , Sensor de Tanner y Meads

3. DESCRIPCIÓN DEL SISTEMA A CONTROLAR

3.1. SISTEMA DE VISIÓN ARTIFICIAL

El dispositivo usado como sensor esta compuesto por una matriz (6 X 6) de leds emisores infrarrojos, que en subgrupos de 4 leds tienen 1 sensor receptor de luz infrarroja (utilizado comúnmente en la recepción de señales de control remoto de electrodomésticos), en total tenemos 9 receptores y 36 emisores, con este método se logra una buena resolución con pocos receptores.

Su funcionamiento se basa en la multiplexación y comparación de estados lógicos entre receptores (sensores) y emisores, mas específicamente se busca que por medio de la reflexión de la luz sobre una superficie se detecte cual led emisor hizo rebotar hacia el receptor la luz (sabiendo que la luz rebota sobre una superficie clara o blanca y no lo hace sobre una negra).

Para entender el funcionamiento se tomara un subgrupo de 4 leds, cada uno recibe la señal A, B, C y D proveniente de un Multiplexor, el microcontrolador se encarga de mandar al multiplexor cual led estará activado, esto lo realizara con un registro llamado S, este tiene 2 bits (para este caso) S1 y S2.

S		
S1	S2	MUX
0	0	A=1
0	1	B=1
1	0	C=1
1	1	D=1

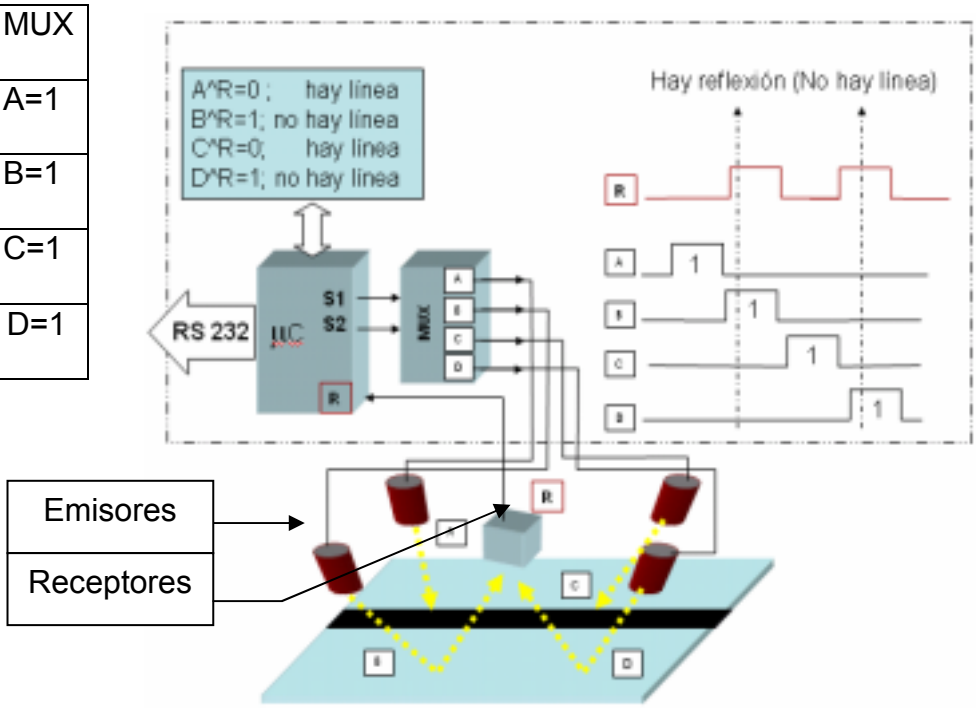


Figura 12. Grupo cuatro emisores y un receptor, acción sobre una línea recta.

Al colocar S en 00 (S1=0 y S2=0) la señal A esta activada, en este caso la luz emitida por este led es absorbida por la línea oscura y la señal del sensor R esta en cero (R=0); para S=01 la luz del led es reflejada por la superficie, entonces R estará activado (R=1). Cuando S=10 entonces R=0 y para S=11 tenemos que R=1. Al final obtendremos la siguiente matriz, esta nos formara la imagen de la superficie

$$\begin{bmatrix} A \,\&\, R & B \,\&\, R \\ C \,\&\, R & D \,\&\, R \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}$$

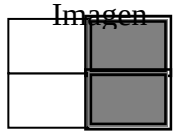


Figura 13. Matriz de la imagen y su representación grafica.

Esta matriz será creada en el software encargado de simular la red neuronal, allí se hará el procesamiento y la toma de desiciones.

Al ampliar el orden de la matriz incrementando el número de leds emisores a 36 y los receptores a 9 obtendríamos una imagen más exacta de la superficie.

La variable “X” distribuye la señal a cada subgrupo de 4 sensores, mientras que la “Y” hace mover la señal entre cada subgrupo; “R” es una palabra de 9 bits que almacena la señal de cada receptor

La matriz completa del sistema será:

(X1,Y1)& R1	(X1,Y2)& R1	(X1,Y3)& R2	(X1,Y4)& R2	(X1,Y5)& R3	(X1,Y6)& R3
(X2,Y1)& R1	(X2,Y2)& R1	……	……	……	……
(X3,Y1)& R4	……	……	……	……	……
(X4,Y1)& R4	……	……	……	……	……
(X5,Y1)& R7	……	……	……	(X5,Y5)& R9	(X6,Y5)& R9
(X6,Y1)& R7	……	……	……	(X6,Y6)& R9	(X6,Y6)& R9

Figura 14. Matriz 6x6 con subgrupos de Rn receptores.

Como ejemplo, pondremos al sistema de visión artificial a un camino en forma de “Y”.

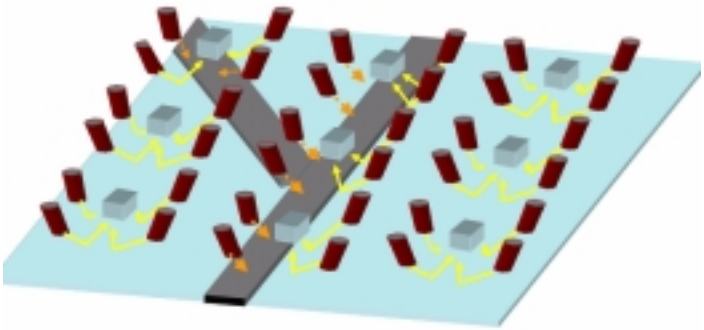


Figura 15. Emisión y recepción de luz infrarroja sobre una trayectoria en forma de “Y”

Las flechas naranjadas indican las emisiones de luz absorbidas por la superficie oscura. La matriz para este caso queda:

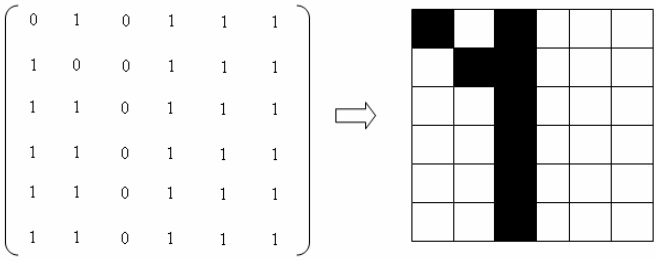


Figura 16. Representación matricial e imagen pixeladas del trayecto en forma de "Y".

La aproximación al plano real es muy aceptable, pero la funcionalidad de la identificación esta en el éxito del entrenamiento de la red.

El esquema para la parte electrónica lo muestra la figura 12 y el microcontrolador encargado de hacer el barrido en la matriz de leds.

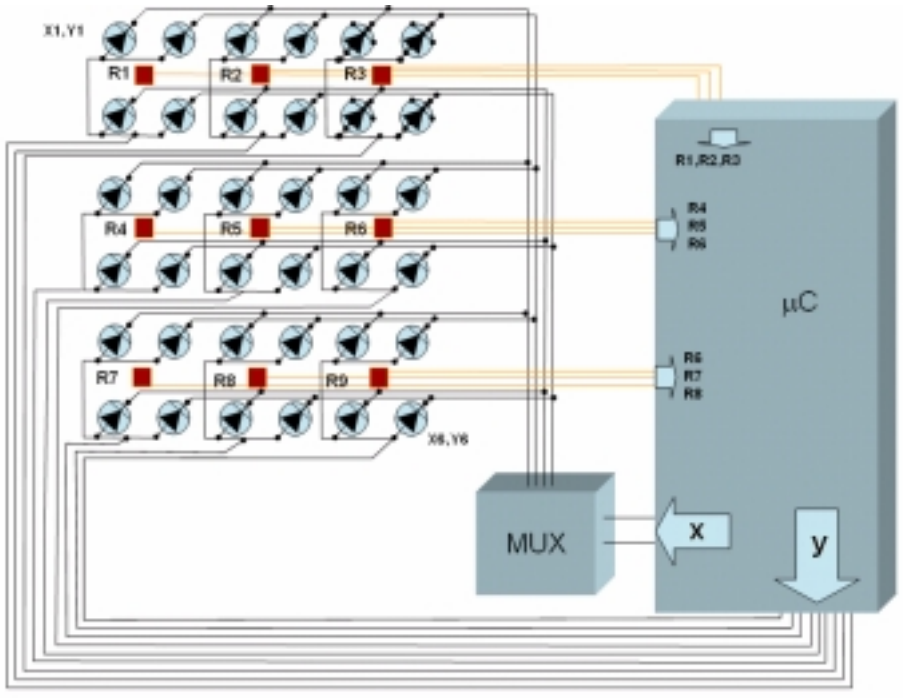


Figura 17. Esquema electrónico para el sistema de visión del robot.

4. IDENTIFICACIÓN DE PARAMETROS DE ENTRADA Y SALIDA

El sistema contara con dos redes neuronales, la primera RNA solo se encargara de la identificación de la trayectoria o forma del camino y la segunda se estará a cargo de la navegación del robot. Ambas RNA recibirán como entrada los 36 bits provenientes del sistema de visión artificial, mas concretamente se tomaran los datos armados por el microcontrolador; antes de detallar sobre cada red es importante saber que son, a donde y que se hará con la información procesada por cada red. Las acciones o desiciones tomadas por la red están interconectadas a un modulo o driver para 2 motores DC, las entradas a controlar de este sistema son la velocidad, y el ángulo de giro del robot.

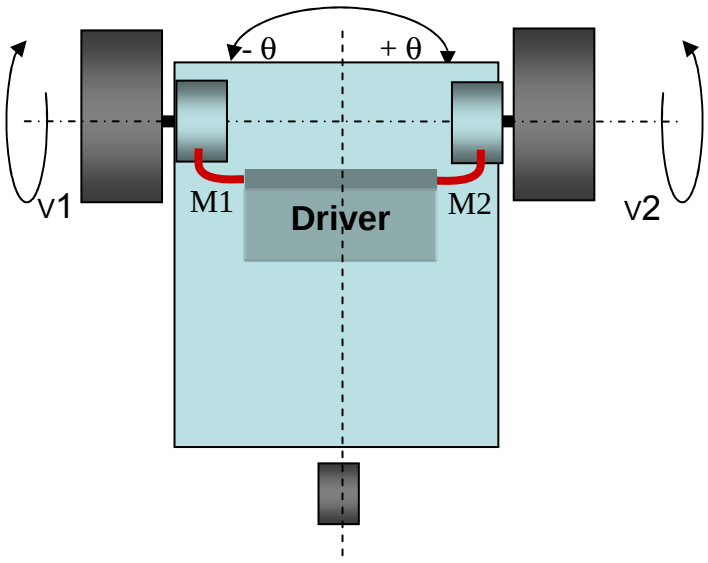


Figura 18. Ángulos de giro para el robot.

La relación entre las velocidades, y el ángulo de giro θ esta expresada por la ecuación que rige el driver controlador, esta determinada por:

$$V1 = \frac{100}{180}(\theta + 90) \quad ; \qquad V2 = \frac{100}{180}(\theta - 90) \qquad ;$$

Donde θ es el ángulo de giro que toma el robot, este puede ser positivo (gira a la derecha) o negativo (gira a la izquierda).

La siguiente tabla muestra en detalle que porcentaje de velocidad debe existir entre los dos motores.

Se debe aclarar que el ángulo de giro que toma el robot en la realidad debe ser ajustado a nivel práctico poniendo la velocidad de los motores necesaria para lograr que coincida con lo afirmado en la tabla 1.

θ^{o}	(Dato) θ	V1	V2
-90	255	0%	100%
-80	241	6%	94%
-70	227	11%	88%
-60	213	17%	83%
-50	199	22%	77%
-40	185	28%	72%
-30	170	33%	66%
-20	156	39%	61%
-10	142	44%	55%
0	128	50%	50%
10	114	55%	44%
20	99	61%	39%
30	85	66%	33%
40	71	72%	28%
50	57	77%	22%
60	43	83%	17%
70	28	88%	11%
80	14	94%	6%
90	0	100%	0%

Tabla 1. Porcentaje de velocidad entre los dos motores para lograr un determinado ángulo de giro.

El driver usado queda con el siguiente esquema.

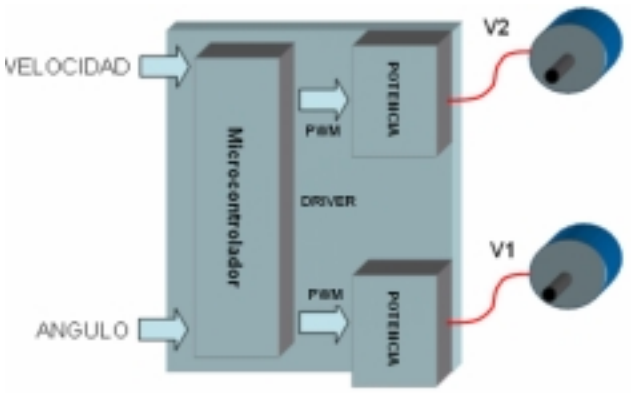


Figura19. Entradas para el driver controlador de los motores.

Como vemos solo es necesario introducir el ángulo de giro y la velocidad, las redes neuronales controlaran estos datos.

Es sistema para control de velocidad y ángulo de giro consta de un microcontrolador, este se encarga de realizar las operaciones necesarias para entregar a los motores DC las señales PWM para su control.

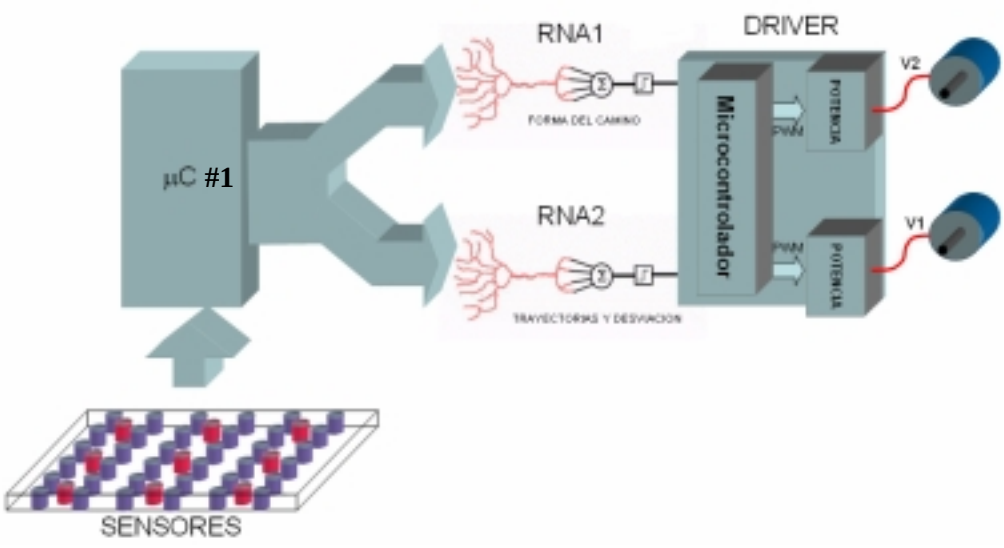
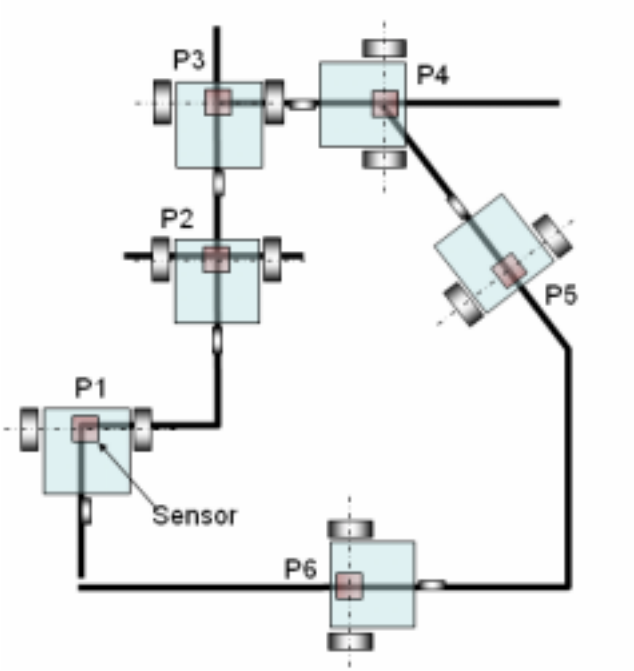


Figura 20. Esquema total para sistema de navegación con RNA.

La RNA1 se encargara de reconocer la forma del camino pero es el microcontrolador encargado del control de los motores quien tiene entre sus datos la velocidad que debe tener los motores de acuerdo a la forma reconocida, por ejemplo: cuando se reconozca una línea recta se le dará al driver esa información en forma directa, pero es el microcontrolador que en memoria tiene la instrucción que para la línea recta los motores deben tener máxima velocidad, entonces este entregara la señal PWM

correspondiente. Según la trayectoria el sistema manejara diferentes velocidades, esto se aclara en la figura 16.

En este ejemplo tenemos el robot en seis posiciones (P1, P2, P3, P4, P5 y P6), decidimos que de acuerdo a la complejidad de la forma se tuviera una velocidad mayor, para el caso de P1 se tendrá una velocidad intermedia, para P6 la velocidad será máxima, mientras que para P2 se tiene una velocidad muy lenta.



	FORMA	VELOCIDAD
P1	└	INTERMEDIA
P2	┼	LENTA
P3	├	LENTA
P4	**Y**	LENTA
P5	**I**	MAXIMA
P6	**I**	MAXIMA

Figura 22. velocidades para el robot

Figura 21. Trayectos para el robot.

Manejar la velocidad de acuerdo a la forma del trayecto nos permite optimizar el tiempo en los recorridos.

La segunda red neuronal (RNA2) estará encargada de la navegación del robot. El sistema para de seguimiento de la trayectoria se basa en la corrección de la trayectoria identificada por la RNA2, pero es el microcontrolador el encargado de hacer la operación de corrección de la trayectoria tratando de hacer que entre el ángulo formado entre la línea identificada y una línea de referencia (línea central del robot) sea cero.

Por ejemplo para una trayectoria de línea recta que se encuentra a un ángulo $\alpha = +45^{o}$ respecto a la línea de referencia, se tomarían las siguientes acciones.

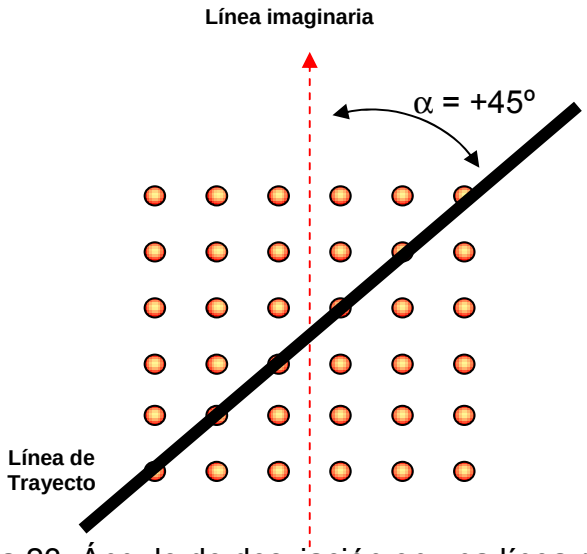


Figura 23. Ángulo de desviación en una línea recta

La RNA2 encargada de la navegación del robot identificara la desviación α, originando un ángulo θ como salida que tendrá la interpretación de α de la red.

Esta RNA2 debe estar apoyada por un *sistema de navegación* que le dicte a esta que rumbo debe tomar, a la vez este sistema debe tener como entrada la salida de

la RNA1, de esta manera se tiene la opción de poder programar las trayectorias del robot ya conocida la ruta por la cual se va ha transitar.

Para en ejemplo citado la RNA1 identifica la forma como una línea recta, por ende el sistema de navegación deberá estar programado con la opción de navegar por el centro; en otro caso, como un trayecto en forma "T" se tienen dos opciones, girar a la izquierda o a la derecha.

Ahora la RNA2 identifica el ángulo de desviación de la línea recta y da la salida $\theta \cong 45º$.

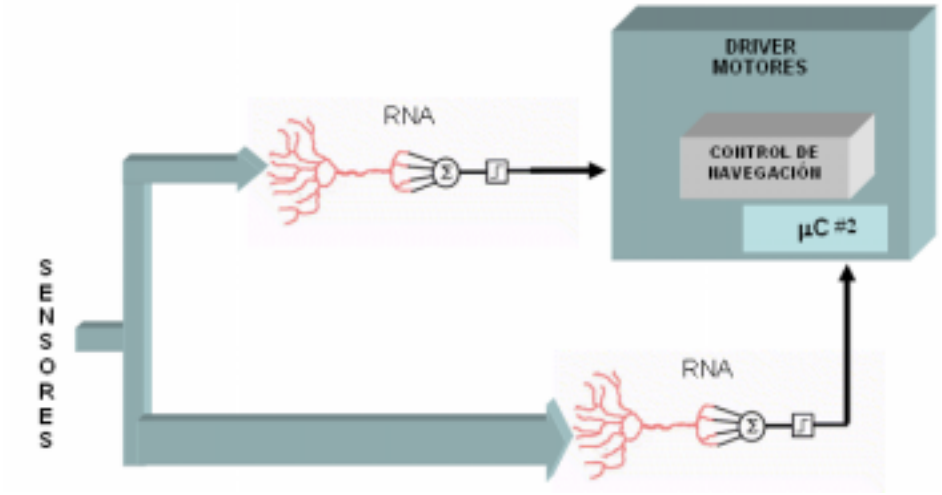


Figura 24. Acción conjunta entre RNA1 y RNA2 para control de la dirección.

El microcontrolador #2 realiza la corrección de la dirección, tratando de que el ángulo α sea igual a cero.

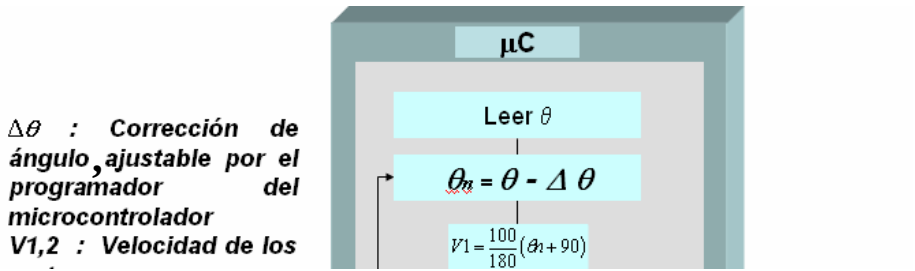


Figura 25. Microcontrolador #2 en la corrección de la dirección.

En resumen las diferentes entradas y salidas para cada RNA quedan:

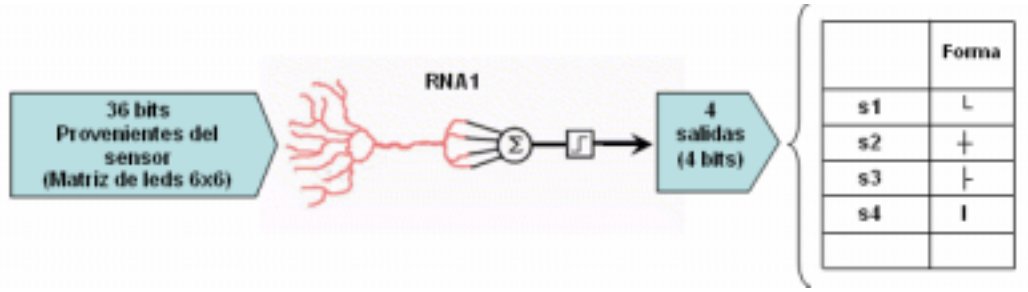


Figura 26. Entradas y salidas para RNA1.

Figura 27. Entradas y salidas para RNA2.

Ya entrenada la primera red neuronal se procede a entrenar la segunda, a continuación una muestran los casos de las entradas y salidas para la RNA2.

En cada caso tenemos la entrada de la matriz que representa la imagen captada del trayecto y los tres bits (IZQ, CEN Y DER) para indicar la dirección a la cual

se quiere guiar el robot. Las matrices mostradas a continuación son validas para el entrenamiento de la RNA1.

5. FIGURAS DE ENTRENAMIENTO

Las figuras empleadas para entrenar la red son formas de trayectoria todas con ángulos rectos. Por la limitada resolución del sensor la red RNA2 leerá ángulos predeterminados (de ±135º, ± 90º, ± 45º y 0º), a continuación presentamos las figuras empleadas para el entrenamiento de la RNA1 Y RNA2.

- **Figuras de entrenamiento para RNA1.**

	Entradas	Entradas	Entradas	Entradas	salidas
Forma: | :	0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0	0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0	0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0	1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1	| : 1 + : 0 ├ : 0 | : 0
Forma: + :	0 0 1 0 0 0 0 0 1 0 0 0 1 1 1 1 1 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0	0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 1 1 1 1 1 0 0 0 1 0 0 0 0 0 1 0 0	0 0 0 1 0 0 0 0 0 1 0 0 1 1 1 1 1 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0	0 0 1 0 0 0 0 0 1 0 0 0 1 1 1 1 1 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0	| : 0 + : 1 ├ : 0 | : 0
Forma: ├ :	0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 1 1 1 0 0 1 0 0 0 0 0 1 0 0 0	0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 1 1 1 0 0 1 0 0 0 0 0 1 0 0 0	0 1 0 0 0 0 0 1 0 0 0 0 0 1 1 1 1 1 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0	0 0 0 1 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0	| : 0 + : 0 ├ : 1 | : 0
Forma: ├ :	0 0 0 1 0 0 0 0 0 1 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0	0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0	0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 1 1 1 0 0 1 0 0 0 0 0 1 0 0 0	0 0 1 0 0 0 0 0 1 0 0 0 0 1 1 1 1 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0	| : 0 + : 0 ├ : 1 | : 0
Forma: | :	0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0	0 0 0 0 0 0 0 0 1 1 1 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0	0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0	| : 0 + : 0 ├ : 0 | : 1

Tabla 2. Forma de trayectos.

Un ejemplo de las trayectorias que puede tomar el robot se muestra a continuación.

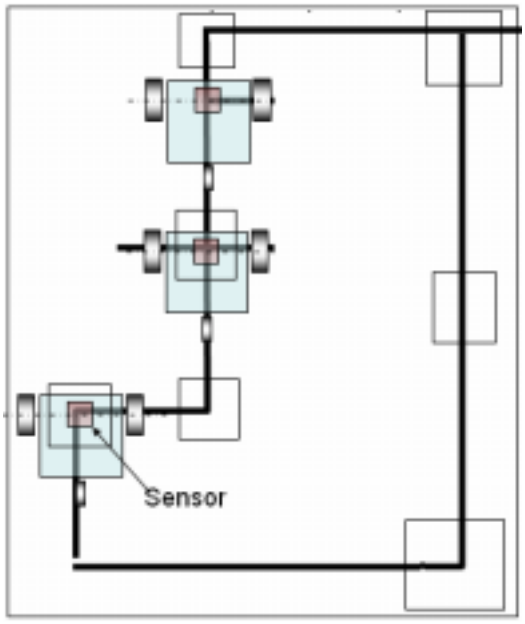


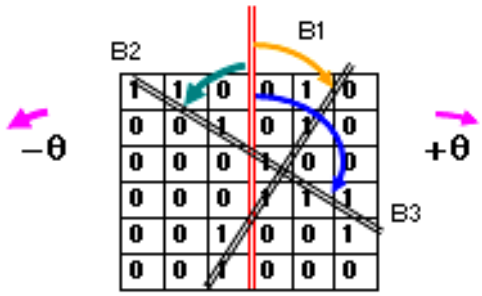
Figura 28. Forma de trayectos y recorrido del robot.

La complejidad de la red a utilizar estará centrada en la resolución del sensor de visión y el formato de la salida a usar, una forma ideal de hacer esta aplicación teniendo una alta resolución del sensor seria obtener una red capaz de navegar en curvas y en intersecciones mas complejas como una “Y”.

- **Figuras de entrenamiento para RNA2.**

Ahora mostraremos las figuras con diferentes situaciones (giros a determinados ángulos) que entrenan la segunda red.

Las entradas A entraran en el conjunto de lo patrones de entrenamiento, y las entradas B serán patrones de prueba de la red. Las salidas A y B están divididas en tres salidas, cada una de esta representa el ángulo de desviación para tres posibles trayectorias que indique el camino.

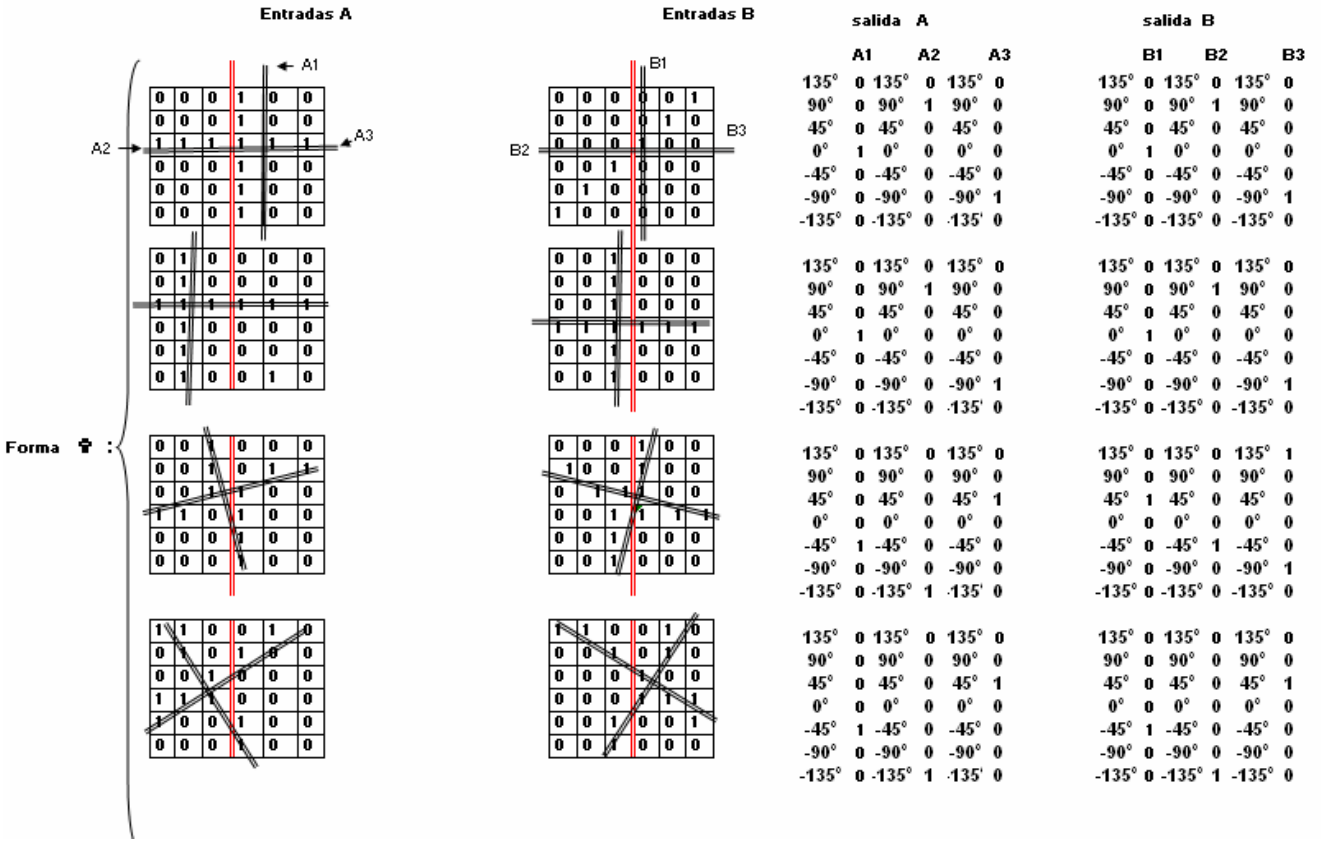


			salida B		
B1		B2		B3	
135°	0	135°	0	135°	1
90°	0	90°	0	90°	0
45°	1	45°	0	45°	0
0°	0	0°	0	0°	0
-45°	0	-45°	1	-45°	0
-90°	0	-90°	0	-90°	0
-135°	0	-135°	0	-135°	0

Figura 29. Ángulos de desviación para una cruz.

A continuación mostraremos los casos de entrenamiento para la RNA2.

Caso 1 Forma ✞.



salida A						salida B					
	A1		A2		A3		B1		B2		B3
135°	0	135°	0	135°	0	135°	0	135°	0	135°	0
90°	0	90°	1	90°	0	90°	0	90°	1	90°	0
45°	0	45°	0	45°	0	45°	0	45°	0	45°	0
0°	1	0°	0	0°	0	0°	1	0°	0	0°	0
-45°	0	-45°	0	-45°	0	-45°	0	-45°	0	-45°	0
-90°	0	-90°	0	-90°	1	-90°	0	-90°	0	-90°	1
-135°	0	-135°	0	-135°	0	-135°	0	-135°	0	-135°	0
135°	0	135°	0	135°	0	135°	0	135°	0	135°	0
90°	0	90°	1	90°	0	90°	0	90°	1	90°	0
45°	0	45°	0	45°	0	45°	0	45°	0	45°	0
0°	1	0°	0	0°	0	0°	1	0°	0	0°	0
-45°	0	-45°	0	-45°	0	-45°	0	-45°	0	-45°	0
-90°	0	-90°	0	-90°	1	-90°	0	-90°	0	-90°	1
-135°	0	-135°	0	-135°	0	-135°	0	-135°	0	-135°	0
135°	0	135°	0	135°	0	135°	0	135°	0	135°	1
90°	0	90°	0	90°	0	90°	0	90°	0	90°	0
45°	0	45°	0	45°	1	45°	1	45°	0	45°	0
0°	0	0°	0	0°	0	0°	0	0°	0	0°	0
-45°	1	-45°	0	-45°	0	-45°	0	-45°	1	-45°	0
-90°	0	-90°	0	-90°	0	-90°	0	-90°	0	-90°	1
-135°	0	-135°	1	-135°	0	-135°	0	-135°	0	-135°	0
135°	0	135°	0	135°	0	135°	0	135°	0	135°	0
90°	0	90°	0	90°	0	90°	0	90°	0	90°	0
45°	0	45°	0	45°	1	45°	0	45°	0	45°	1
0°	0	0°	0	0°	0	0°	0	0°	0	0°	0
-45°	1	-45°	0	-45°	0	-45°	1	-45°	0	-45°	0
-90°	0	-90°	0	-90°	0	-90°	0	-90°	0	-90°	0
-135°	0	-135°	1	-135°	0	-135°	0	-135°	1	-135°	0

Figura 30. Figuras, matrices (entradas A y B) y salidas (salidas An y Bn) para trayecto en forma de cruz

Caso 2 Línea recta.

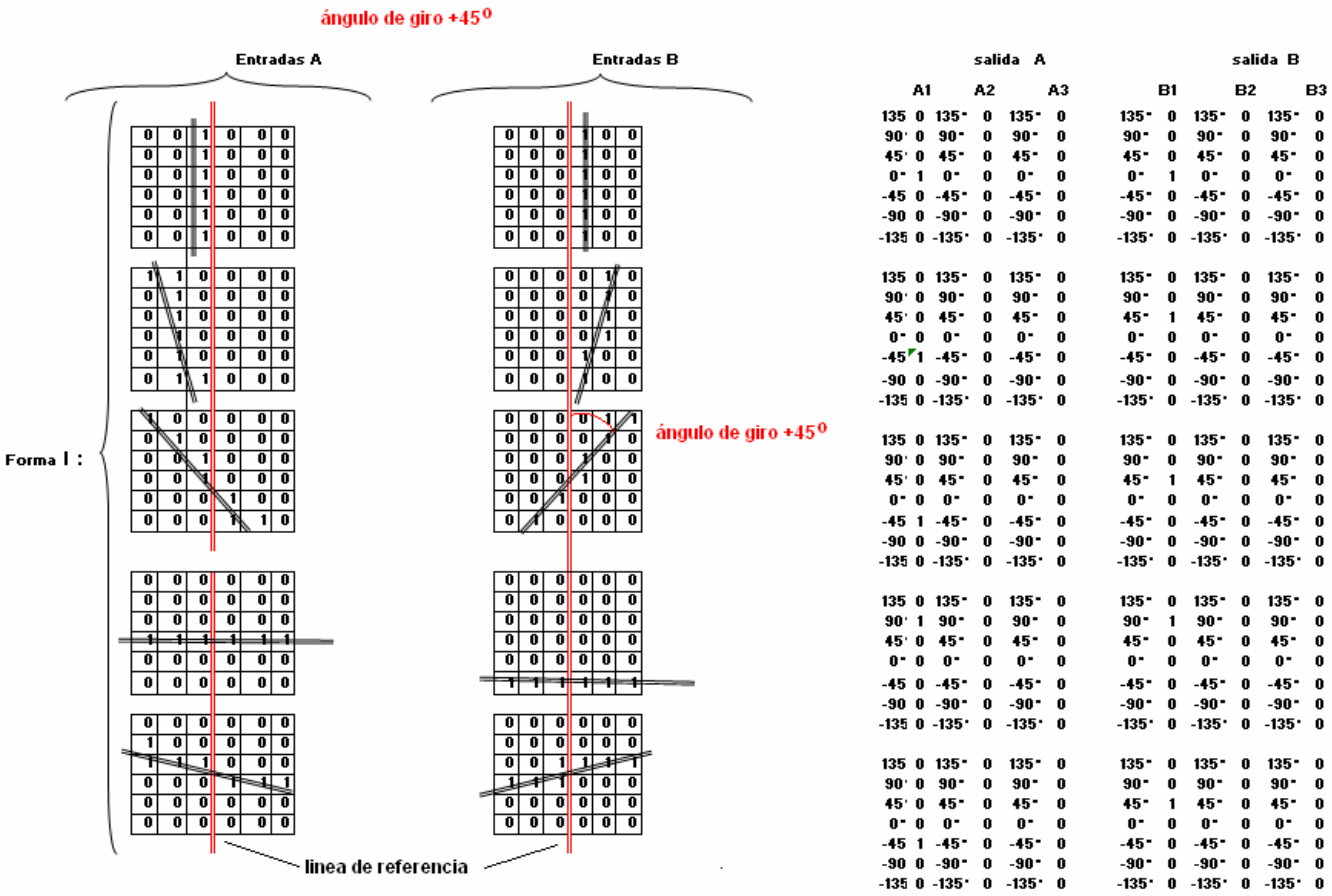


Figura 31. Caso 2, línea recta y ángulo de inclinación respecto a la referencia.

La poca resolución del sensor no permite obtener ángulos como 50º o 85º, por esto todo ángulo que este entre el rango de los 90º y 0º se leerán como 45º.

Caso 3 Forma “L”.

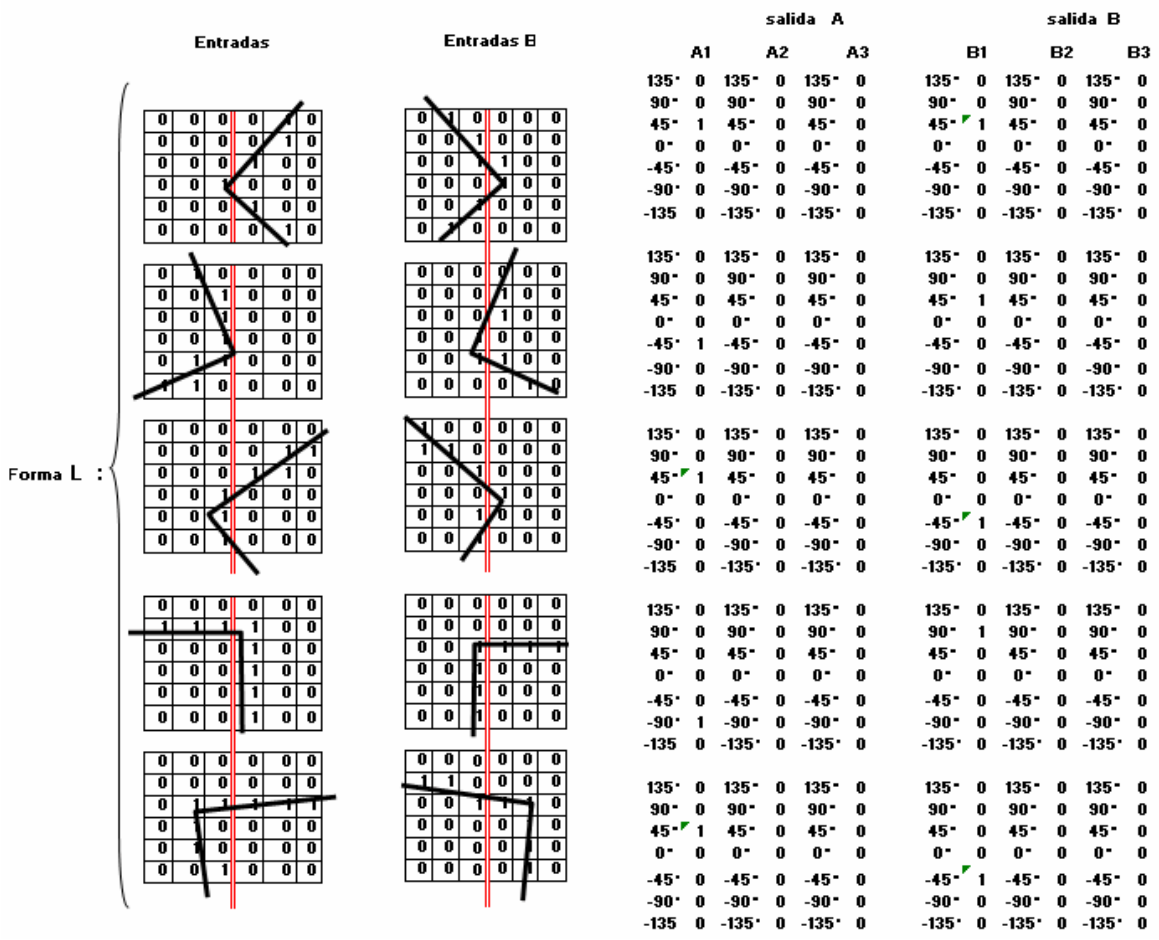


Figura 32. Figuras, matrices (entradas A y B) y salidas (salidas An y Bn) para trayecto en forma de “L”

Caso 4 Forma “T”.

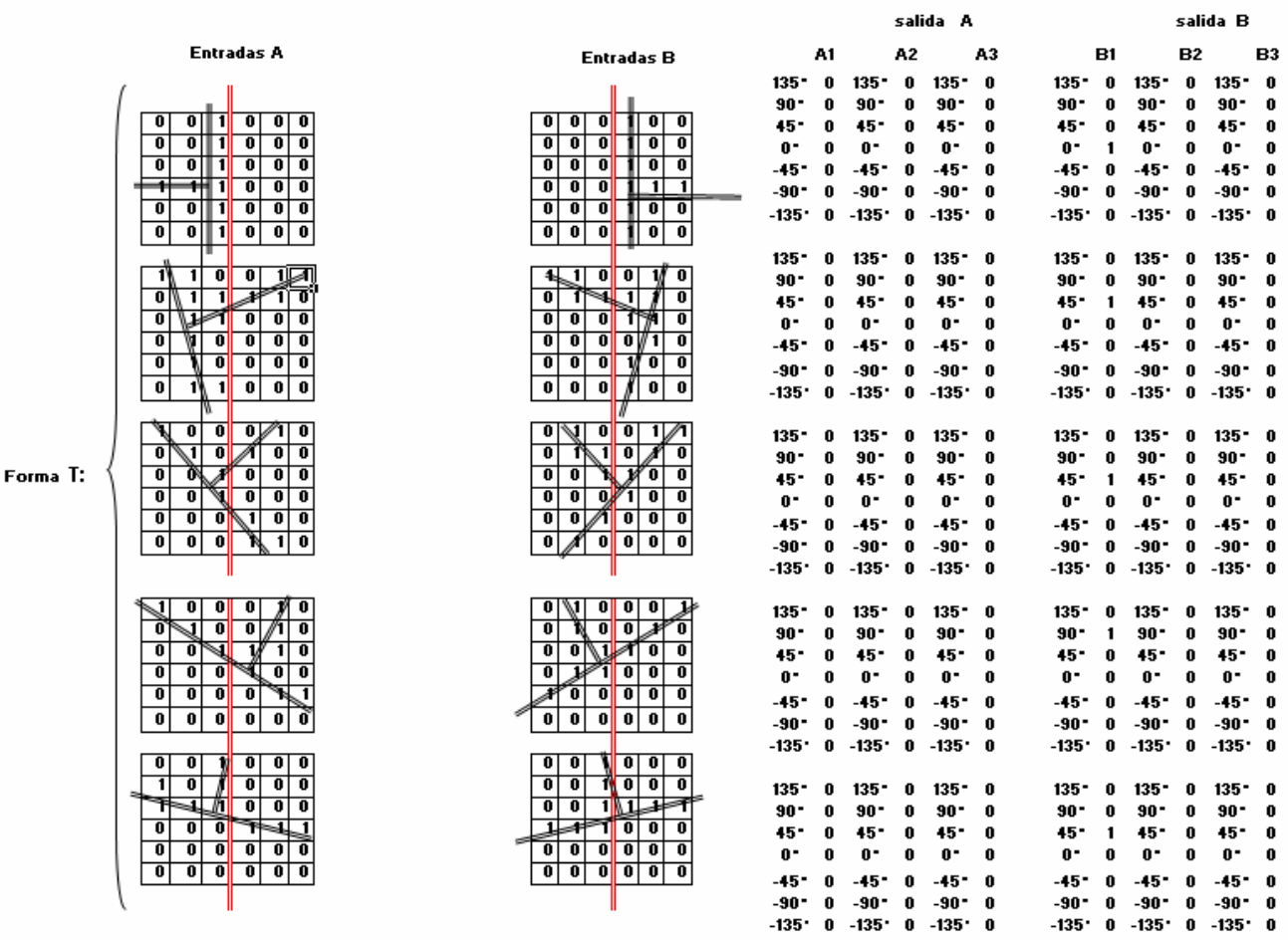


Figura 33. Figuras, matrices (entradas A y B) y salidas (salidas An y Bn) para trayecto en forma “T”

Los patrones descritos para la RNA2 deben usarse como patrones de entrenamiento de la RNA1, con esto se tiene un mejor entrenamiento de la red pues los casos son mas complejos ya que incluyen las formas con ángulos de giro.

6. EQUIPO DE CÓMPUTO EMPLEADO.

El sistema de cómputo empleado para desarrollar las pruebas de simulación de la red neuronal contenía las siguientes características:

Hardware:

- Procesador AMD DURON de 1.8 Mhz
- Memoria RAM de 128Mb
- Disco duro de 40 Gb/10 Gb libres
- 4 puertos USB
- Memoria virtual automática

Software:

- Sistema operativo: Windows XP
- Simulador de redes neuronales: Matlab

Los tiempos de procesamiento en entrenamiento y simulación están en relación directa con este equipo de cómputo.

7. JUSTIFICACION DEL TIPO DE RED

El algoritmo Backpropagation es un algoritmo de aprendizaje supervisado, el cual necesita conocer cual es la salida esperada que actualiza pesos y ganancias siguiendo la regla de gradientes conjugados ya que esta modificación del algoritmo Backpropagation converge en muy pocas iteraciones, y es incluso uno de los algoritmos más rápidos para redes multicapas. Una de las mayores ventajas de las redes multicapa, y en especial de la red Backpropagation, es que pueden aproximar cualquier función si se escoge una adecuada configuración para la red y un adecuado número de neuronas en la capa oculta, escogencia que depende de la experiencia del desarrollador de la red. La red Bakcpropagation es un excelente aproximador de funciones, aunque es imposible determinar una configuración exacta de la red para cada aplicación. El proceso de aprendizaje no es fijo para ninguna red neuronal, el éxito consiste en probar con diferentes configuraciones hasta obtener la respuesta deseada; para esta aplicación se escogieron dos redes, una red 36:15:4, es decir que para un vector de entrada de 36 elementos y esperando 4 salidas de la red que indican la forma de una trayectoria, se tienen 15 neuronas en la primera capa oculta; la otra red es 36:10:7 y se encarga de encontrar el ángulo de desviación del trayecto.

8. SIMULACION Y ANALISIS DE RESULTADOS RNA2

En esta parte del documento se simularan en Matlab una red Backpropagation de dos capas, recordemos que para Matlab la capa de entrada no es considerada una capa, por lo tanto usamos: nodos de entrada, capa1 y capa2; además se mostraran las pruebas que se hicieron hasta encontrar un aprendizaje de la red aceptable para la aplicación. Para cubrir los objetivos de esta monografía solo simularemos dos patrones, la línea recta y la forma “L”.

Las variables **pl** y **sl** representan las entradas de entrenamiento para la línea recta y **pr** y **sr** las entradas para la forma L. la estructura y formación de estas matrices se muestran en el capítulo 5.

Empezaremos mostrando el algoritmo para crear la red (net) para en reconocimiento de los ángulos de inclinación para las trayectorias (ver anexo A).

```
%forma L
%entradas

pl1=[0  0   0   0   1   0....
     0  0   0   0   1   0....
     0  0   0   1   0   0....
     0  0   1   0   0   0....
     0  0   0   1   0   0....
     0  0   0   0   1   0]';

pl2=[0  1   0   0   0   0....
     0  0   1   0   0   0....
     0  0   1   0   0   0....
     0  0   1   0   0   0....
     0  1   1   0   0   0....
     1  1   0   0   0   0]';



pl3=[0  0   0   0   0   0....
     0  0   0   0   1   1....
     0  0   0   1   1   0....
     0  0   1   0   0   0....
```

```
     0    0    1    0    0    0....
     0    0    1    0    0    0]';


pl4=[0    0    0    0    0    0....
     1    1    1    1    0    0....
     0    0    0    1    0    0....
     0    0    0    1    0    0....
     0    0    0    1    0    0....
     0    0    0    1    0    0]';

pl5=[0    0    0    0    0    0....
     0    0    0    0    0    0....
     0    1    1    1    1    1....
     0    1    0    0    0    0....
     0    1    0    0    0    0....
     0    0    1    0    0    0]';

pl6=[0    1    0    0    0    0....
     0    0    1    0    0    0....
     0    0    1    1    0    0....
     0    0    0    1    0    0....
     0    0    1    0    0    0....
     0    1    0    0    0    0]';

sl1=[0
0
1
0
0
0
0];
sl2=[0
0
0
0
1
0
0];
sl3=[0
0
1
0
0
0
0];



sl4=[0
0
0
0
0
1
```

```
0];

sl5=[0
0
1
0
0
0
0];
sl6=[0
0
1
0
0
0
0];

%*************************************************************
%forma linea recta
%entradas

pr1=[0  0   1   0   0   0....
     0  0   1   0   0   0....
     0  0   1   0   0   0....
     0  0   1   0   0   0....
     0  0   1   0   0   0....
     0  0   1   0   0   0]';

pr2=[1  1   0   0   0   0....
     0  1   0   0   0   0....
     0  1   0   0   0   0....
     0  1   0   0   0   0....
     0  1   0   0   0   0....
     0  1   1   0   0   0]';

pr3=[0  0   0   0   1   1....
     0  0   0   0   1   0....
     0  0   0   1   0   0....
     0  0   0   1   0   0....
     0  0   1   0   0   0....
     0  1   0   0   0   0]';

pr4=[0  0   0   0   0   0....
     0  0   0   0   0   0....
     0  0   0   0   0   0....
     1  1   1   1   1   1....
     0  0   0   0   0   0....
     0  0   0   0   0   0]';

pr5=[0  0   0   0   0   0....
     1  0   0   0   0   0....
     1  1   1   0   0   0....
     0  0   0   1   1   1....
     0  0   0   0   0   0....
     0  0   0   0   0   0]';
```

```
%salidas

sr1=[0
     0
     0
     1
     0
     0
     0];


sr2=[0
     0
     0
     0
     1
     0
     0];


sr3=[0
     0
     1
     0
     0
     0
     0];


sr4=[0
     1
     0
     0
     0
     0
     0];


sr5=[0
     0
     0
     0
     1
     0
     0];

P=[pl1 pl2 pl3 pl4 pl5 pl6 pr1 pr2 pr3 pr4 pr5 ];
S=[sl1 sl2 sl3 sl4 sl5 sl6 sr1 sr2 sr3 sr4 sr5];
[Sb,Q] = size(S);
Sa = 10;
net = newff(minmax(P),[Sa Sb],{'logsig' 'logsig'},'traingda');
```

En esta función (newff) se introducen las características de la red como:

La matriz de máximos y mínimos de los parámetros de entrenamiento, las funciones de transferencia (o función de activación) para las dos capas de la red.

La utilización de la función 'logsig' (logarítmica sigmoidal) permite apreciar los valores de salida que estarían entre cero y uno, de esta forma identificaremos el comportamiento de la red a las respuestas deseadas.

A Continuación se realizara el entrenamiento.

```
%    TRAINING THE NETWORK %
net.trainParam.goal = 0.001;     % Sum-squared error goal.
net.trainParam.show = 20;        % Frequency of progress displays (in
                                                    epochs).
net.trainParam.epochs = 5000;    % Maximum number of epochs to train.
net.trainParam.lr=0.05;                % Rata de entrenamiento
[net,tr] = train(net,P,S);
```

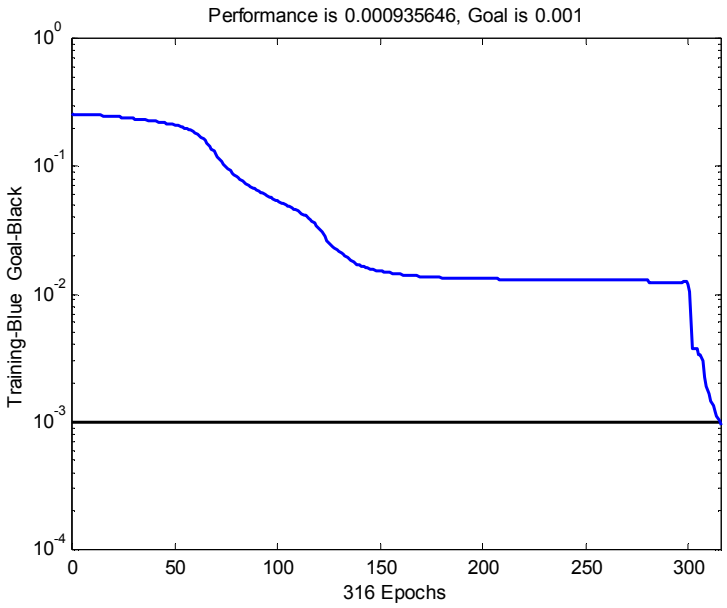


Figura 34. Error global vs iteraciones.

Para un error de 10^{-3} se hacen 316 iteraciones para realizar el entrenamiento, Esto se aprecia en la figura 28. Ya entrenada la red se harán las pruebas mediante la utilización de la función de Matlab: sim('red neuronal', 'matriz de prueba'), simula

la acción de la red ante una entrada determinada; empezaremos con las pruebas con las matrices con las cuales fueron entrenada la red.

sim(net,pl1)

ans =

0.0000

0.0009

0.9875

0.0390

0.0026

0.0002

0.0008

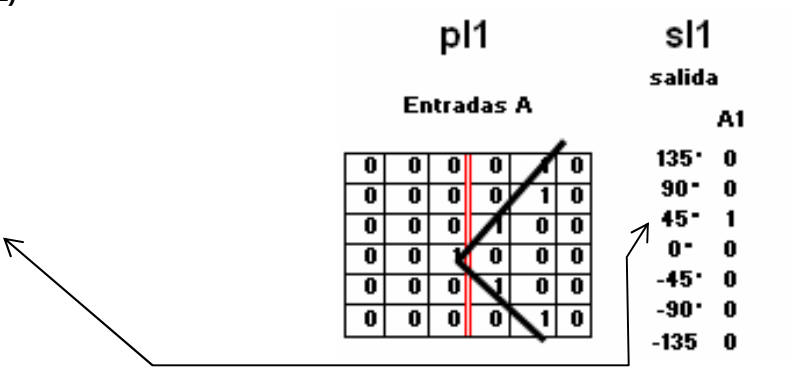


Figura 35. Entradas pl1 y salidas sl1 para RNA2

Para pl1 (primera matriz con la que se entreno la red) vemos una respuesta correcta de 0.9875, como lo muestra la flecha en la figura 29 donde hay una correspondencia entre el valosr arrojado por la función 'sim' y la matriz sl1 (matriz de entrada con la cual fue entrenada la red).

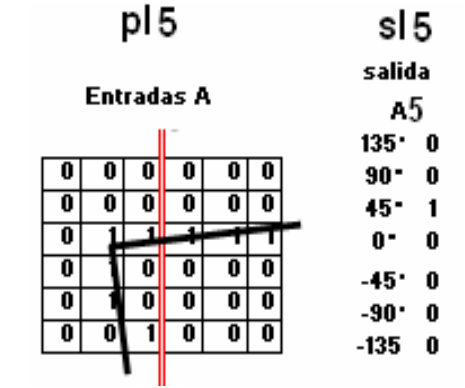
Un segundo caso esa mostrado en la figura 30, donde se vuelve a mostrar la correspondencia entre la salida de la simulación (*sim(net,pl5)*) y la salida esperada sl5.

sim(net,pl5)

ans =

0.0001

0.0000



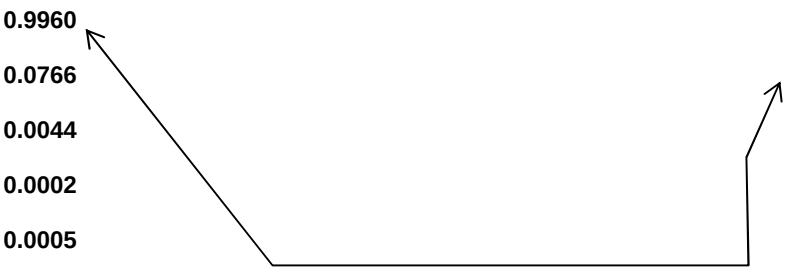


Figura36. Entradas pl5 y salidas sl5 para RNA2

En las figuras 31 y 32 se harán dos comprobaciones mas para el caso de reconocimiento de ángulo de desviación de un trayecto en forma de ‘L’.

```
sim(net,pl2)
ans =
   0.0003
   0.0009
   0.0021
   0.0073
   0.9684
   0.0000
   0.0002
```

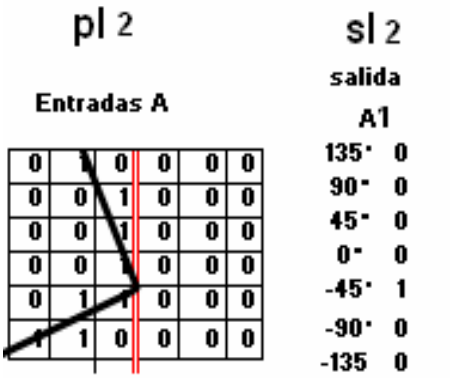


Figura 37. Entradas pl2 y salidas sl2 para RNA2

```
sim(net,pl4)
ans =
   0.0012
   0.0003
```

0.0004

0.0580

0.0008

0.9992

0.0005

Figura 38. Entradas pl4 y salidas sl4 para RNA2

Vemos que la red ha respondido correctamente con los datos con los cuales ha sido entrenada, ahora realizaremos simulaciones con parámetros de prueba (parámetros que no han sido usados en el entrenamiento), para ello se utilizaran las matrices de prueba (prueb1, prueb2, prueb3, prueb4 y prueb5). La figura 35 Muestra la matriz que se va ha simular correspondiente a una figura vista por el sensor, además de la salida esperada out1.

```
prueb1=[1  0  0  0  0  0....
        1  1  0  0  0  0....
        0  0  1  0  0  0....
        0  0  0  1  0  0....
        0  0  1  0  0  0....
        0  0  1  0  0  0];
```

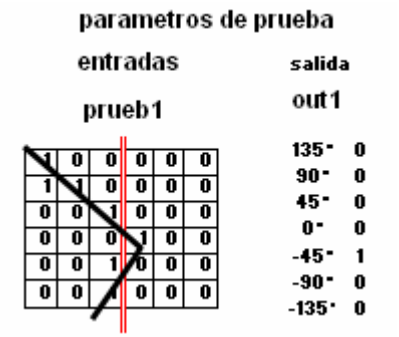


Figura 39. Entradas pl4 y salidas sl4 para RNA2

sim(net,prueb1)

ans =

0.0006

0.0018

0.0002

Para este caso la red tiene dificultades para reconocer este parámetro de entrada, por que el valor de la quinta fila (0.1917) debió ser mayor que todos los valores del resultado de la simulación.

0.1939

0.1917

0.0088

0.0001

Al fallar es este intento de reconocimiento se reestrenará la red introduciendo la matriz prueb1 como un nuevo dato de entrenamiento ,la siguiente grafica muestra el numero de iteraciones para conseguir un error de 10^{-3} .

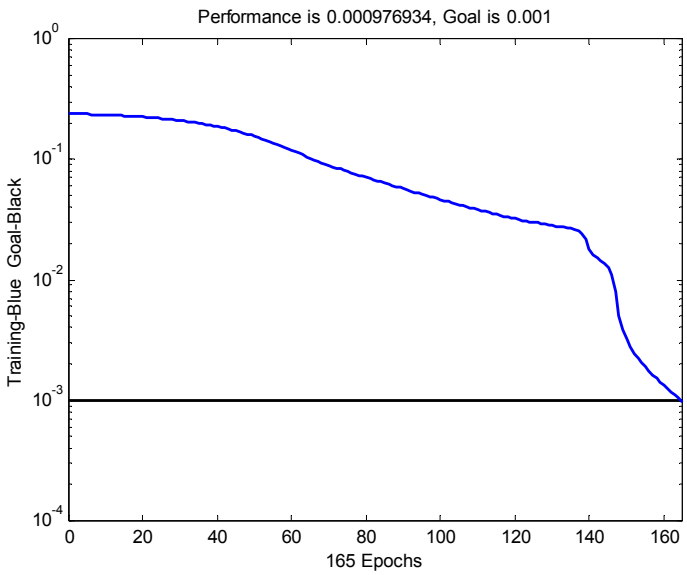


Figura 40. Error global vs iteraciones.

Después del reentrenamiento se prueba la red nuevamente con la matriz prueb1.

sim(net,prueb1)

ans =

0.0001

0.0019

0.0109

0.0128

0.9747

0.0288

0.0012

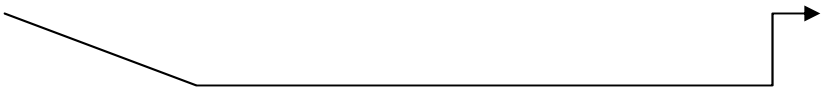


Figura 41. Parámetros de prueba prueb1.

La red ya esta interpretando la entrada para prueb1 como lo muestra la figura, ahora introduciremos otra entrada de prueba mostrada en la figura 38 para probar que tanto generaliza la red.

```
prueb2=[0  0   0   0   0   0....
        1  1   0   0   0   0....
        0  0   1   1   1   0....
        0  0   0   0   1   0....
        0  0   0   0   1   0....
        0  0   0   0   1   0]';
```

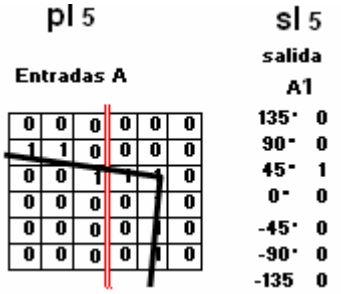


Figura 42. Entradas pl5 y salidas sl5 para RNA2

sim(net,prueb2)

ans =

0.0064
0.0024
0.4910
0.0145
0.1844
0.1725
0.0069

Para este caso la red muestra progresos de aprendizaje, vemos que en la tercera fila de la matriz de salida hay un dato de 0.491 (49%), este es el que representa menos error y es la salida valida para el reconocimiento de esta figura.

Veamos el comportamiento para una entrada con prueb3. mostrado en la figura

```
prueb3=[0  0   0   0   0   0....
        0  0   0   0   0   0....
```

```
     0  0   1   1   1   1....
     0  0   1   0   0   0....
     0  0   1   0   0   0....
     0  0   1   0   0   0]';
```

ans =

0.0936

0.0166

0.9723

0.0031

0.0037

0.0017

0.0065

Figura 43. Parámetros de prueba prueb3.

Nuevamente encontramos una confusión en la red, se están confundiendo los ángulos +90º y +45º ; se tratara de solucionar el problema con mas entrenamiento y luego verificaremos con otro caso de +90º

Introduciendo la matriz prueb3 como parámetro de entrenamiento tenemos:

```
prueb3=[0   0   0   0   0   0....
        0   0   0   0   0   0....
        0   0   1   1   1   1....
        0   0   1   0   0   0....
        0   0   1   0   0   0....
        0   0   1   0   0   0]';
sim(net,prueb3)

sprueb3=[0
     1
     0
     0
     0
     0
     0];

%reentrenamiento con prueb3.
P=[pl1 pl2 pl3 pl4 pl5 pl6 pr1 pr2 pr3 pr4 pr5 prueb1 prueb3];
S=[sl1 sl2 sl3 sl4 sl5 sl6 sr1 sr2 sr3 sr4 sr5 sprueb1 sprueb3];
[Sb,Q] = size(S);
Sa = 10;
net = newff(minmax(P),[Sa Sb],{'logsig' 'logsig'},'traingda');
%    TRAINING THE NETWORK %
net.trainParam.goal = 0.001;     % Sum-squared error goal.
net.trainParam.show = 20;      % Frequency of progress displays (in epochs).
net.trainParam.epochs = 5000;  % Maximum number of epochs to train.
net.trainParam.lr=0.05;              % Rata de entrenamiento
[net,tr] = train(net,P,S);
```

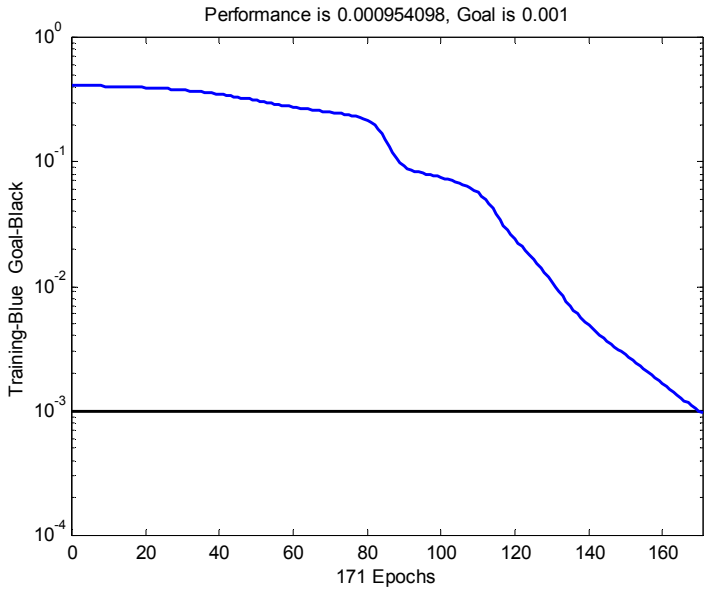


Figura 44. Error global vs iteraciones.

Después de este último entrenamiento se verifica el comportamiento para prueb3.

sim(net,prueb3)

ans =

0.0000

0.9309 *

0.0482

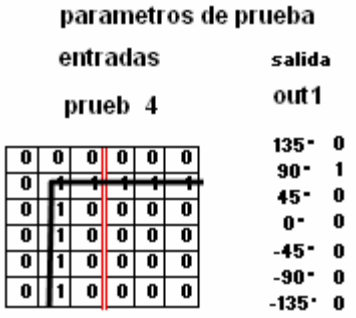
0.0505

0.0024

0.0334

0.0214

Después del entrenamiento la red acertó. A continuación se introducirá otra figura a +90º y trataremos de demostrar la capacidad de generalización de la red.



```
prueb4=[0  0   0   0   0   0....
        0  1   1    1   1   1....
        0  1   0   0   0   0....
        0  1   0   0   0   0....
        0  1   0   0   0   0....
        0  1   0   0   0   0]';
```

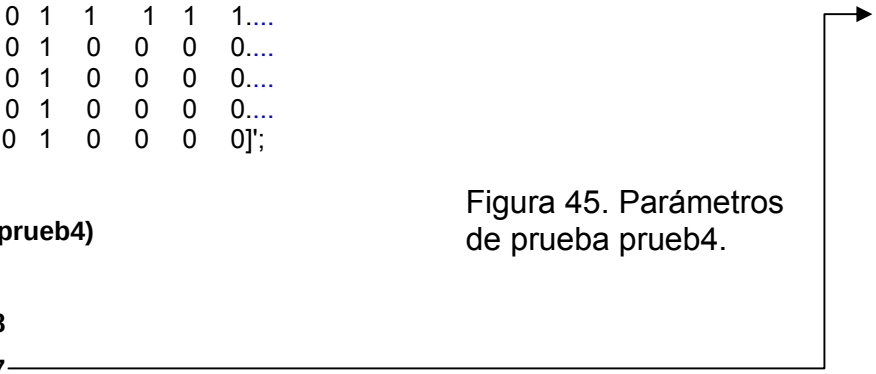


Figura 45. Parámetros de prueba prueb4.

```
sim(net,prueb4)
ans =
   0.0138
   0.0007
   0.5700
   0.0135
   0.3672
   0.0153
   0.0017
```

La red no ha podido con la identificación de la imagen, se optara por otro reentrenamiento, si después de este no hay generalización se cambiara algún parámetro de la red (como el numero de neuronas de la primera capa) para buscar un mejor comportamiento.

Introduciendo la matriz prueb3 como parámetro de entrenamiento tenemos:

```
prueb4=[0   0  0  0  0  0....
        0   1  1  1  1  1....
        0   1  0  0  0  0....
        0   1  0  0  0  0....
        0   1  0  0  0  0....
        0   1  0  0  0  0]';

sprueb4=[0
   1
   0
   0
   0
   0
   0];
```

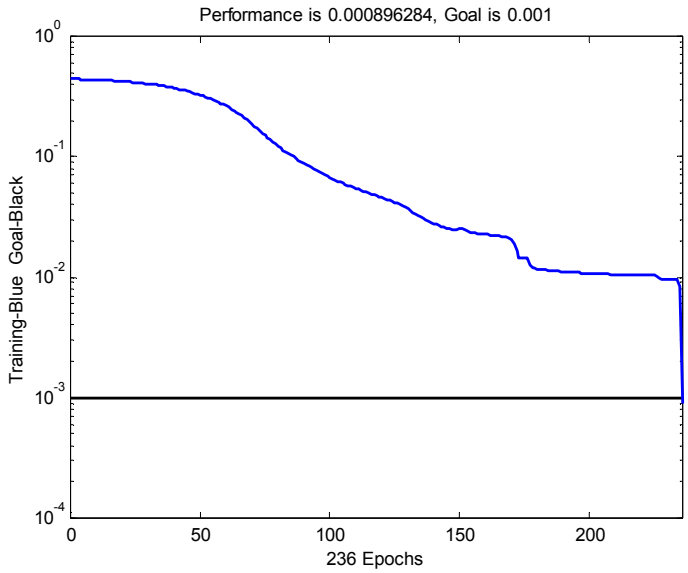


Figura 46. Error global vs iteraciones.

Para verificar el correcto funcionamiento de la red se ha realizado la simulación con prueb4, como lo muestra la siguiente figura, donde se muestra la salida deseada.

```
sim(net,prueb4)
ans =
   0.0003
   0.9427
   0.0152
   0.0097
   0.0153
   0.0660
   0.0067
```

Figura 47. Parámetros de prueba prueb4.

En esta quinta prueba intentaremos ver la capacidad de generalización de la red para una nueva forma “L” para los trayectos a 90º

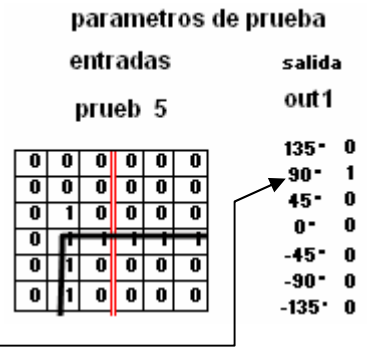
```
prueb5=[0   0  0  0  0  0...
        0   0  0  0  0  0...
        0   0  0  0  0  0...
```

```
        0   1  1  1  1  1....
        0   1  0  0  0  0....
        0   1  0  0  0  0]';
```

Figura 48. Parámetros de prueba prueb5.

```
sim(net,prueb5)

ans =
    0.0003
    0.9149
    0.1715
    0.0083
    0.0002
    0.0037
    0.0043
```



Como vemos la red esta respondiendo positivamente, ahora realizaremos pruebas para identificar los ángulos para la línea recta. Ver figura 49.

```
pr1=[0   0  1  0  0  0....
   0  0  1  0  0  0....
   0  0  1  0  0  0....
   0  0  1  0  0  0....
   0  0  1  0  0  0....
   0  0  1  0  0  0]';
```

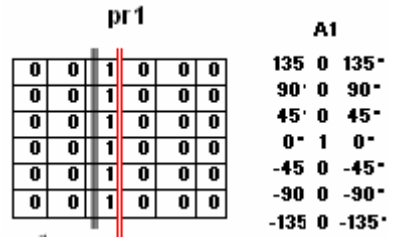


Figura 49. Parámetros pr1 para la línea recta.

```
sim(net,pr1)
ans =
   0.0130
   0.0096
   0.0035
   0.9635
   0.0080
```

Después de la simulación la red muestra un buen comportamiento para este caso.

0.0787

0.0004

sim(net,pr2)

ans =

0.0014

0.0030

0.0223

0.0014

0.9866

0.0105

0.0001

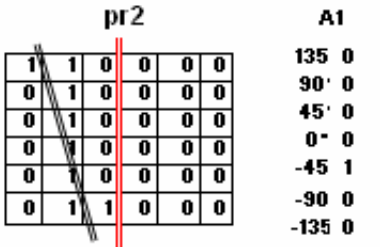


Figura 50. Parámetros pr2 para la línea recta.

Veamos como trabaja con las diagonales.Observemos las pruebas para ángulos positivos como lo muestra la siguinte figura.

```
prprueba1=[0   0  0  0  1  1....
           0   0  0  0  1  0....
           0   0  0  1  0  0....
           0   0  0  1  0  0....
           0   0  1  0  0  0....
           0   1  0  0  0  0]';
```

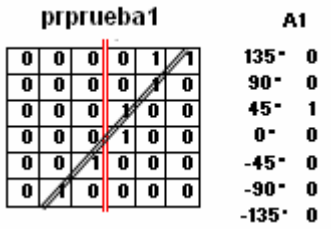


Figura 51. Parámetros prprueba1 para la línea recta.

Con este caso se demostrara la generalización de la red ya que la matriz prprueba1 no ha sido ingresada como parámetro de entrenamiento y ha tenido una salida de 45º.

sim(net,prprueba1)

ans =

0.0001

0.0049

0.9881

0.0060

0.0030

0.0291

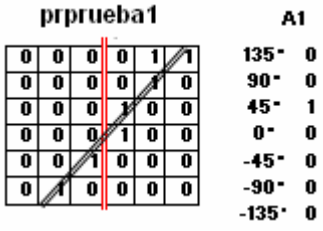


Figura 52. Parámetros prprueba1 para la línea recta.

0.0058

Se obtuvo un buen comportamiento de la red para el caso de la línea recta, a pesar de no hacer reentrenamiento, esto se puede justificar por la poca complejidad que presenta la línea recta

9. SIMULACION Y ANALISIS DE RESULTADOS RNA1

La RNA1 solo se encarga del reconocimiento de la forma de las trayectorias, se aclara que algunos de los parámetros para el entrenamiento de esta red se usaron

en el entrenamiento de la RNA2. A continuación se muestra el algoritmo usado en Matlab para el entrenamiento de la RNA1.

```
%forma linea recta
%entradas

fR1=[0   0  1  0  0  0....
     0   0  1  0  0  0....
     0   0  1  0  0  0....
     0   0  1  0  0  0....
     0   0  1  0  0  0....
     0   0  1  0  0  0]';

fR2=[0   0  0  1  0  0....
     0   0  0  1  0  0....
     0   0  0  1  0  0....
     0   0  0  1  0  0....
     0   0  0  1  0  0....
     0   0  0  1  0  0]';

fR3=[0   0  0  0  0  0....
     0   0  0  0  0  1....
     0   0  0  0  1  0....
     0   0  0  1  0  0....
     0   0  1  0  0  0....
     0   1  0  0  0  0]';

fR4=[1   0  0  0  0  0....
     0   1  0  0  0  0....
     0   0  1  0  0  0....
     0   0  0  1  0  0....
     0   0  0  0  1  0....
     0   0  0  0  0  1]';

%salidas

olR=[1
     0
     0
     0];
%*************************************************************

%entradas forma L

fL1=[0   0  0  0  0  0....
     0   0  0  0  0  0....
     1   1  1  0  0  0....
     0   0  1  0  0  0....
     0   0  1  0  0  0....
     0   0  1  0  0  0]';

fL2=[0   0  0  0  0  0....
     0   0  0  0  0  0....
     0   0  1  1  1  1....
```

```
     0   0  1  0  0  0....
     0   0  1  0  0  0....
     0   0  1  0  0  0]';

fL3=[0   0  0  0  0  0....
     0   0  1  1  1  1....
     0   0  1  0  0  0....
     0   0  1  0  0  0....
     0   0  1  0  0  0....
     0   0  1  0  0  0]';

fL4=[0   0  0  0  0  0....
     1   1  1  0  0  0....
     0   0  1  0  0  0....
     0   0  1  0  0  0....
     0   0  1  0  0  0....
     0   0  1  0  0  0]';

%SALIDA
olL=[0
0
0
1];
%entradas forma cruz

fC1=[0   0  1  0  0  0....
     0   0  1  0  0  0....
     1   1  1  1  1  1....
     0   0  1  0  0  0....
     0   0  1  0  0  0....
     0   0  1  0  0  0]';



fC2=[0   0  0  1  0  0....
     0   0  0  1  0  0....
     0   0  0  1  0  0....
     1   1  1  1  1  1....
     0   0  0  1  0  0....
     0   0  0  1  0  0]';

fC3=[0   0  0  1  0  0....
     0   0  0  1  0  0....
     1   1  1  1  1  1....
     0   0  0  1  0  0....
     0   0  0  1  0  0....
     0   0  0  1  0  0]';

fC4=[0   0  1  0  0  0....
     0   0  1  0  0  0....
     1   1  1  1  1  1....
     0   0  1  0  0  0....
     0   0  1  0  0  0....
     0   0  1  0  0  0]';
%salidas
olC=[0
```

```
1
0
0];
%entradas forma T
fT1=[0   0  1  0  0  0....
     0   0  1  0  0  0....
     0   0  1  0  0  0....
     0   0  1  1  1  1....
     0   0  1  0  0  0....
     0   0  1  0  0  0]';

fT2=[0   0  1  0  0  0....
     0   0  1  0  0  0....
     0   0  1  0  0  0....
     0   0  1  0  0  0....
     0   0  1  1  1  1....
     0   0  1  0  0  0]';

fT3=[0   1  0  0  0  0....
     0   1  0  0  0  0....
     0   1  1  1  1  1....
     0   1  0  0  0  0....
     0   1  0  0  0  0....
     0   1  0  0  0  0]';

fT4=[0   0  0  1  0  0....
     1   1  1  1  0  0....
     0   0  0  1  0  0....
     0   0  0  1  0  0....
     0   0  0  1  0  0....
     0   0  0  1  0  0]';
%salidas

olT=[0
0
1
0];

P=[fL1 fL2 fL3 fL3 pl1 pl2 pl3 pl4 pl5 fC1 fC2 fC3 fC4 fT1 fT2  fT3 fT4
fR1 fR2 fR3 fR4 pr1 pr2 pr3 pr4 pr5];
S=[olL olL olL olL olL olL olL olL olL olC olC  olC olC olT olT olT olT
olR olR olR olR olR olR olR olR olR];
[Sb,Q] = size(S);
Sa = 15;
net = newff(minmax(P),[Sa Sb],{'logsig' 'logsig'},'traingda');


%    TRAINING THE NETWORK %

%==================================

net.trainParam.goal = 0.01;   % Sum-squared error goal.

net.trainParam.show = 20;      % Frequency of progress displays (in
epochs).
```

```
net.trainParam.epochs = 5000;  % Maximum number of epochs to train.

net.trainParam.lr=0.05;        % Rata de entrenamiento
[net,tr] = train(net,P,S);
```

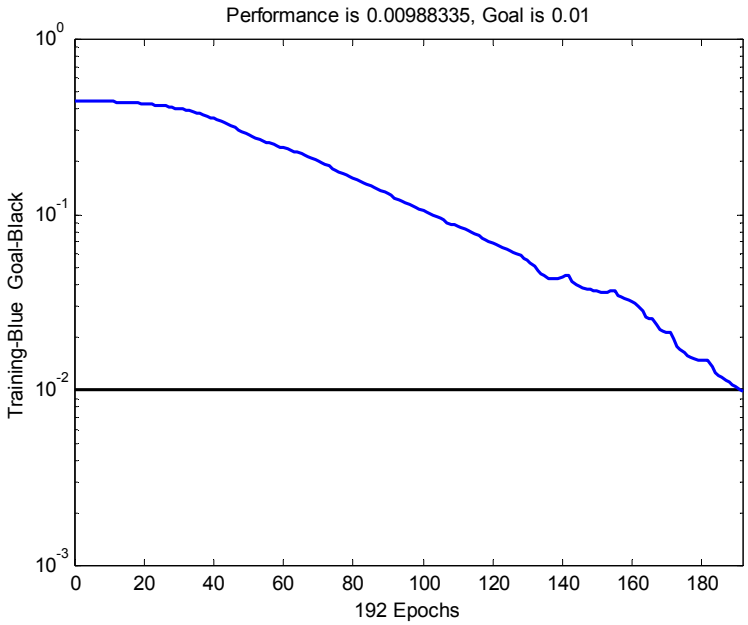


Figura 53. Error global vs iteraciones.

Ya entrenada la red se procede a probar esta con los parámetros entrenados.

Para la forma de línea recta tenemos las matrices mostradas en la siguiente tabla:

	Entradas fR1	Entradas fR2	Entradas fR3	Entradas fR4	Salidas deseadas (de entrenamiento) salidas oIR
Forma: I :	0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0	0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0	0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0	1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1	I : 1 + : 0 ⊢ : 0 ⌊ : 0

sim(net,fR1) ans =	sim(net,fR2) ans =	sim(net,fR3) ans =	sim(net,fR4) ans =	
I : *0.7720*	*0.9019*	*0.9780*	*0.9891*	Salidas simuladas
+ : 0.0013	0.0497	0.0027	0.0134	
⊢ : 0.2536	0.1715	0.0095	0.0881	
⌊ : 0.1238	0.0130	0.0570	0.0182	

T

Tabla 3. Entradas entrenamiento para una línea recta con sus respectivas salidas simuladas.

En el cuadro anterior se muestran las correspondencias entre los datos de entrenamiento y los datos esperados después de la simulación, los resultados de esta simulación aplican, ya que se encuentran dentro de lo esperado. Para comprobar la capacidad de generalización de la red se simularan valores no suministrados anteriormente en el entrenamiento de la red.

```
pruebafR1=[0   0  0  0  1  0....
           0   0  1  1  0  0....
           0   1  1  0  0  0....
           1   1  0  0  0  0....
           0   0  0  0  0  0....
           0   0  0  0  0  0]';
```

Figura 54. Parámetros pruebafR1 para la línea recta.

```
sim(net,pruebafR1)
ans =
    0.3389
    0.0483
    0.0048
    0.0337
```

Aquí se aprecia que el valor de 0.3389 tiene un menor error respecto a las otras salidas, por lo tanto aplica.

En la siguiente tabla se muestran las entradas usadas durante el entrenamiento para la identificación de la forma “L”, con las respectivas simulaciones.

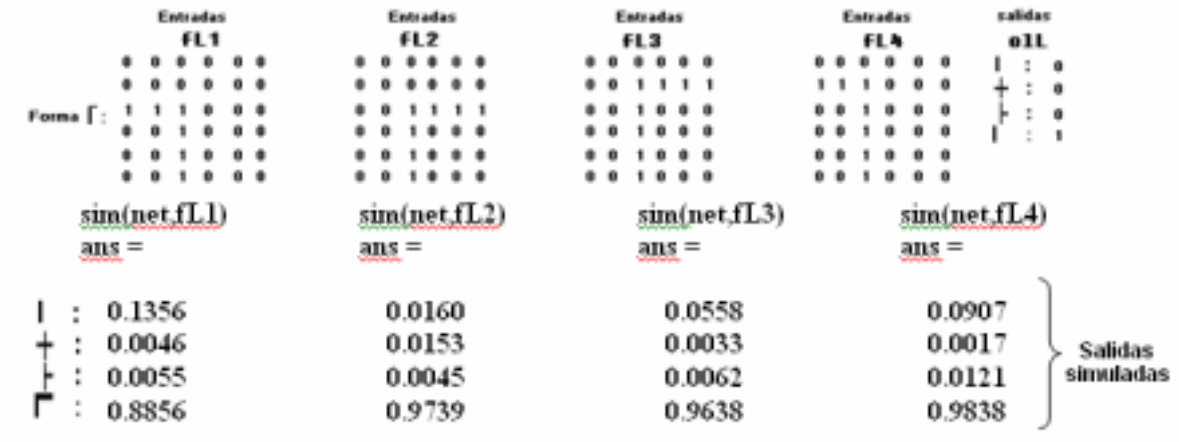


Tabla 4. Entradas entrenamiento para la forma “L” con sus respectivas salidas simuladas.

La simulación para este caso resulto exitosa ya que la red logro reconocer la forma “L” del camino. Para comprobar la capacidad de generalización de la red se introducirán valores pertenecientes a la forma “L” con cierto ángulo de inclinación.

```
pruebafL1=[0   0  0  0  0  0....
           1   1  0  0  0  0....
           0   0  1  1  1  0....
           0   0  0  0  1  0....
           0   0  0  0  1  0....
           0   0  0  0  1  0]';
```

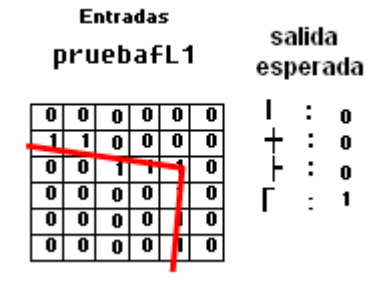


Figura 55. Parámetros pruebafL1 para la forma “L”.

sim(net,pruebafL1)

ans =

0.5413
0.1804
0.0071
0.9227

La red ha presentado un buen comportamiento, a pesar de poseer dos salidas con altos valores en la salida (“I” y “ ”) se identifica claramente que hay un valor que prevalece sobre el otro, este es el valor que representa la salida esperada para esta simulación.

10. CONCLUSIONES

El modelo empleado para el sistema de seguimiento de trayectoria para prototipos de robots con tracción en dos llantas a través de la tecnología de las redes neuronales, permitirá navegar sobre trayectorias de distintas formas de una

manera natural y segura, eliminando movimientos innecesarios y en forma de zig-zag..

La utilización de *los sensores de visión artificial* parecen adecuados en navegación robótica ya que se reduce el consumo de recursos como energía, procesado de datos, y espacio tan limitado en una plataforma móvil.

El sistema de visión artificial usado en esta monografía es de muy poca complejidad, este se puede mejorar aumentando el numero de pixeles y disminuyendo el tamaño del sensor, esto se puede lograr con la utilización de fibra óptica, debido a que la delgada fibra óptica reemplazarían en varias numero de veces el tamaño que ocupan los leds infrarrojos, de esta forma se puede tener las fibras ópticas como medio receptor de la luz como un sensor de poco tamaño mientras que los leds emisores y receptores se ubican en otro lugar. Esta monografía se ha ofrecido como un punto de partida para el desarrollo de futuros proyectos como el guiado de robots autónomos y el reconocimiento de formas en dos dimensiones. También ofrece un punto de partida para la mejora de los métodos usados en los sistemas de navegación de los vehículos autónomos como son las sillas de ruedas para pacientes discapacitados.

La sencillez de los casos de reconocimiento empleados en este proyecto ha colaborado con el éxito en el entrenamiento y posterior prueba de las figuras propuestas, sin embargo la limitación en el número de píxeles del sensor (36 píxeles) es la principal causa en los casos de confusión presentados en la

identificación de ángulos, mas específicamente para las identificaciones de desviación de trayectoria para la RNA2 en los casos de +90º, con un adecuado numero de píxeles, un buen set de entrenamiento y pruebas en la topología de la red usada tendríamos mejores resultados con un mayor numero de ángulos a identificar.

Al realizar la totalidad de los entrenamientos y pruebas para la RNA1 Y RNA2, tenemos la seguridad que el algoritmo Backpropagation es fácil de implementar, y tiene la flexibilidad de adaptarse para aproximar cualquier función, demostrando buena capacidad de generalización. La simulación y el entrenamiento realizado en Matlab ofrecen agilidad en el proceso de iteraciones hechas en la búsqueda de los errores globales empleados en el proyecto, esto se puede justificar en gran medida a la capacidad del equipo usado y a la utilización del algoritmo de optimización numérica Gradiente conjugado, cuya función en Matlab es 'traingda'; esta modificación del algoritmo Backpropagation converge en muy pocas iteraciones, y es incluso uno de los algoritmos más rápidos para redes multicapa
Es importante recalcar que no hay una regla específica para determinar el número de capas ocultas, ni el número de neuronas que debe contener cada una de ellas para un problema específico, para nuestro caso siempre se mantuvieron el número de neuronas pues no se encontró un numero de fallas significativas en las etapas de prueba.

Por la capacidad de generalización de la red que se ha demostrado anteriormente, el sistema puede interpretar curvas de diferentes ángulos, es decir, la red esta en

capacidad de de relacionar el forma de una curva con una trayectoria en forma de “L”, esto logra que el vehiculo haga sus funciones de corrección de trayectoria y control de velocidad en las curvas. Al tener mejoras en la captación de las imágenes (aumento del número de píxeles del sensor), se puede entrenar la red con casos particulares de curvaturas para el control de la velocidad del robot, entonces la red identificará que tanto esta pronunciada la curva; de esta manera que el robot puede saber que velocidad aplicara ante una determinada curva.

BIBLIOGRAFIA

Luis Jañez Esacalada's. Visión Artificial y Visión Humana, aplicación a la percepción Visual en Robótica [on line]. Copyright © 1997 by ITC. http://sirio.psi.ucm.es/PROYECTOS/VISIONROBOT/vavh.html

Alfredo Catalina Gallego. Introducción a las Redes Neuronales Artificiales. Marzo-Abril 1995. Revista del *Grupo Universitario de Informática* de la *Universidad de Valladolid*. Desarrollado por: http://www.gui.uva.es/login/13/redesn.html

Pablo esteles valencia. Selección de características para clasificadores Neuronales [on line]. Diciembre 1999 Chile. Universidad de Chile cipres.cec.uchile.cl/~em753/pdf/seleccion.pdf

José R. Hilera González. Sistema de reconocimiento óptico de caracteres (ocr) con redes neuronales [on line]. Departamento de Ciencias de la Computación Universidad de Alcalá de Henares. www.cc.uah.es/hilera/docs/1996/c_jiacse1/c_jiacse1.htm

Ing. Maria Isabel Acosta. Tutorial de redes neuronales [on line]. Universidad tecnológica de Pereira publicado en el año 2000. ohm.utp.edu.co/neuronales/Varios/

William Vallejo. Sistema de Visión Artificial para Determinar el Flujo de Café Soluble en un Prototipo de Aglomeración de Polvos, Mediante Redes Neuronales Artificiales [on line]. Publicado en monografías.com http://www.monografias.com/trabajos10/cafe/cafe.shtml#Intro

R. Martínez, C. Sagüés, J. J. Guerrero. Corrección visual de un robot móvil con homografías [on line]. Dpto. de Informática e Ingeniería de Sistemas, Universidad de Zaragoza. España 2002. webdiis.unizar.es/~rmcantin/ papers/Martinez-Cantin03JA.pdf.

Jose Antonio Boluda Grau. Vision log-polar y arquitecturas especiales de procesamiento de imágenes [on line]. Universidad de Valencia. Departamento de Informática. España 2001. tapec.uv.es/~jboluda/curso_doctorado/ curso_doctorado_vision_chips.pdf