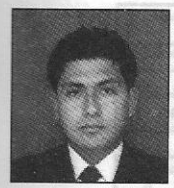
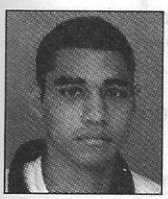
Identificación de formas geométricas y colores mediante procesamiento y reconocimiento de imágenes

Director: Giovanni Bermúdez



Jorge Ariel Díaz Cepeda
Tecnólogo en electrónica (2000), especialista en Control Electrónico e instrumentación (2001), ingeniero en Control Electrónico e Instrumentación (2003), Universidad Distrital Francisco José de Caldas, Facultad Tecnológica
Jadiam@hotmail.com



Miguel Ricardo Pérez Pereira
Tecnólogo en electrónica (2000), especialista en Control Electrónico e instrumentación (2001), ingeniero en Control Electrónico e instrumentación (2003), Universidad Distrital Francisco José de Caldas, Facultad Tecnológica.
perez_pereira_m@yahoo.com

Resumen

Este artículo describe un sistema de reconocimiento de patrones visuales que utiliza la red neuronal ART2 y se implementará en el robot RM-V1, un brazo articulado con 5 grados de libertad. Con una cámara Web, que actuará como sensor, se tomará una foto del medio externo con todas las figuras geométricas en desorden. Cada figura geométrica se introducirá en la red neuronal, la cual arrojará un resultado, de acuerdo con la figura que se analice. El usuario podrá elegir la figura que la red neuronal debe reconocer y que el manipulador robótico debe recoger.

Palabras clave
Reconocimiento de imágenes, formas geométricas, red neuronal

Introducción

El reconocimiento de imágenes está ligado directamente con la robótica y los diferentes campos que la contienen, específicamente el área industrial, pero también es útil en medicina y ciencias afines. En el desarrollo de esta tesis, se enfoca el reconocimiento a la utilización del Robot RM-V1[1], como alternativa práctica para demostrar la funcionalidad de una red neuronal.

Captura de la imagen

Al principio, en el desarrollo de la investigación, se pensó en controlar una cámara web desde su propio hardware, es decir, hacer el controlador, lo cual no es factible porque es necesario tener datos que sólo el fabricante posee.

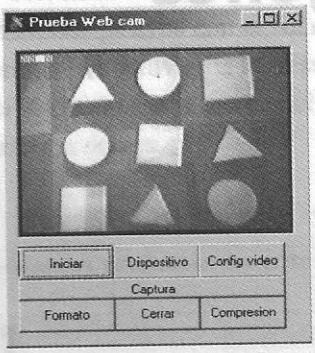


Figura 1. Software desarrollado para la captura de la imagen.

Se empezó con una *Webcam* de puerto paralelo. Después de analizar el problema, se dedujo que lo mejor era utilizar una librería OCX. Se estudiaron dos librerías que controlan *Webcams* (AXIS CameraServerControl y JPCsoftware); la óptima fue la distribuida por JPCsoftware, pues ofrece más opciones de control y mejores herramientas. A pesar de la eficacia de este control para capturar la imagen, ésta era de poca calidad, debido a la escasez de colores. Era urgente cambiar esa cámara por otra más reciente. En consecuencia se utilizó una cámara digital de última generación, con interfaz de puerto USB y mejor calidad de imagen, que arrojó mejores resultados. El software que se desarrolló (figura 1) para avanzar en esta etapa del proyecto, no presentó mayor complejidad, pues su finalidad era capturar una imagen y controlar el momento de la captura.

Explicación del programa

El programa de captura, llamado Prueba WebCam, tiene siete botones de control, (figura 1).

Iniciar: inicia el programa, cargando la configuración necesaria para el dispositivo. Es necesario para utilizar cualquiera de los otros botones.

Dispositivo: muestra una lista de los dispositivos de captura instalados en el PC, es decir, qué controladores posee. Se selecciona el que se desea utilizar.

Configuración video: abre una ventana que muestra todas las opciones de configuración del dispositivo, como brillo, realce, gama, etc. Estas opciones varían según el dispositivo seleccionando.

Captura: al oprimir este botón, la imagen en la ventana es capturada y guardada en un directorio creado para esa función.

Formato: muestra una lista de los formatos disponibles para guardar la imagen, así como la cantidad de bits y el tamaño. Para este caso, el valor predeterminado es 320X240 a 24 bits en formato BMP.

Cerrar: cierra el dispositivo de captura, más no el programa.

Compresión: muestra los diferentes tipos de compresión que soporta el dispositivo y que utiliza Windows.

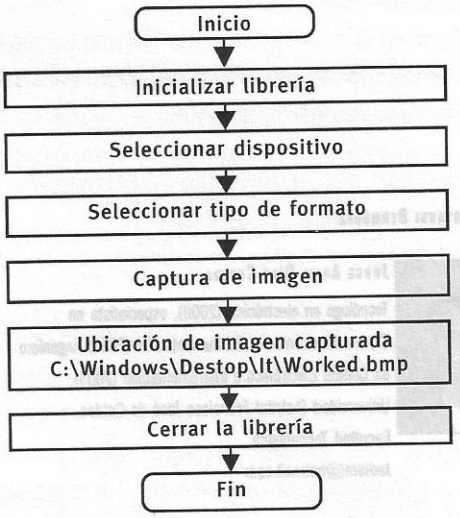


Figura 2. Diagrama de flujo para controlar la captura de la imagen con la cámara *Webcam*.

El algoritmo utilizado para el desarrollo del software es mostrado en la figura 2.

1 Brazo articulado de 5 ejes, casa Mitsubischi.

Tratamiento de la imagen

La información acerca del formato BMP fue relevante, pues los programas existentes para leer este tipo de archivo sólo muestran la imagen, pero no el valor de cada píxel. Se pensó utilizar un editor de texto, como el block de notas, o un programa matemático como Mathcad o Matlab, pero esto complicaría la situación del problema, pues habría que manejar distintos programas para controlar la cámara y para procesar la imagen. Al investigar más a fondo sobre este tipo de formato se encontró información concluyente para la elaboración de un programa que puede leer un archivo BMP y cargarlo en un vector (figura 3), con el valor exacto de cada píxel y la información que contiene el archivo BMP.

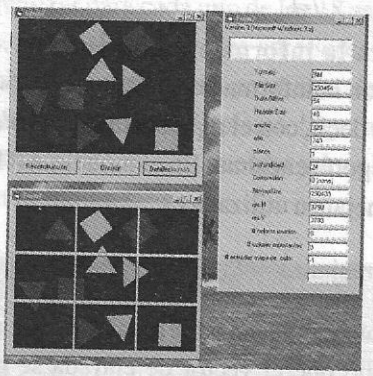


Figura 3. Software de lectura y escritura del formato BMP.

Debido a que el algoritmo de la red Neuronal debe procesar la imagen dividida en nueve cuadrículas, se utilizaron nueve vectores y se redibujaron en el orden adecuado para que la imagen conservara su perspectiva. El diagrama de flujo es mostrado en la figura 4.

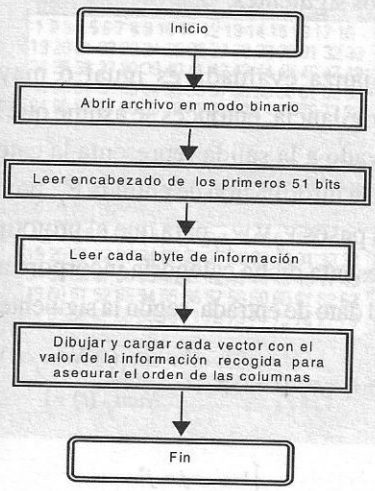


Figura 4. Diagrama de flujo utilizado para leer la información de la imagen capturada.

Comprobación del funcionamiento de la red neuronal ART2

La utilidad y el funcionamiento de la red ART2 pueden entenderse mejor con el siguiente ejemplo, adaptado de [Kung]. Se trata de utilizar una red ART2 para clasificar puntos del plano, de forma que la red establezca categorías *(clusters)* en función de la vecindad entre los puntos. Como el aprendizaje se lleva a cabo sin supervisión, las clases no se conocen *a priori.* Por tanto, debe crearlas la propia red quien las crea, generando los prototipos representantes de cada una de ellas. En este ejemplo, los prototipos representarán los centros de gravedad del conjunto de puntos incluidos en cada clase creada por la red. Se utilizan 200 puntos, que se representan mediante sus dos coordenadas *(x, y)* de valor real en el rango [0…5]. Se procede a presentar en la entrada de la red las coordenadas de los 200 puntos.

Para examinar la funcionalidad del ejemplo se desarrolló un programa que efectúa esta operación introduciendo varios puntos aleatoriamente, mostrando cada vez resultados diferentes (figura 5). En este ejemplo se ha establecido un valor para el parámetro de vigilancia ñ = [1…7], siendo 7 el máximo posible.

Figura 5. Programa realizado para comprobar la funcionalidad de la red ART2.

Resultados de la red neuronal

Para llegar al correcto desarrollo de la red neuronal (en el problema que concierne a esta tesis), se debió hacer cambios en su estructura. Éstos no afectaron los principios y lineamientos de la red ART, sino que la ayudaron a adaptarse con más facilidad, y originaron una nueva evolución de la red neuronal ART2, la cual fue nombrada como ART2

(rfg). La sigla rfg correponde a reconocimiento de figuras geométricas.

Cambios realizados a la red ART2

Los cambios hechos a la red ART2 dieron origen a una variante de esta misma, que arrojó mejores resultados.

Estos cambios son:

- Suprimir la habilidad de crear nuevas clases. Como las clases se conocen con anterioridad, la red no necesita crear prototipos nuevos, sino centrarse en los prototipos ya existentes para la búsqueda del carácter a reconocer.
- Parámetro de comparación ρ. En la red ART2 convencional este parámetro es fijo. Cuando se introduce el vector para su reconocimiento, la red muestra un valor de ρ. Si dicho valor es superior o inferior al preestablecido, se determina la pertenencia de la clase, en la nueva red, como ya están las clases armadas. El vector de entrada se compara con todas las clases preestablecidas y se guardan todos los parámetros ρ; después se busca la clase que dio el parámetro de menor valor y el ejemplo se asocia a esta clase.

Funcionamiento de la ART2 (rfg)

Teniendo en cuenta los cambios introducidos en la red neuronal, el algoritmo de funcionamiento también sufre algunas modificaciones. Dicho algoritmo queda de la siguiente manera:

- Se presenta una información de entrada $E_k = (e_1^{(K)}, ..., e_N^{(K)})$ a la red. En este caso, los componentes $e_i^{(k)}$ son valores reales.
- Cada neurona (ne_i) de la capa de entrada recibe el valor del componente correspondiente real del vector E_K y lo envía a todas las neuronas de la capa de salida a través de las conexiones correspondientes (W_{ji}).
- Cada neurona (ns_j) de la capa de salida compite con las demás de esta capa hasta que sólo una permanece activa. Aquí se plantea otra diferencia con respecto al modelo discreto, ya que en este caso se supone que la neurona vencedora es aquella j^* que verifica una mínima diferencia (distancia euclidiana) entre el patrón de entrada y los pesos de las conexiones entre esta neurona y las de la capa de entrada:

$$S_{ns_j} = \begin{cases} 1 \text{MIN} \left\| E_K - W_j \right\| = \text{MIN} \left(\sum_{i=1}^{N} e_i^{(k)} - W_{ji} \right) \\ 0 \end{cases}$$

La neurona vencedora (ns_{j^*}) envía su salida (1) a través de las conexiones hacia atrás (v_{ij^*}). Cada neurona i-ésima de la capa de entrada recibe el valor

$$x_i = \sum_{j=1}^{M} V_{ij} S_{ns_j} = V_{ij^*} = w_{j^* i}$$

- Se compara la información de entrada $.....,e_N^{(k)})$ con la información $x = (x_1,...,x_n) = (w_{j^*1},...,w_{j^*N})$ recibida; es decir, con el representante o prototipo de la clase o categoría enésima. En el modelo ART2, la relación de semejanza utilizada es:

$$\text{Relación de semejanza} = \left\| E_K - X \right\| = \left\| E_K - V_{j^*} \right\| = \sum_{i=1}^{N} \left| e_i^{(K)} - w_{j^* i} \right|$$

En este punto empieza la diferencia, ya que no se tomó la decisión, sino que el vector de entrada se prueba con todos los vectores prototipo y se van guardando los parámetros ρ resultantes. Con cada vector prototipo se repiten los pasos 1, 2, 3, 4, 5.

Después se determina qué vector prototipo dio el parámetro de vigilancia menor y, sobre este vector, se le aplican los pasos siguientes.

Si la semejanza evaluada es igual o mayor que el parámetro de vigilancia, entonces se asume que la neurona que se ha activado a la salida representa la categoría más apropiada para la información de entrada E_K, procediéndose al ajuste de los pesos v_{ij} y $w_{j^* i}$ para que el prototipo almacenado que representa dicha categoría incorpore algunas características del dato de entrada, según la siguiente expresión:

$$w_{j^* i}(t+1) = v_{ij^*}(t+1) = \frac{e_i^{\,k} + w_{j^* i}(t).Num_{j^*}(t)}{Num_{j^*}(t)+1}$$

$$S_{ns} = \begin{cases} 1 & j = j^* \\ 0 & j \neq j^* \end{cases}$$

Siendo $Num_{j^*}(t)$ el número de vectores de entrada que han sido considerados hasta el instante t como de la clase j^*.

Almacenamiento de los vectores en archivo TXT

Debido que a la red neuronal le toma bastante tiempo aprender, cada vez que se inicia el programa de entrenamiento, es necesario guardar los pesos de la red neuronal y cargarlos en vectores. La forma más simple es guardar estos valores en un archivo de tipo TXT. Lo complicado está a la hora de recargar los vectores, con la información guardada en el archivo TXT, pues hay que leer número a número el archivo y, de hecho, cada espacio. Este proceso se tiene que realizar para cada uno de los 18 archivos de texto generados. Al igual que para las anteriores partes del proyecto, también se elaboró un programa demo para comprobar la funcionalidad del mismo (figura 6). El código del programa utilizado para guardar y leer los vectores se muestra en la figura 7.

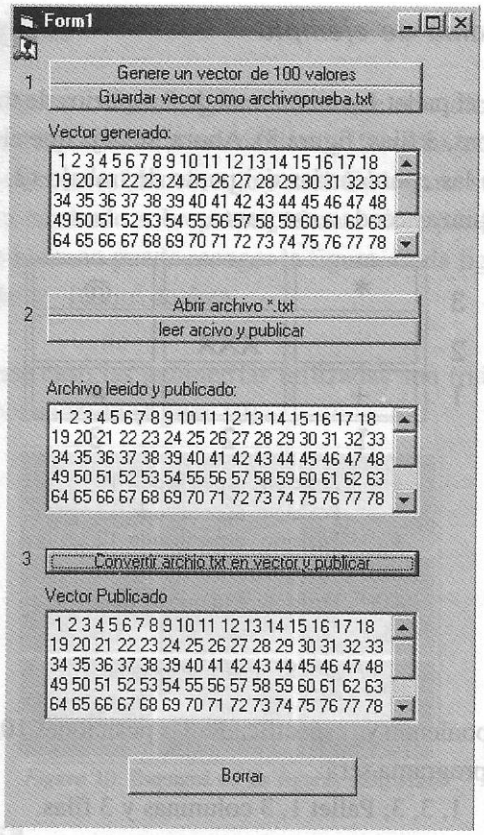


Figura 6. Sofware de lectura y escritura desde un archivo TXT.

```
        CommonDialog1.Filter = «Todos los archivos (*.*)|*.*|Archivos de
texto» &_
          «(*.txt)|*.txt|Archivos por lotes (*.bat)|*.bat»
          ' Especificar el filtro predeterminado
          CommonDialog1.FilterIndex = 2
          ' Presentar el cuadro de diálogo Abrir
          CommonDialog1.ShowOpen
          ' Presentar el nombre del archivo seleccionado
          MsgBox CommonDialog1.FileName
          d = CommonDialog1.FileName

          Exit Sub

        ErrHandler:
          ' El usuario ha hecho clic en el botón Cancelar
          Exit Sub
        End Sub

        Private Sub Command4_Click()
        Text2.Text = «»

        Open d For Input As #1
        Do While Not EOF(1)  ' Repite el bucle hasta el final del archivo.
        'Text3.Text = Text3.Text & Input(1, #1)
        g = g & Input(1, #1) ' guarda en esta variable g el contenido del file.txt
        conta = conta + 1
        Loop
        Close #1
        Text2.Text = g

        End Sub

        Private Sub Command5_Click() ' convierte el archivo txt a un vector,
                    del tamaño determinado
        Dim vec
        Dim k, col, kk, seg
        Dim conta2, z As Integer
        z = 1

        For k = 2 To conta
          z = k
          vec = Mid(g, z, 1)
            If vec = Chr(32) Or k = conta Then
              z = k - conta2 + 1
                If conta2 = 1 Then: z = 2: conta2 = 2
              col = col + 1
              vec = Mid(g, z, (conta2 - 1))
              vector2(0, col) = vec
              conta2 = 0
              Text3.Text = Text3.Text & « « & vector2(0, col)
            End If
          conta2 = conta2 + 1
        Next k
        End Sub

        Private Sub Command6_Click()
        Text1.Text = «»
```

```
Text2.Text = «»
Text3.Text = «»
g = «»
conta = 0
End Sub

Dim vector(0, 100) As Double
Dim aa, b, c
Dim conta As Integer

Private Sub Command1_Click()
Dim b As Integer
For b = 0 To 99
c = c + 1
vector(0, b) = c
aa = aa & « « & vector(0, b)
Next b
Text1.Text = aa
c = 0
End Sub

Private Sub Command2_Click()
Set fs = CreateObject(«Scripting.FileSystemObject»)
Set a = fs.CreateTextFile(«c:\mis
documentos\archivoprueba.txt», True)
a.WriteLine (aa)
a.Close

End Sub
Private Sub Command3_Click()
  ' Establecer CancelError a True
  CommonDialog1.CancelError = True
  On Error GoTo ErrHandler
  ' Establecer los indicadores
  CommonDialog1.Flags = cdlOFNHideReadOnly
  ' Establecer los filtros
```

Figura 7. Código del programa de lectura y escritura de un archivo TXT.

Este programa genera un vector de 100 valores y lo guarda como un archivo de extensión txt; luego es leído, y cada vez que se encuentra un chr (32)[2], carga el valor leído en archivo en el vector determinado.

El manipulador

Para la implementación del proyecto en el cual el robot tiene que recoger las figuras ya reconocidas por la red neuronal, se utilizó una función que tiene incluido el *set* de instrucciones del robot. Esta función permite manejar una grilla virtual, calculada automáticamente por la unidad de control. Haciendo uso de esta propiedad, se programó una grilla virtual de 9 cuadrículas, es decir, 3 filas y 3 columnas, donde la trayectoria para recoger las figuras no es relevante.

La función PT (pallet)

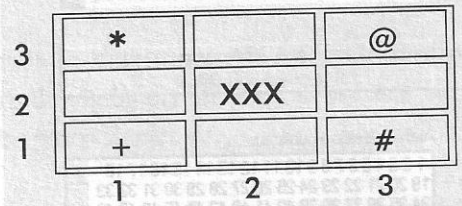
Define las coordenadas de una rejilla de puntos en un número de pallet concreto e identifica las coordenadas como el número de posición correspondiente al número de pallet especificado (con PA).

Este comando calcula las coordenadas de un punto de rejilla de un número de pallet especificado. Identifica las coordenadas con el número de posición correspondiente al número de pallet especificado. El comando PA[3] debe ser ejecutado antes de ejecutar este comando. Después de que el comando PT ha sido ejecutado, son borrados los datos de posición definidos anteriormente para el número de posición de la tarjeta.

Para definir la rejilla de un número de pallet concreto, hay que establecer sus cuatro esquinas. Los contadores de pallet deben de ser activados adecuadamente (con SC) para especificar un punto concreto del pallet.

Programa de ejemplo

Sea el pallet 1, con un total de 9 espacios de trabajo (3 columnas, 3 filas, figura 8). Ahora, se deja que el sistema calcule las coordenadas del punto de trabajo (2, 2), para que la pinza acceda a ese punto.



+ : Posición especificada 10
* : Posición especificada 11
# : Posición especificada 12
@: Posición especificada 13
XXX Posición a la que se quiere ir

Figura 8. Pallet de ejemplo.

Suponiendo ya especificadas las posiciones 10, 11,12 y 13, el programa será:

PA 1, 3, 3; Pallet 1, 3 columnas y 3 filas.

[2] El carácter 32, en el código ASCII, representa un espacio en blanco.

[3] Pallet Assign. Define el número de puntos de una rejilla en la dirección de columnas y filas, para un número concreto de pallet.

SC[4] 11, 2; Columna 2, se agrega 2 en el contador 11.
SC 12, 2; Fila 2, se agrega 2 en el contador 12.
PT 1; Calcula la poción (2, 2) del pallet.
MO[5] 1; Vaya a esa posición.

En el programa demo (figura 9), que trabaja con el control del robot, se pueden variar las propiedades de velocidad y posición del mismo. Este programa sirve también para calibrar la posición de la mesa y el tamaño de la foto con respecto al tamaño de la cámara.

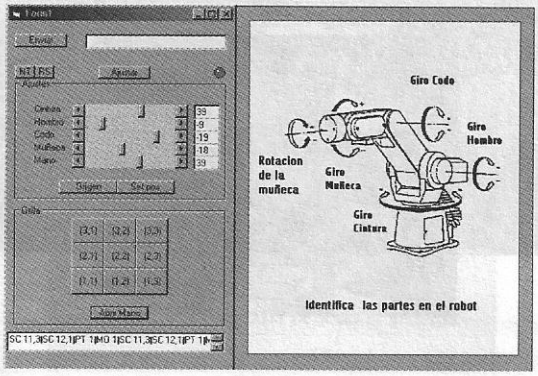


Figura 9. Programa demo para manejar el robot.

Las figuras

Las figuras o formas geométricas que se utiliza en el proyecto fueron hechas en madera y forradas en papel Contact de colores blanco, rojo y azul. En la parte superior tienen un cuadrado sobrepuesto, el cual sirve para que la pinza de la mano pueda recoger la figura. Cada figura no tiene más de 10 cm^2 de área.

En este caso las geometrías utilizadas son triángulo, círculo y cuadrado (figura 10).

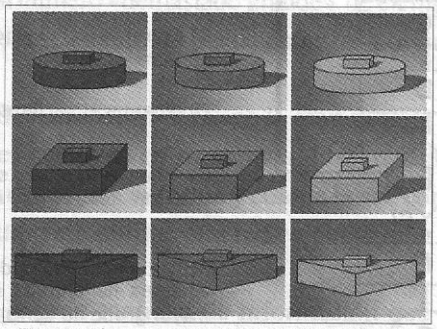


Figura 10. Esquema de las figuras utilizadas.

La mesa

Fue necesario construir una mesa, ajustable manualmente, donde se puede variar la altura y la posición de la cámara digital. Esta mesa tiene un entrepaño negro que se utiliza como fondo de la imagen capturada, es totalmente opaco y ayuda evitar la reflexión de la luz (figura 11).

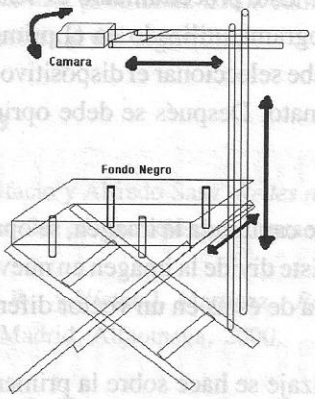


Figura 11. Mesa de trabajo.

Resultados generales

Al agrupar cada una de las partes desarrolladas y descritas anteriormente, se obtuvieron dos paquetes de software. En el primero se entrena la red neuronal; en el segundo se aplica el aprendizaje al reconocimiento en conjunto con el robot.

Software de aprendizaje

Está constituido por cuatro de las cinco partes mencionadas en este artículo: la captura, el procesamiento de la imagen, la red neuronal y el almacenamiento de los vectores. Las ventanas de ejecución son mostradas en la figura 12.

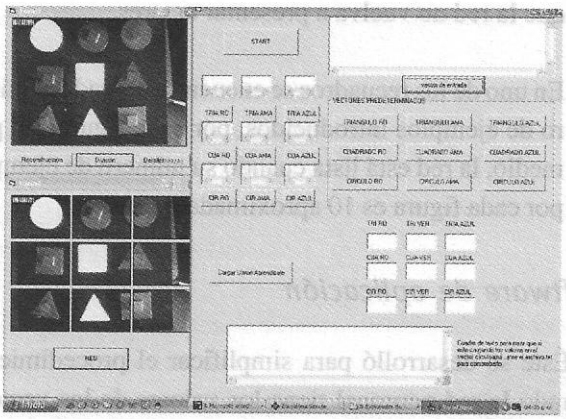


Figura 12. Software de aprendizaje.

[4] Set Counter. Carga determinado valor en un contador concreto.

[5] Move. Mover el final de la mano a una posición especificada

A continuación se explicará el procedimiento utilizado en este software para el aprendizaje de la red neuronal.

1. La captura. Este procedimiento es sencillo, pues es el mismo programa utilizado en el primer módulo. Primero se debe seleccionar el dispositivo deseado y, luego, el formato. Después se debe oprimir el botón de captura.

2. Después de capturada la imagen, se oprime el botón de división. Éste divide la imagen en nueve grillas iguales y carga una de éstas en un vector diferente.

3. El aprendizaje se hace sobre la primera de las grillas, donde se ubica la figura para crear un prototipo. Luego, se oprime el botón RED.

4. El botón RED abre una nueva ventana, donde se encuentran ubicados los botones de cada una de las figuras que se van introduciendo a la red. Cuando se presiona el botón de la figura correspondiente, se crea automáticamente el prototipo de la misma y se deshabilita el botón hasta que se introduzcan todos los prototipos.

5. Introducidos los prototipos, se coloca cualquier figura en el primer recuadro, se oprime el botón de vector de entrada, y se le da *start*. La red arroja el valor más cercano a la figura introducida; si el resultado es erróneo, manualmente se le dice a la red qué figura se introdujo en realidad. Este procedimiento se repite hasta que la red no vuelva a presentar errores.

6. En uno de los recuadros se encuentra ubicado el número de ejemplos introducidos, por cada figura. En promedio, la red está lista cuando su número de ejemplos por cada figura es 10 aproximadamente.

Software de aplicación

Éste se desarrolló para simplificar el procedimiento. Cuando la red neuronal tiene los valores de los vectores adecuados, éstos son cargados en este nuevo paquete, en el cual sólo se tiene que indicar cuál figura necesita ser reconocida y recogida por el robot.

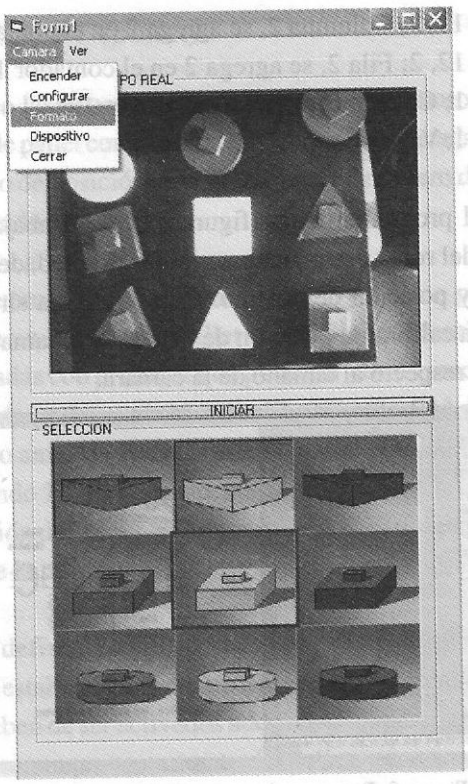


Figura 13. Software de aplicación.

Diagrama de flujo general

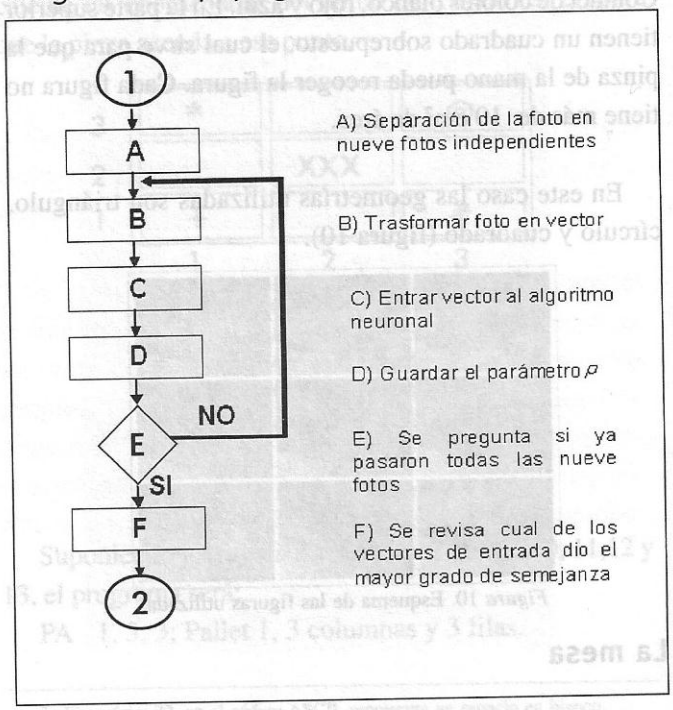


Figura 14. Algoritmo general del proyecto.

Hay que tener en cuenta que los vectores de comparación o clases en la red son sólo nueve, y fueron introducidos a la red de forma manual y con antelación.

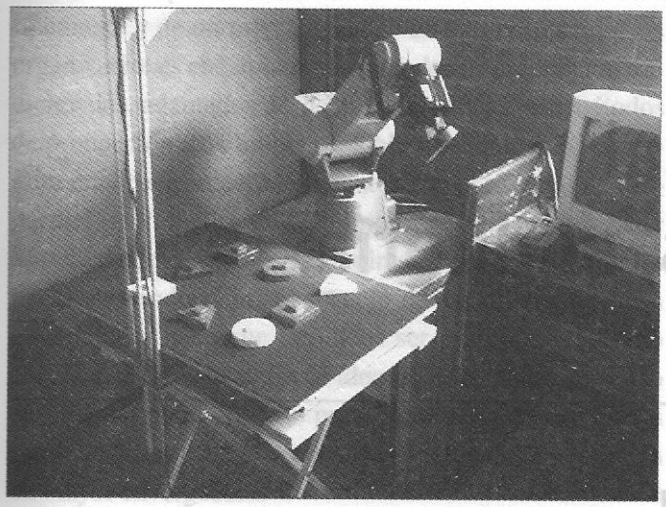


Figura 15. Fotografía de todo el sistema.

Conclusiones

- Es fundamental tener un medio ambiente estable, en especial lo relacionado con la iluminación, pues la luz debe ser uniforme y constante para evitar cambios en los valores de entrada del vector después de la captura de la imagen.
- Cuando la red es entrenada, los valores de ρ tienden a disminuir proporcionalmente. Esto hace suponer que, con un número «demasiado grande» de ejemplos aprendidos, la red neuronal puede empezar a olvidar.
- La red Neuronal ART2 tiene la gran ventaja de aceptar valores análogos en su entrada. Por consiguiente, es capaz de reconocer forma y color a la vez con el mismo algoritmo sin utilizar otra red en paralelo.
- Dada la gran cantidad de información que maneja el software, los vectores son muy grandes, y es necesario guardarlos. Cuando la información es cargada de nuevo en la red, el tiempo de procesamiento de la carga demora entre 40 segundos y 1 minuto.
- Los cambios realizados al algoritmo de la red neuronal ART2 fueron indispensables, debido a que el algoritmo original no cumplía los requerimientos necesarios para la solución de este problema. Estos cambios no afectaron la idea original del algoritmo.

Referencias

[1] Martín, Bonifacio y Alfredo Sanz. *Redes neuronales y sistemas difusos.* 2a. ed. Barcelona, Alfaomega-Rama, 2003.

[2] Hilera, José R. y Víctor J. Martínez. *Redes neuronales artificiales*. Madrid, Alfaomega, 2000.

[3] Hecht-nielsen R. Neurocommputing: Picking the Human Brain. This paper appears in: *Spectrum, IEEE*, 25,36-41, march, 1988.

[4] Kohonen T. An Introduction to Neural Computing. Vol. 1, 3-6, 1988.

[5] Maren A.J., C.T .Harston y R.M. Pap. Handbook of Neural Computing Applications. Academic Press, 1990.

[6] Grossberg S. How Does the Brain Buid a Cognitive Code? Psychological Review, 87, 1-51, 1980.

[7] Torres Fernando, Jorge Pomares, Pablo Gil, Santiago T. Puente y Rafael Aracil. *Robots y sistemas sensoriales*, Prentice Hall, 2002.